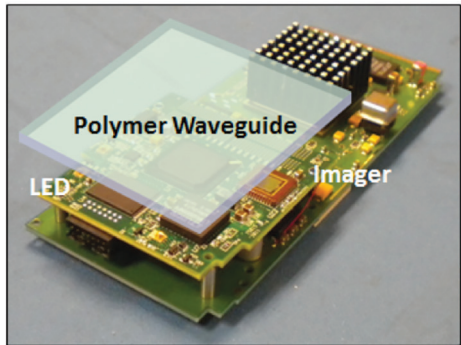


LINCOLN LABORATORY *JOURNAL*  
MIT Lincoln Laboratory  
244 Wood Street  
Lexington, MA 02420-9108



### Securing embedded systems

A physical unclonable function (PUF) implemented on a printed circuit board provides for the unique identification of and cryptographic key derivation for an embedded system. The PUF adds an LED and an imager to a printed circuit board, which is then coated with a polymer waveguide. The PUF secures the boot process for the system.

Page 110

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# LINCOLN LABORATORY *Journal*

Volume 22 ■ Number 1 ■ 2016

## CYBER SECURITY

Innovative cyber R&D  
at Lincoln Laboratory

- Reverse engineering to find vulnerabilities
- Discovering cyber threats via social media
- Developing a secure, resilient cloud
- Exploring moving target techniques

Volume 22 ■ Number 1 ■ 2016

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
**LINCOLN LABORATORY** *Journal*

Gregory P. Hamill Editor  
Megan Cichon Editor  
Dorothy S. Ryan Editor  
Ariana L. Tantillo Editor  
Kari L. Siegle Graphic Artist  
Alicia A. LaDuke Administrator

**COMMUNICATIONS AND COMMUNITY OUTREACH OFFICE**

David R. Granchelli, Manager

The *Lincoln Laboratory Journal* (ISSN 0896-4130) is published twice a year by Massachusetts Institute of Technology, Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420-9108. Subscriptions are free of charge, but provided only to qualified recipients (government employees and contractors, libraries, university faculty, and R&D laboratories). Requests for individual copies, subscriptions, or permission to reprint articles should be submitted to the Subscription Coordinator, Room S3-106, at the above address. The editorial offices can be reached at (781) 981-4204 or at [journal@ll.mit.edu](mailto:journal@ll.mit.edu).

**Journal Online:** Selected articles from back issues, starting in 1988, are available at [www.ll.mit.edu/publications/journal/journal.html](http://www.ll.mit.edu/publications/journal/journal.html). Contents of the current issue will be posted on this site approximately 45 days following print publication.

This material is based upon work supported by the U.S. Department of the Air Force under Air Force Contract Nos. FA8721-05-C-0002 and FA8702-15-D-0001. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

**Postmaster:** Please send address changes to the above address.

©2016 Massachusetts Institute of Technology. All rights reserved.



Follow MIT Lincoln Laboratory online.

Facebook: MIT Lincoln Laboratory (Official)

LinkedIn: <http://www.linkedin.com/company/mit-lincoln-laboratory>

Twitter: @MITLL



### On the Cover

This depiction of Lincoln Laboratory's network traffic during a few minutes of one day was generated by the patented visualization software tool Network Observatory, developed by the Florida Institute for Human and Machine Cognition (IHMC). In this visualization, time is depicted along the vertical axis; packets are represented by dots whose colors indicate the locations from which the data originate and whose density corresponds to file size; and the top and bottom of the display denote the world and the systems within the Laboratory, respectively. Using a variety of software tools designed by both in-house developers and outside commercial and research organizations, analysts protect Lincoln Laboratory networks by carefully monitoring and analyzing *all* traffic coming to and going from the Laboratory and by blocking malicious traffic.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
**LINCOLN LABORATORY Journal**

■ VOLUME 22, NUMBER 1, 2016 ■

**4 Cyber Security Research at Lincoln Laboratory**

This issue of the *Lincoln Laboratory Journal* focuses on the innovative work Lincoln Laboratory conducts in the research, development, evaluation, and deployment of cyber-resilient components and systems to help ensure successful national security missions.

**Marc A. Zissman and Robert K. Cunningham**



**Lab Notes**

**5 Securing Data**

A novel cryptographic technology simplifies the process of securing data used in military applications.

**11 Keeping an Eye on Cyber Threats**

The use of actual network data enables the development of robust, realistic cyber security tools.

**13 Training the Cyber Defensive Line**

A game-like competition is helping build experts in cyber “disaster response.”



**16 Can a Game Teach Practical Cyber Security?**

Lincoln Laboratory’s Capture the Flag challenge is designed to help college students learn cyber defense techniques.

**19 Recruiting the Next Generation of Cyber Security Specialists**

Two Lincoln Laboratory outreach activities are encouraging high-school students to pursue careers in cyber security.



## Features



### 24 Advanced Tools for Cyber Ranges

Cyber ranges are used to model the high volume of network traffic and the cyber attacks encountered via the Internet. Lincoln Laboratory has developed a variety of tools to automate cyber range operations, increase the fidelity of emulated network traffic, and visualize range activity.

**Timothy M. Braje**

### 33 Threat-Based Risk Assessment for Enterprise Networks

Lincoln Laboratory has created a network security model to guide the development of assessments that identify and prioritize cyber security risks. For the most important cyber threats, the researchers have designed practical risk metrics that can be computed automatically and continuously from security-relevant network data.

**Richard P. Lippmann and James F. Riordan**

### 46 Finding Malicious Cyber Discussions in Social Media

Today's analysts manually examine social media networks to find discussions concerning planned cyber attacks, attacker techniques and tools, and potential victims. Applying modern machine learning approaches, Lincoln Laboratory has demonstrated the ability to automatically discover such discussions.

**Richard P. Lippmann, William M. Campbell, David J. Weller-Fahy, Alyssa C. Mensch, Giselle M. Zeno, and Joseph P. Campbell**

### 60 Cloudbreak: Answering the Challenges of Cyber Command and Control

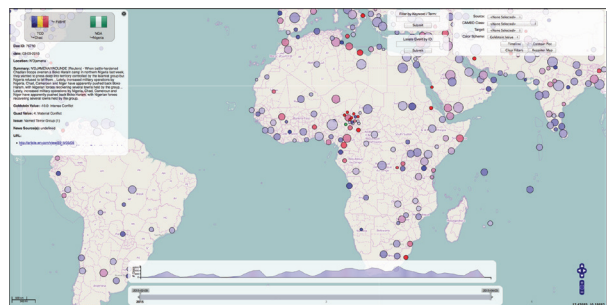
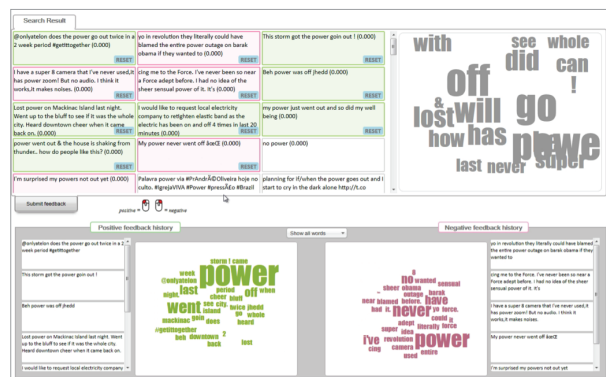
Cloudbreak, a flexible, user-centered framework for the development of command-and-control systems, allows the rapid prototyping and insertion of new capabilities for existing systems.

**Diane Staheli, Vincent F. Mancuso, Matthew J. Leahy, and Martine M. Kalke**

### 74 Recommender Systems for the Department of Defense and Intelligence Community

Recommender systems, which selectively filter information for users, can hasten analysts' responses to complex events such as cyber attacks. Lincoln Laboratory's research on recommender systems may bring the capabilities of these systems to analysts.

**Vijay N. Gadepally, Braden J. Hancock, Kara B. Greenfield, Joseph P. Campbell, William M. Campbell, and Albert I. Reuther**



## 90 Repeatabe Reverse Engineering with the Platform for Architecture-Neutral Dynamic Analysis

Researchers at Lincoln Laboratory have developed a platform to facilitate software reverse engineering that can lead to profound insight into how software behaves. This tool was recognized with a 2015 R&D 100 Award for being one of the year's 100 most innovative technologies.

**Ryan J. Whelan, Timothy R. Leek, Joshua E. Hodosh, Patrick A. Hulin, and Brendan Dolan-Gavitt**

## 100 Moving Target Techniques: Leveraging Uncertainty for Cyber Defense

Cyber moving target techniques involve randomizing cyber system components to reduce the likelihood of successful attacks, adding dynamics to a system to shorten attack lifetime, and diversifying otherwise homogeneous collections of systems to limit attack damage. A review of five dominant categories of cyber moving target techniques assesses their benefits and weaknesses.

**Hamed Okhravi, William W. Streilein, and Kevin S. Bauer**

## 110 Secure Embedded Systems

Lincoln Laboratory's co-design approach for simultaneously incorporating functionality and cyber security in an embedded system uses a security coprocessor to cryptographically ensure system confidentiality and integrity while maintaining functionality.



**Michael Vai, David J. Whelihan, Benjamin R. Nahill, Daniil M. Utin, Sean R. O'Melia, and Roger I. Khazan**

## 123 Secure and Resilient Cloud Computing for the Department of Defense

Lincoln Laboratory is developing technology that will strengthen the security and resilience of cloud computing so that the Department of Defense can confidently deploy cloud services for its critical missions.

**Nabil A. Schear, Patrick T. Cable, Robert K. Cunningham, Vijay N. Gadepally, Thomas M. Moyer, and Arkady B. Yerukhimovich**

## 136 Securing the U.S. Transportation Command

The U.S. Transportation Command moves soldiers, equipment, and supplies around the world. To help ensure that this critical supply chain is functioning efficiently, Lincoln Laboratory is working with the command to develop a software architecture that will provide an enterprise network with ample computational power, strong cyber security, and resiliency to attacks and disruptions.

**Jeffrey M. Diewald, Kajal T. Claypool, Jesslyn D. Alekseyev, George K. Baah, Uri Blumenthal, Alfred Cilcius, William L. Pughe, Joseph A. Cooley, Robert K. Cunningham, Jonathan R. Glennie, Edward F. Griffin, and Patrick J. Pawlak**

## Transitions

## 153 Technology Transfer

A roundup of Lincoln Laboratory technology transfer opportunities in cyber security.

# Cyber Security Research at Lincoln Laboratory

Marc A. Zissman and Robert K. Cunningham

Department of Defense missions increasingly are fought in and through the cyber domain. While significant efforts have been made to defend U.S. assets, processes, and data, adversaries have proven adept at stealing data and disrupting operations. Lincoln Laboratory conducts research, development, evaluation, and deployment of cyber-resilient components and systems designed to ensure successful national security missions despite cyber attack and exploitation. This issue of the *Lincoln Laboratory Journal* focuses on some of this innovative work.



As we write the introduction to this *Lincoln Laboratory Journal*, the Laboratory and many U.S. government civilian, military, and contractor communities are trying to understand the implications of the incidents at the U.S. Office of Personnel Management (OPM), where reports indicate that records containing the personally identifiable information of more than 21 million individuals were stolen. Even after this incident is studied and appropriate remediation steps are applied, confidence in the United States' near-term ability to maintain the confidentiality of sensitive information is likely to remain low.

Many cyber incidents reported in the popular press are similar to OPM-like breaches, i.e., breaches in the confidentiality of enterprise-class information systems; however, the cyber security problem is actually much broader than the narrow issue of sensitive information leaked from large servers. While one of the main goals of cyber security is to maintain *confidentiality* to ensure that only authorized parties can access certain data, another important goal is to maintain the *integrity* of the data so that the data are correct, only authorized parties can change them, and approved individuals can track those changes. A third goal is to guarantee the *availability* of data so they are accessible when necessary.

While ensuring the confidentiality, integrity, and availability of enterprise-level computing and data is important, the Department of Defense (DoD) must also be concerned with the security of its computing systems that interface with physical systems at the “tactical edge.” The DoD operates many computing systems in places where network connectivity may be intermittent, latency

may be long, bandwidth may be limited, the physical environment could be harsh, and the risk of overrun might be high. These systems include, for example, satellite communication terminals, tanks, aircraft, and wearable devices that contain embedded computing and networking. If the DoD is to have confidence that its warfighting missions will succeed, it must establish proper controls on the confidentiality, integrity, and availability of the computing and data of these tactical-edge systems.

From a technical perspective, the Laboratory believes that the DoD and other national security organizations face six major challenges when operating in the cyber domain:

- Understanding and operating in the overlap between the cyber domain and other physical domains (land, air, sea, and space). Planners need to design and operators need to execute integrated, effective operations across all domains, and they need to understand how actions taken in one domain can affect other domains.
- Assessing and quantifying cyber protection and cyber effects. Many national security organizations require processes for conducting threat-based, objective, quantitative cyber assessments and need the ability to prove that systems are secure and that missions will succeed against specific cyber threats.
- Building and maintaining realistic environments for force and capability development. The nation needs cyber ranges and other infrastructure to conduct scalable, repeatable, scientific, realistic and inexpensive testing, training, and mission rehearsal and to develop appropriate concepts of operations.
- Developing effective cyber situational awareness, decision support tools, and command-and-control systems. These systems need to provide commanders, analysts, and operators a full understanding of “blue” (i.e., United States), “red” (i.e., adversary), and “grey” (i.e., other) cyber status. They must also provide details on how that cyber status impacts national security missions. Operators also need tools that establish appropriate control of relevant cyber assets.
- Architecting, designing, and building cyber-resilient systems. Cyber systems must be capable of continuing support for a mission even in the face of cyber attacks that cannot be anticipated or stopped.
- Identifying means for compromise and exploitation of adversary cyber systems so that adversary missions are less likely to succeed.

In addition to these six technical challenges, the Laboratory is also mindful that the nation faces several critical nontechnical cyber security challenges, e.g., ensuring cyber operations policy is rational and permits effective, efficient execution, and equipping and training a well-organized cyber force. The Laboratory asserts that progress against all these challenges should make it more expensive for adversaries to compromise our missions via actions in the cyber domain; we seek to force adversaries to expend significantly more of their resources to achieve their goals while we keep U.S. missions assured at modest cost.

Lincoln Laboratory has been developing technology to support the cyber security objectives of the DoD, intelligence community, and law enforcement for more than 15 years, and our work generally focuses on the six technical challenges previously outlined. In this issue of the *Lincoln Laboratory Journal*, we will discuss some of our work addressing four of these challenges:

#### **ASSESSING AND QUANTIFYING CYBER PROTECTION AND CYBER EFFECTS**

Richard Lippmann and James Riordan have developed threat-based metrics for assessing the effectiveness of security measures established in large enterprise networks. Their article “Threat-Based Risk Assessment for Enterprise Networks” outlines their approach and describes the impact the metrics are having within U.S. government departments.

#### **BUILDING AND MAINTAINING REALISTIC ENVIRONMENTS FOR FORCE AND CAPABILITY DEVELOPMENT**

Capable cyber ranges are required to assess the performance and effectiveness of cyber tools and concepts of operations. In his article titled “Advanced Tools for Cyber Ranges,” Timothy Braje defines a modular architecture for cyber range software and describes instantiations of several cyber range software components, including those that can provide range automation, robust traffic generation, and range activity monitoring.

#### **DEVELOPING EFFECTIVE CYBER SITUATIONAL AWARENESS, DECISION SUPPORT TOOLS, AND COMMAND-AND-CONTROL SYSTEMS**

In their article titled “Cloudbreak: Answering the Challenges of Cyber Command and Control,” Diane Staheli, Vincent Mancuso, Matthew Leahy, and Martine Kalke

describe a successful process that deploys next-generation command-and-control tools to DoD combatant commands through the use of easily developed, assembled, and extended modular building blocks. These tools support operations in the cyber and physical domains at many of the regional and functional combatant commands worldwide.

Cyber attackers often use social media networks to discuss cyber security tools, cyber attacks, cyber defenses, and potential victims for targeted attacks. If analysts examining the discussions find potential threats, they can alert system administrators. With this information, administrators can better detect, defend against, and recover from future attacks. In their article titled “Finding Malicious Cyber Discussions in Social Media,” Richard Lippmann, William Campbell, David Weller-Fahy, Alyssa Mensch, Giselle Zeno, and Joseph Campbell outline their work in applying modern machine learning approaches to find cyber-related discussions in social media.

Processing large datasets and rapidly recommending an effective response to a cyber attack are vexing problems. In their paper “Recommender Systems for the Department of Defense and Intelligence Community,” Vijay Gadepally, Braden Hancock, Kara Greenfield, Joseph Campbell, William Campbell, and Albert Reuther discuss an approach to acquiring and processing information to provide timely, prioritized responses for analysts.

#### **ARCHITECTING, DESIGNING, AND BUILDING CYBER-RESILIENT SYSTEMS**

To permit more effective and efficient analysis of vulnerabilities in software systems, Ryan Whelan, Timothy Leek, Joshua Hodosh, Patrick Hulin, and Brendan Dolan-Gavitt have developed an open-source platform for repeatable reverse engineering of software systems. In their article titled “Repeatable Reverse Engineering with the Platform for Architecture-Neutral Dynamic Analysis,” the authors describe the system’s architecture and several applications.

In their article titled “Moving Target Techniques: Leveraging Uncertainty for Cyber Defense,” Hamed Okhravi, Kevin Bauer, and William Streilein discuss techniques for randomizing cyber system components to increase the workload on a cyber attacker. Many such techniques have been described in the literature, and the authors review the strengths and weaknesses of those techniques.

Lincoln Laboratory has developed a methodology for the co-design of functionality and security within embedded systems. These new systems are designed to be more resilient to cyber attacks. Michael Vai, David Whelihan, Benjamin Nahill, Daniil Utin, Sean O’Melia, and Roger Khazan describe this architecture and its uses in their article “Secure Embedded Systems.”

Over the past few years, researchers have developed cloud computing services that offer substantial benefits to users, such as the ability to store and access massive amounts of data, to deliver computing services on demand, to widely share information, and to scale resource usage. Lincoln Laboratory is developing technology that will strengthen the security and resilience of cloud computing so that the DoD can confidently deploy cloud services for its critical missions. This work is described by Nabil Schear, Patrick Cable, Robert Cunningham, Vijay Gadepally, Thomas Moyer, and Arkady Yerukhimovich in their article “Secure and Resilient Cloud Computing for the Department of Defense.”

Lincoln Laboratory work has been leveraged to support several U.S. government missions, including the U.S. Transportation Command, which moves soldiers, equipment, and supplies worldwide in support of the U.S. military. To support this mission, infrastructure is being upgraded to make it more efficient and secure. In “Securing the U.S. Transportation Command,” Jeff Diewald, Kajal Claypool, Jesslyn Alekseyev, George Baah, Uri Blumenthal, Alfred Cilius, William Pughe, Joseph Cooley, Robert Cunningham, Jonathan Glennie, Edward Griffin, and Patrick Pawlak describe the process of enhancing the mission with a more secure architecture and detail threats to mission success.

In addition to these articles, we start this issue with a set of quick-read Lab Notes that cover cyber technology and its uses, as well as cyber security education. “Keeping an Eye on Cyber Threats” describes how we use tools developed at the Laboratory to protect the Laboratory’s networks, and “Keeping Secrets Secure” describes an award-winning technology that Laboratory researchers developed to dramatically simplify the challenging problem of cryptographic key management. The Laboratory’s education efforts range from training for military officers, as described in “Training the Cyber Defensive Line,” to gaming for undergraduate students, as detailed



in “Can a Game Teach Practical Cyber Security?” to motivating high-school students to learn about the cyber field, as depicted in “Recruiting the Next Generation of Cyber Security Specialists.”

Much work remains to ensure that critical national security missions are resilient to cyber exploitation and attack. The work presented in this issue of the *Lincoln Laboratory Journal* includes reference architectures, new technologies, and deployed prototype systems that should help the United States progress down the right path. ■

#### About the Authors



**Marc A. Zissman** is the associate head of the Cyber Security and Information Sciences Division. He joined the Laboratory in 1983, and his early research focused on digital speech processing. In the late 1990s, he began his cyber security work by joining the Defense Advanced Research Projects Agency (DARPA)–sponsored

Lincoln Laboratory research team. The team designed and executed the first quantitative, objective, repeatable assessment of computer network intrusion-detection systems. He has since served in a series of Laboratory leadership roles, including developing and leading the execution of a strategic plan to establish the Laboratory’s cyber security mission area. He has held several advisory positions to the U.S. government and the North Atlantic Treaty Organization (NATO), e.g., he has been serving as a member of the Army Science Board since 2011. He holds a bachelor’s degree in computer science and a bachelor’s degree, master’s degree, and doctorate in electrical engineering from the Massachusetts Institute of Technology.



**Robert K. Cunningham** is the leader of the Secure Resilient Systems and Technology Group. He is responsible for initiating and managing research and development programs in information assurance and computer and platform security. His early research at Lincoln Laboratory focused on machine learning, digital image processing,

and image and video understanding. As part of this effort, he contributed to early drafts of the real-time message passing interface (MPI/RT) specification. Later, as a member of the technical staff in the Information Systems Technology Group, he pursued system security research and development, initially investigating intrusion-detection systems that do not require advance knowledge of the method of attack, then moving on to consider detection and analysis of malicious software. He has patented security-related technology, presented and published widely, and served as general chair or program chair for many conferences and workshops. He has also served on several national panels, such as the U.S. Army Cyber Materiel Development Strategy Review Panel, and led national teams, such as the National Security Agency’s working group for computer network defense research and technology transition. He holds a bachelor’s degree in computer engineering from Brown University, a master’s degree in electrical engineering from Boston University, and a doctorate in cognitive and neural systems from Boston University.

# Lab Notes

NEWS FROM AROUND LINCOLN LABORATORY

## CRYPTOGRAPHY

### Securing Data

A novel technology simplifies secure military communications and has the potential to be beneficial for a wide array of applications

**The Department of Defense** (DoD) military strategy relies in part on the development of advanced system technologies that can enable new capabilities for warfighters in the field. For example, imagine the advantages of a new drone or small satellite that can see through foliage and deliver high-resolution, tactical imagery. Because these new technologies are promising, developers focus on creating the systems' major subcomponents quickly, leaving important considerations like cyber security on the back burner. When developers eventually turn to incorporating security features, they often face several complications because they are so far along in the design lifecycle. At this stage, redesigning the system to add security features

typically results in crippled system usability, major design delays, superficial security, and large cost overruns.

To address this problem, researchers at Lincoln Laboratory are developing new tools, including a software component known as the Lincoln Open Cryptographic Key Management Architecture (LOCKMA).<sup>1</sup> This software quickly and inexpensively simplifies the task of securing data and communication in a wide variety of systems and may even be employed during later stages of the design cycle.

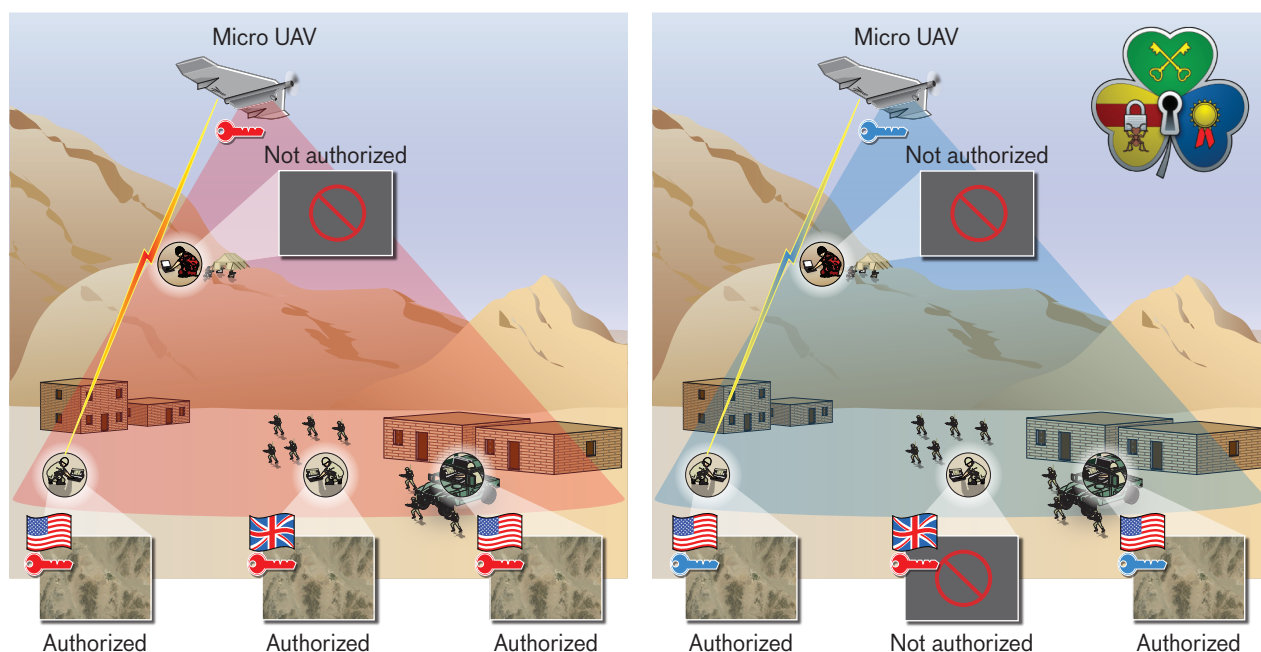
Cryptography is a vital tool for passing sensitive information to intended recipients and keeping that same information from prying eyes. Through the encryption of data into an unintelligible sequence of characters called ciphertext, a sender scrambles a message with an algorithm and a cryptographic key, and the intended recipient decrypts, or unscrambles, the message by using a symmetric, i.e. used for both encryption and decryption of data,

<sup>1</sup>Lincoln Open Cryptographic Key Management Architecture (LOCKMA) is available for licensing through the MIT Technology Licensing Office (TLO) under MIT case number 16575L. For more information about LOCKMA and LOCKMA-related patents, contact the TLO at [tlo-atto@mit.edu](mailto:tlo-atto@mit.edu).

algorithm and key. Though several encryption solutions are widely used to secure data today, they all have one major shortfall: key management.

Key management is the process of managing cryptographic keys, that is, generating secure cryptographic keys, making them available to authorized users, and storing them. It is arguably the most difficult aspect of cryptography, says Daniil Utin of the Secure Resilient Systems and Technology Group at Lincoln Laboratory, because developing a new key management scheme may inadvertently introduce security vulnerabilities caused by system bugs and development oversights. "Developers make key management systems that combine low-level cryptographic functions into a secure design that supports high-level security functions; it is a complicated process. During the design process, developers can sometimes unintentionally create an insecure system. Even a small bug in the key management system, such as a biased random number generator that enemies can easily exploit, can create a big security vulnerability," says Utin.

Some key management solutions rely on manual key distribution. For example, if two military units plan to send encrypted radio messages to each other, they must first download cryptographic keys onto a Key Processor computer over secure phone lines by using the Electronic Key Management System (EKMS) or over a digital network by using the Key Management Infrastructure (KMI). The units must then manually program the keys into the radio of each communicating device. The key-loaded



UAV video accessible only to authorized terminals   GCS operator can modify access during a mission

A LOCKMA user can transmit data from a ground control system (GCS) to intended recipients via an unmanned aerial vehicle (UAV) by providing keys to authorized users and can deny access to unauthorized users. The LOCKMA software transmits these authorizations to the intended recipients.

radios are used for just one mission; if the units need to send encrypted information during a future mission, they must download and install new keys into the radios. This key distribution process presents several risks, according to Benjamin Nahill of the Secure Resilient Systems and Technology Group. For instance, it may be difficult or impossible for units in the field to access EKMS or KMI. If an enemy captures a unit's radio, the enemy could eavesdrop on communication or impersonate the radio operator. All units involved in the communication must therefore obtain and manually program new keys into their radios. Says Nahill, "It is difficult to ensure that each unit has the correct key, so there is a need for dynamic key management."

Designed to reduce development errors and simplify the key management process, LOCKMA implements storage, organization, and management of key-related information, including key life-cycles, authorized users, and communication channels. Prior to field deployment, each LOCKMA-enabled device undergoes cryptographic provisioning during which LOCKMA generates private keys for each device that will be used in the field. The LOCKMA software uses a public key infrastructure (PKI) service to create certificates that cryptographically bind public keys to individual devices. If a unit wants to securely send a message, it can provide LOCKMA with each intended recipient's device certificate and then use LOCKMA to securely

distribute symmetric mission keys and corresponding metadata for message decryption. Because device certificates typically have a lifetime of several years, units can use the same radios for consecutive missions, bypassing multiple journeys to distribution bases. If adversaries capture a radio, LOCKMA will enable the distribution of new mission keys to all authorized radios but the captured one, preventing the enemy from receiving any new communication. The certificate of the captured radio can later be revoked through PKI.

The military is increasingly relying on the use of unmanned aerial vehicles (UAV) to distribute tactical information. For example, a unit might deploy a UAV throughout mountainous landscape to

## Lab Notes

locate enemy troops or scout tactical locations. The UAV's radio sends a signal back to the unit's ground station, displaying the live video feed from the UAV's camera. However, adversaries are gaining technology to intercept these feeds. To prevent unauthorized video access, LOCKMA can be integrated into the UAV's radio to encrypt the feed and create access restrictions. Using LOCKMA, a unit can identify intended video-feed recipients by their certificates and then send the symmetric key to a group of authorized recipients to decrypt the feed.

The Department of Defense is currently working with Draper Laboratory and the National Security Agency to integrate LOCKMA into devices that could benefit from its simplicity, focusing their research efforts on digital radios attached to small tactical devices like UAVs, according to Utin.

LOCKMA's key management messages are based on a cryptographic language standard called Cryptographic Message Syntax, which allows the software

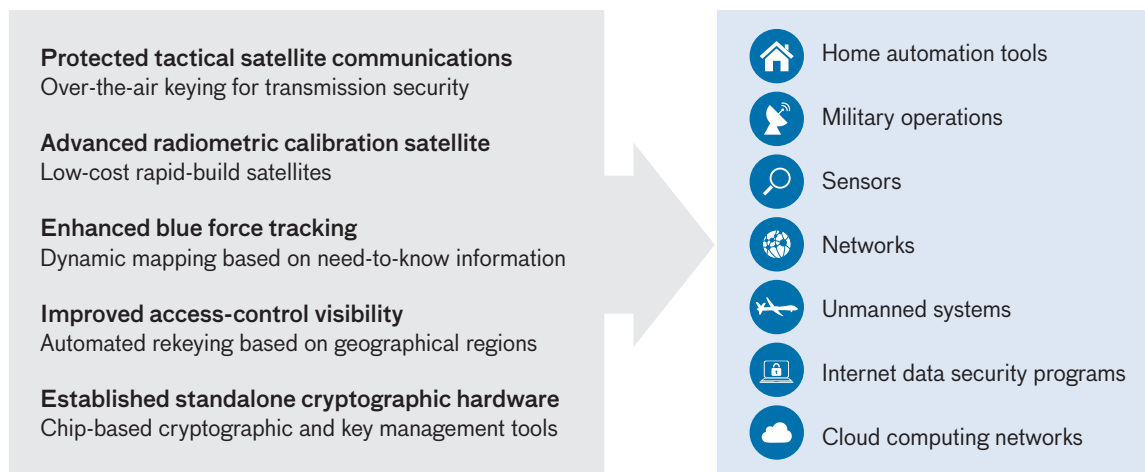
to understand and operate seamlessly with most cryptographic algorithms, modes, or key lengths. LOCKMA also works with many operating systems (e.g., Windows, Linux, Android, iOS) and independently, allowing application developers to integrate LOCKMA into devices with minimal changes to LOCKMA's application code.

"Overall, LOCKMA is much more flexible and easier to integrate than traditional key-management systems because, without its holistic approach to security, the entire cryptographic process, from key creation, to management, to delivery would be much more difficult and error prone," says Utin.

LOCKMA's application programming interface is easy for users to understand even without advanced cryptography knowledge. It hides all cryptographic complexities under the hood, allowing application developers to quickly integrate LOCKMA into devices. The technology saves time and costs compared to a custom key-management system that requires

substantial expertise, time, and capital to develop, integrate, and test. Traditional custom-built solutions are also prone to security flaws that are often exploited by adversaries, resulting in substantial additional mitigation and repair costs.

The recipient of a 2012 R&D 100 Award, LOCKMA may become more commonplace as researchers look to integrate it in commercial applications. For example, an increasing number of homes are connecting to "smart" management applications that control energy use and security systems, e.g., Google Nest. Homeowners could use LOCKMA-enabled devices to secure communications within home networks and to thwart hackers. For government organizations, LOCKMA may be useful in tactical operations. "Consider the Boston Marathon bombings," says Utin. "After the attack took place, organizations, including the police and FBI, were communicating over standard shortwave radios. The radios gave



The LOCKMA software can be applied to many applications that require cryptographic key management. Researchers are currently working to develop features (left) that enable LOCKMA to be used in commercial applications (right), such as home automation and cloud computing.

anyone, including the suspects, access to those communications. If organizations employ LOCKMA-enabled devices to protect their communications both online and in the field, they can securely provide necessary information to authorized recipients and dynamically accommodate access control changes in real time. LOCKMA really can make a huge difference in national security.”

NETWORK SECURITY

# Keeping an Eye on Cyber Threats

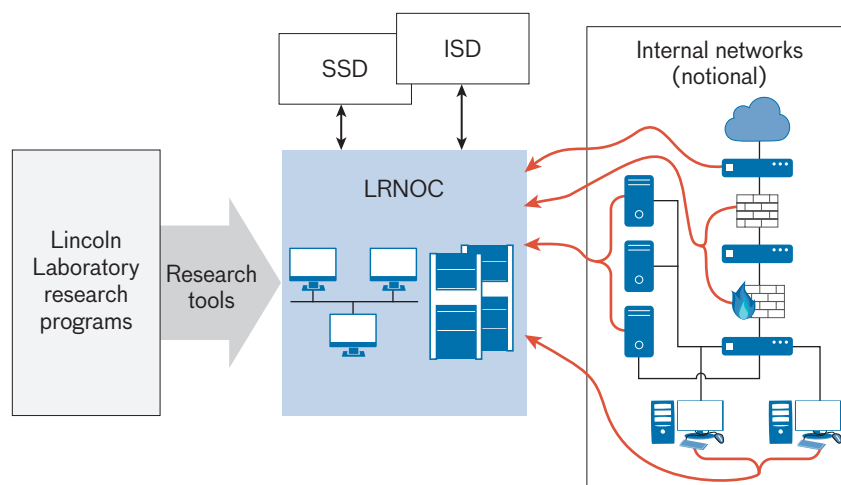
Researchers use real-time data from Lincoln Laboratory networks to monitor and develop countermeasures against cyber threats

**It is a silent threat:** as you read through your morning email and catch up on the news, hackers could be working to steal your passwords and sift through your files. Without warning, your private virtual world could become public. Many computer users know how devastating a cyber attack can be. But imagine the same thing happening to your office network. Now imagine it happening to an even larger network, such as that operated by a local government agency or financial institution. In 2014, threat became reality when researchers discovered an Internet security vulnerability named

Heartbleed in the widely deployed network security library OpenSSL. Some analysts said Heartbleed had the potential to be the most catastrophic vulnerability ever found. It allowed hackers to probe web servers by simply sending a short command packet, i.e., a heartbeat packet, to the server. The packet would ask the server to echo information back to the user with extra data attached, leading previously secure websites to “bleed” information that could include private data, such as passwords and credit card numbers. Heartbleed affected more than 500,000 networks, including Lincoln Laboratory’s, but the Laboratory’s Security Services Department (SSD) and Information Services Department (ISD) had a unique cyber defense resource that allowed them to quickly access the data they needed to identify, assess, and neutralize the threat: the Lincoln Research Network Operations Center (LRNOC).

“When Heartbleed was released publicly, ISD and SSD requested the researchers at LRNOC to help determine its impact and ensure no critical information was leaked,” says Tamara Yu of Lincoln Laboratory’s Cyber Systems and Operations Group. “Using LRNOC network data, researchers were able to quickly use research tools to assist ISD and SSD in their investigation and assess potential impact. Fortunately, no sensitive information was leaked.” Because the risk for a security breach and loss of sensitive information is high, researchers at LRNOC are working on ways to not only fight cyber threats like Heartbleed but also prevent them.

The LRNOC provides an environment in which cyber security researchers and analysts can use live network data to develop and test new techniques for defending Lincoln Laboratory’s enterprise network.



The Lincoln Research Network Operations Center (LRNOC) is the hub of cyber traffic research at Lincoln Laboratory. It gathers information from internal networks (right) and shares important information (potential threats) with the Security Services Department (SSD) and Information Services Department (ISD). Research teams also use LRNOC network traffic and data to create cyber security tools.

## Lab Notes

For many research purposes, data produced by tools that utilize statistical models can be accurately labeled and can enable repeatable experiments. However, live data are “messy” and include many unusual and unexpected formats. Any algorithm developed to detect attacks needs to be tested out in a “real world” setting in order for developers to truly understand the algorithm’s strengths and weaknesses.

“Researchers use LRNOC’s high-quality, real-time traffic data to create security tools,” says Jeffrey O’Connell of Lincoln Laboratory’s Cyber System Assessments Group. “With real data, we’re forced to push the limit of our tools, making them more capable, resilient, and adaptable. It is a very effective way to prepare for the next cyber attack.”

Armed with standard and Laboratory-developed tools, analysts sift through troves of network data looking for suspicious anomalies that may warrant further investigation. LRNOC serves as an incubator and a proving ground for next-generation tools that government sponsors look to adopt to protect their own networks from cyber attacks.

To ensure that the live data are protected, the LRNOC is on a network that is separate from the main Laboratory network. Laboratory researchers are bound by a user agreement that is consistent with ethical practices and Laboratory security policies and that regulates data removal and research activities.

Because a vast amount of data, including system and application logs, network security appliance alerts, and raw traffic, are fed into LRNOC each day, researchers have



Lincoln Laboratory researchers work in the Lincoln Research Network Operations Center (LRNOC) to identify and mitigate cyber threats.

created several tools to process the data. For example, one tool stores and aggregates each network packet entering and leaving the Laboratory network. The LRNOC infrastructure allows this tool to select packets on the basis of various features so analysts can define custom filters that capture and track communication with specific characteristics, such as heartbeat packets. Another tool, Scalable Cyber Analytic Processing Environment (SCAPE), works in a similar manner but creates feeds based on network information, such as logs and event data, rather than on raw network packets. SCAPE is a processing environment that monitors and correlates data feeds in real time to provide situational awareness about the state of the network.

These aggregator tools come in handy during attacks. Because LRNOC stores all network traffic data, researchers were able to carve out aggregated data from a period of time when the Heartbleed bug was

active, gather pertinent data from that time period, and then investigate the selected data to determine abnormalities, such as a heartbeat packet. Researchers were then able to investigate the packets and work with SSD and ISD to determine if any sensitive information had been compromised.

Within LRNOC, research teams mainly work on two areas: monitoring Laboratory operations and developing or testing next-generation tools. Researchers collaborate with SSD and ISD to understand the challenges the network is up against, according to O’Connell. For example, if an LRNOC researcher finds suspicious traffic, he or she reports it to SSD and ISD staff, who potentially implement blocks on the network. If ISD finds a system on the network that might be compromised, they pass responsibility to SSD, who makes the final call on whether or not the system should be pulled from the network. If the system is pulled, SSD investigates it

to find malicious components and may also isolate a malware sample and hand it back to the LRNOC for analysis. Working as a team, SSD, ISD, and LRNOC defend the Laboratory's network and support security protocols for sponsors.

"Our multifaceted, coordinated efforts in responding to not only daily incidents at the Laboratory but also several high-profile and potentially highly harmful exploits highlight the excellent collaboration among the various response teams, including LRNOC, ISD, and SSD," says Scott Mancini of SSD. "Their attention to detail and obvious dedication to protecting the Laboratory against daily threats are exceptional."

In addition to using LRNOC for defending the Laboratory's systems, research teams also use the network to develop security tools for government sponsors. Sponsors are eager to use programs created within LRNOC that can accurately protect their own networks from cyber attacks. For example, Laboratory researchers worked to enhance defender awareness of the specific types of scanning that adversaries conduct against the networks that they target. A Department of Defense sponsor requested and received a tool for fingerprinting five types of reconnaissance scans: Nmap, Strobe, Amap, Braa, Angry IP. This deliverable was developed and tested in the LRNOC by applying a detection algorithm on the real network data feeds in the LRNOC enclave.

Because LRNOC constantly evolves network security with each threat, its data are useful for developing current, accurate cyber defenses.

"In just 30 minutes, LRNOC can see multiple reconnaissance activities probing the Laboratory's defenses," says O'Connell. "We keep seeing new threats and new ways to come out and be ahead of the curve."

#### CYBER TECHNOLOGY

## Training the Cyber Defensive Line

A game-like competition is helping build experts in cyber "disaster response"

### The number of attacks on

computer networks is massive; for example, in 2013, the Pentagon reported getting 10 million attempted cyber intrusions a day.<sup>1</sup> These attacks are also growing in sophistication, primarily because cyber attackers are using combinations of techniques such as inserting malicious code (malware) or email phishing, and are adding complexity to the attack by involving multiple parties.<sup>2</sup> And, cyber intruders are breaching

<sup>1</sup>B. Fung, "How many cyberattacks hit the United States last year?" *National Journal*, 8 March 2013, available at <http://www.nextgov.com/cybersecurity/2013/03/how-many-cyberattacks-hit-united-states-last-year/61775/>.

<sup>2</sup>"Verizon 2015 Data Breach Investigation Report Finds Cyberthreats Are Increasing in Sophistication; Yet Many Cyberattacks Use Decades-Old Techniques," PRWire, 15 April 2015, available at <http://www.prnewswire.com/news-releases/verizon-2015-data-breach-investigations-report-finds-cyberthreats-are-increasing-in-sophistication-yet-many-cyberattacks-use-decades-old-techniques-300066005.html>.

systems in just minutes.<sup>2</sup> Network operators, who are typically tasked with day-to-day maintenance of the computer systems, are hard-pressed, and often not trained, to address this flood of advanced, novel attacks.

In response to the proliferation and growing complexity of cyber threats, the U.S. Cyber Command (USCYBERCOM) over the last three years has created squads who will act as cyber "strike teams" in the field to protect the nation's networks. To help the Department of Defense (DoD) build such cyber protection teams, staff from Lincoln Laboratory's Cyber Security and Information Sciences Division, in collaboration with several other federally funded research and development centers (FFRDC) and university-affiliated research centers (UARC), developed and conducted a series of exercises designed to evaluate the capabilities of cyber defenders. Not exactly games, these exercises, collectively called Project C, pit a red team attacking the network against a blue team defending it. The red team plans an attack strategy, and the blue team develops countermeasures to thwart the attack. "The blue team needs to learn about the network and how best to defend it, locate any attacks, defeat them, and, finally, redefend the network," says Douglas Stetson, associate leader of the Laboratory's Cyber System Assessments Group.

Project C's primary goals are to assess and improve the performance of cyber teams and to advance technologies for cyber ranges (i.e., virtual environments for training cyber analysts and developing cyber defense tools). "Physical bodies are not the solution alone," according

## Lab Notes

to Lee Rossey, former leader of the Cyber System Assessments Group, who helped establish Project C. “You need the methodology and the tools.” Rossey likens an effective cyber team to a football team: each player has a role, and they’ve all read the playbook and understand the team’s offensive and defensive strategies. To develop a cyber playbook, Project C researchers investigated a number of questions: What makes one cyber team more successful than another? Why is one set of defenses more effective than another? How can we improve a team’s capabilities? Answers to these questions will ultimately direct researchers to ways for improving subsequent rounds of training.

Project C sessions are conducted to help members of cyber protection teams be prepared to assist agencies undergoing serious cyber attack. How quickly a cyber team should be deployed to a site depends on two factors: the severity of the incident and the asset under attack. While an intrusion accomplished by a lone hacker most likely is handled expeditiously by an in-house computer security group, a coordinated assault by “well-armed” cyber adversaries requires highly trained, cyber security rapid responders. Because the DoD cannot constantly defend all data on its systems, the department has created a three-tiered Prioritized Defended Asset List for key missions and systems on a given network. Cyber teams are called in more quickly for higher-priority assets that are critical to the government’s continued functioning than for lower-ranked systems. Rossey also notes that “just because a net-



Blue team members analyze traffic and logs to determine whether an attack has occurred against their network. More than 60 personnel from active-duty, reserve, and guard units assigned to USCYBERCOM’s cyber protection forces participated in the Project C exercises conducted at Lincoln Laboratory.



During the exercises, observers watch the cyber range activity and follow how blue team players respond to cyber incidents, gauging the effectiveness, creativity, and speed of the measures deployed to counter the red team attacks.

work goes down, it doesn’t mean that you’re under attack.”

A Project C exercise is a multi-day event. At the start of each day’s session, the staff members leading the exercise give participants a full briefing on the Project C format; the red team gets an additional briefing on their attack scenarios. The “battle” typically runs from 8:00 a.m. to 1:00 p.m. Before the red team begins its attacks, the blue team patches all known operating system (e.g., Windows 7) errors so that teams do not have to consider those errors when devising their stratagems. To make the exercise for the blue team as real as possible, the red team typically generates four to six different attacks

derived from real-world threats detailed in Verizon’s Data Breach Investigations Report (available on request from <http://www.verizonenterprise.com/DBIR/>). The blue team must ensure that their defensive actions preserve the integrity, confidentiality, and availability of all data. As the blue team works to mitigate threats, the red team is figuring out the blue team’s strategies, and when one type of attack is defended, the red team tries another.

Noncombatant teams, called white teams, monitor the process, give advice if necessary, and score the results (number of successful and unsuccessful attacks, number of attacks identified by the blue team,



mitigation results). Red team attack actions, blue team actions (even those not correlated to an attack), chat logs, network traffic, and other data are collected throughout the session, and a summary out-briefing is conducted in the afternoon. The five-hour multi-attack exercise, which in reality would be a situation spanning a few days, is fast-paced and stressful. In the out-briefing, blue team interactions resulting from the pressurized exercise (e.g., inadequate communication, heated discussions) are analyzed because the team dynamics are as important to the successful resolution of attacks as are the expertise and tools the team brings to the conflict.

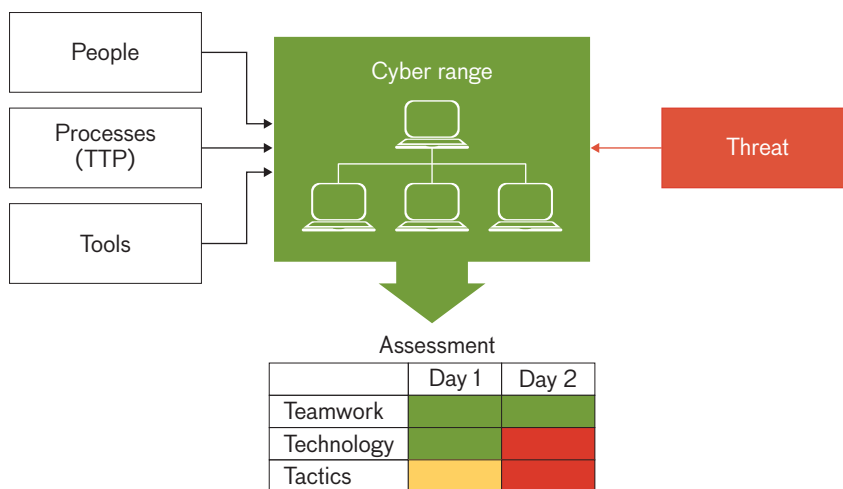
One significant advantage Project C has over other serious gaming scenarios used in DoD cyber defense training is that it can simulate any of the various government networks and communication environments, such as ShoreNet for naval ships, warfighter communications in the field, power-grid-management networks, or command-and-control systems for the nation's missile defense systems. Project C allows cyber teams to work within a notional network-connected environment nearly identical to the real one they may be asked to defend. This virtual network environment, enabled by the Laboratory's LARIAT and KOALA tools,<sup>3</sup> includes all important elements, such as servers, users, and network activity.

Another advantage of Project C is that it is scalable and adaptable

to different levels of attack severity and sophistication and to any type of network. The 2013 Defense Science Board (DSB) Task Force Report on Resilient Military Systems and the Advanced Cyber Threat classifies various types of cyber attackers. These cyber invaders range from individuals with commonplace equipment who simply employ malware developed by others to nation states that have the ability to execute cyber attacks that employ clever, new tactics. These classifications characterized in the DSB report also reflect the level of felonious intent of the perpetrators. Less malicious hackers break into networks for the challenge of doing so. Others invade systems seeking data that they can sell (e.g., the government's proprietary technical information). Critical threats to the United States are attackers targeting information that may give their nation states a military advantage. Project C's scalability and adaptability make it a valuable tool

for improving the skills of respondents to all these types of attackers. It also provides an opportunity for participants to try out innovative cyber security technologies.

Experience gained by the researchers from Lincoln Laboratory, colleagues from FFRDCs and UARCs, and the Project C participants is being applied to the future strategy for training cyber protection teams. With guidance from the Laboratory's technical staff, the DoD held evaluation exercises last summer to compare the skills of three teams who had undergone a five-week Project C-type pilot training program in April and May to the skills of teams that had not engaged in such red team/blue team exercises. Analysis of the summer 2015 assessment sessions will be used to inform the direction of USCYBERCOM's cyber defense training. You might say Lincoln Laboratory is helping draft the playbook for the DoD's cyber protection defensive line.



Project C sought to assess the people, processes, and tools of the cyber protection teams in a realistic environment with a realistic threat. The Project C format provides a day-by-day evaluation of how the team members interacted, how their technology worked, and how effective their tactics, techniques, and processes (TTP) were. In the "stoplight" evaluation chart, green indicates a highly successful performance; yellow, a satisfactory performance; and red, a breakdown or failure in performance.

<sup>3</sup>For more information about these tools, see the article "Advanced Tools for Cyber Ranges" in this issue of the *Lincoln Laboratory Journal*.

### CYBER EDUCATION

# Can a Game Teach Practical Cyber Security?

Lincoln Laboratory's Capture the Flag competition challenges college students to defend cyberspace

**Thousands of teams** around the world bearing names like the Plaid Parliament of Pwning, ghet-tohackers, or Shellphish compete each year in contests to infiltrate opponents' computer services while defending their own systems from cyber attacks. In these competitions, teams playing on networks of virtual machines earn points by breaching other teams' services to capture information that the contest administrators hide within the programming. Called a flag, the information is typically a lengthy string of random, hard-to-guess code. The first of these Capture the Flag (CTF) events was held at the 1996 DEF CON,<sup>1</sup> now one of the world's largest hacker conventions. Since then, CTF competitions have sprouted up in dozens of countries, often organized by university departments and technology companies seeking to improve students' and employees' skills in devising techniques and tools to ensure network security.

<sup>1</sup>DEF CON is an annual event that attracts not only computer hackers but also researchers from academia, industry, and government agencies.

To investigate what educational benefits CTF competitions provide to participants and whether CTF play leads to the development of innovative strategies applicable to real-world cyber defense, researchers from Lincoln Laboratory developed a CTF event for college students.

Early on, the core team of technical staff members from the Cyber Security and Information Sciences Division—Joseph Werther, Michael Zhivich, Timothy Leek, and Andrew Davis—decided that their CTF competition would be structured as an attack-defend format. Some CTFs focus on either offensive or defensive actions. In contests in which

which defensive techniques must be developed under pressure and time constraints. To further simulate the demanding pace of real attack mitigations, the Lincoln Laboratory CTF also allowed teams to score points only if their services were operational. “Requiring that team services be up in order to score points either offensively or defensively provides a very strong incentive for every team to risk running services as soon as and as much as possible,” says Davis. Finally, the attack-defend format is more challenging and, the Laboratory's organizers believe, more fun than a single-focus one.

**Capture the Flag can provide a sandbox in which prototype technologies, both defensive and offensive, can be tested and evaluated.**

only attacks earn points, competitors focus on techniques to breach security and forego protecting their systems. In defense-only matches, players employ functions to keep their services running despite assaults the CTF administrators have embedded in their virtual systems; these players do not face the pressure of devising defenses while also crafting attacks against others and foiling a steady barrage of onslaughts from other teams.

The dual format has significant advantages. Requiring success at both attack (capturing flags) and defense (securing services) to score points compels players to interact continually with opponents and their own systems. The attack-defend approach creates a dynamic, realistic environment in

Lincoln Laboratory's first CTF competition was held at MIT on 2 and 3 April 2011 and was open to Boston-area college students. Forty-five registered players from six schools showed up to spend 18 hours of their weekend attacking and defending a web application server. The virtual system was modeled as a Linux operating system running an Apache server and employing a MySQL database and Hypertext Preprocessor scripting language. So that students could experiment on an application whose code would be accessible, the open-source WordPress content management software for creating websites and blogs was chosen as the target. In addition, because the flexible architecture of WordPress allows plugins, the organizers could peri-



Lincoln Laboratory's third Capture the Flag (CTF) competition drew 165 college students to MIT for a 48-hour marathon of attacking and defending Android services. Students came from MIT, Boston University, UMass-Boston, Northeastern, Brandeis, Wellesley, Worcester Polytechnic Institute, Rensselaer Polytechnic Institute, New York University Polytechnic School of Engineering, and Dartmouth College. Many of the participants and event organizers posed for a post-event photograph. The official MIT Lincoln Laboratory CTF flag is held by a participant in the back, while in front a member of one of the top three teams is holding a replica of a check for the team's prize money.

odically add new vulnerabilities for the students to mitigate.

Many universities and businesses that run CTF competitions do so online, with registered teams downloading the necessary software and instructions so that they can tackle the challenges made available on contest days. Lincoln Laboratory's CTF organizers chose to hold an onsite event. "The competition is so much more exciting live. The energy in the room is invigorating," says Leek. "There is a lot more interaction between team members."

The Laboratory's CTF development team, which in 2011 also included Nickolai Zeldovich from MIT's Computer Science and Artificial Intelligence Laboratory, found that the algorithm used for scoring the play is vital to the dynamic nature of the competition. Designing a scheme that rewards players for achieving the defensive goals of maintaining data confidentiality, availability, and integrity and that

also awards points for offensive successes is a balancing act. After the first day of the 2011 event, the CTF organizers noticed that equally weighting offensive and defensive results encouraged teams to shut down their servers when they were planning their offense, thereby denying attackers access to the servers and increasing their own scores for maintaining data confidentiality and integrity. A revised weighting method to reward teams whose services were accessible created the motivation for them to focus efforts toward more defensive actions.

Patrick Hulin, a member of MIT's winning team in the 2012 CTF competition and now on staff in the Cyber System Assessments Group, credits his team's success to their emphasis on defense. "We narrowly focused on the essential tasks we had to complete in order to succeed under the scoring algorithm. It was more important to keep your services operating than to attack teams other

than the leaders, so we wasted very little time working on the fun but not necessarily relevant offensive moves that took a lot of work for very little actual gain in the standings."

According to the surveys that participants filled out online after the competition, the 2011 CTF event was a success. The students appreciated the challenges presented by the game; most of them thought they had improved their skills; and many reported an increased interest in a career in cyber security (though these students did note a previous interest in such a career).

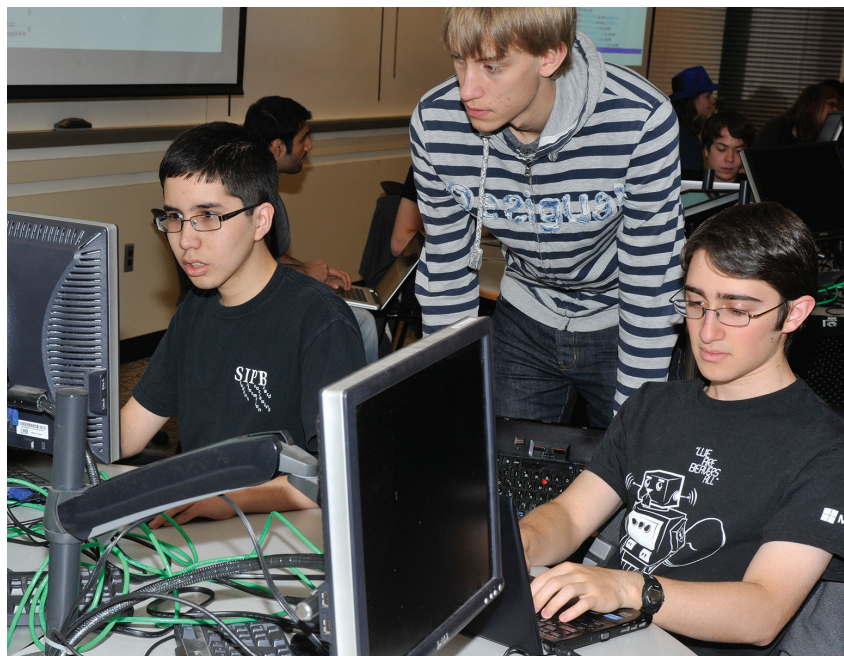
Era Vuksani, now a researcher in the Laboratory's Cyber Systems and Operations Group and formerly a member of Wellesley College's 2011 CTF team, says, "I learned a lot from being in that environment where you had to be very proactive in defending yourself from adversaries as well as be ready to wipe your machine and start over as need be. You had to be adaptable at a moment's notice."

## Lab Notes

The organizers applied lessons learned from the 2011 event to full-scale competitions held in 2012 and 2013 at MIT and a few practice sessions, or mini-CTFs, offered in 2014 at the MIT Lincoln Laboratory Beaver Works center near the MIT campus in Cambridge, Massachusetts.

In 2012, the students were tasked with sustaining the security of an enterprise web server, and the 2013 competition charged participants with supporting several apps for an Android platform and the corresponding backend services on Linux virtual machines. As word of the Laboratory's CTF spread, participation grew: in 2012, 62 students from six schools participated, and 165 players from 10 regional universities took on the 2013 Android challenge. The events also grew longer; in 2013, the students were in competition for 48 hours straight, eating while working and taking turns catching naps.

The 2013 CTF event also introduced a new element—evaluating an outside organization's technology. Employees from Raytheon BBN Technologies tested out their Advanced Adaptive Application (A3) Environment prototype by trying to defend the CTF's App Store against attacks from the competition teams. After a flaw identified in the A3 Environment software on the first day was remedied, the prototype was able to secure the App Store. The Laboratory's CTF organizers concluded that "CTFs can provide a sandbox in which prototype technologies, both defensive and offensive, can be tested and evaluated," but with the caveat that the



Success at Capture the Flag depends a great deal on the teamwork exhibited by the competitors as they plan attack and defend strategies.

technology developers need to be on hand to fix problems. From their participation in CTF, the A3 Environment researchers gained improvements to their code, validation of their defensive policies, and a corpus of attacks they could use in building later iterations of their technology.

Creating a challenging, well-functioning CTF competition requires a significant investment in software development. The Laboratory researchers devoted a great deal of effort to conceiving the scenarios, cyber vulnerabilities, and scoring strategies, and then to building the software and interfaces that enabled these.

Lincoln Laboratory's CTF experience has been successful on a number of fronts. First, the researchers who worked on developing the competitions acquired some answers to their initial questions.

- Do CTF events help educate students in cyber security? The answer is a qualified yes. Students who are already inclined to engage in cyber defense, who have perhaps tried online CTF games, will strengthen their competencies in computer security. Zhivich likens the learning to that of athletes sharpening skills through practice: "Good ballplayers get better as they play more." It is harder to say how much CTF participation teaches students who do not have prior experience in computer security; in the Laboratory's CTF games, the less experienced students did not amass high scores but felt they took away new awareness of the cyber field. Although precompetition tutorials on cyber defense tactics, common cyber tools, and web applications

were appreciated by the students who attended these sessions, the researchers cannot say definitively that the tutorials resulted in helping to “level the playing field” for the inexperienced CTF teams. “We believe CTF works as a kind of group self-guided, project-based instruction,” says the CTF research team’s 2014 paper chronicling their findings from hosting the events.<sup>2</sup>

- Can CTF events generate new ideas for real-world cyber defense tactics and tools? Again, the answer is not definitive. The researchers monitoring the competitions directed their attention to keeping the game running smoothly. They state in their 2014 paper that they would like to understand better what teams do to win CTF competitions and that “it may be possible to discover new advanced techniques for attack and defense by providing college students a safe [i.e., not incurring legal repercussions for hacking real networks] place to play.” However, the experience with the A3 Environment shows that a CTF event can be used as a test bed for new technology.

On another front, the collegiate CTF competition introduced the Laboratory and many talented young people to each other. Indeed, five staff members hired into the Cyber Security and Information Sciences Division had participated in

<sup>2</sup> A. Davis, T. Leek, M. Zhivich, K. Gwinnup, and W. Leonard, “The Fun and Future of CTF,” 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, available at <https://www.usenix.org/node/184963>.

one of the Laboratory’s CTF events. As the world becomes more dependent on computer networks to conduct all its activities, nations and private businesses are eager to find the best-qualified people to secure their network services. Hosting a CTF event could be one avenue for organizations to meet those people.

Finally, the CTF events resulted in personal successes for participants and organizers. In the post-game surveys, students cited not only increased understanding of cyber security but also improved teaming skills as takeaways from the competitions. Resolving the technical demands of crafting the scenarios and developing the automated scoring application were interesting projects for the Laboratory staff members. Furthermore, both students and the CTF staff had fun.

The researchers who conducted the CTF events under funding from the National Security Agency have completed their investigation into CTF’s role in enhancing education and development in cyber security techniques. Their published experiences can serve as a road map for future Lincoln Laboratory CTF events and for other organizations considering the establishment of CTF competitions.<sup>2,3</sup> In addition, the research team is looking to make their infrastructure available as an open-source codebase.

<sup>3</sup> J. Werther, M. Zhivich, T. Leek, and N. Zeldovich, “Experiences in Cyber Security Education: The MIT Lincoln Laboratory Capture-the-Flag Exercise,” *Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, 2011, available at [http://www.ll.mit.edu/mission/cybersec/publications/publication-files/full\\_papers/2011\\_08\\_08\\_Werther\\_CSET\\_FP.pdf](http://www.ll.mit.edu/mission/cybersec/publications/publication-files/full_papers/2011_08_08_Werther_CSET_FP.pdf).

## EDUCATIONAL OUTREACH

# Recruiting the Next Generation of Cyber Security Specialists

Two Lincoln Laboratory outreach activities seek to steer high-school students toward careers in cyber security

**Today’s cyber security specialists** are too few in number and lack the skills needed to defend networks supporting the nation’s government agencies, financial institutions, power grids, and transportation systems. As cyber attacks escalate in frequency and sophistication, this shortage of adequately trained personnel will become even more acute, particularly within the U.S. government.

Lincoln Laboratory is trying to address one of the roots of the shortage in cyber security professionals: the lack of cyber security education in school curricula. Two programs designed for high-school students—CyberPatriot and LLCipher—have been a part of the Laboratory’s efforts to help fill this gap. By engaging these precollege students in activities that highlight the appeal of cyber security work, the Laboratory hopes they will be motivated to pursue undergraduate studies and eventually careers in the field.

Since 2011, Lincoln Laboratory has sponsored teams of high-

## Lab Notes

school students participating in the CyberPatriot National Youth Cyber Defense Competition, a program initiated in 2009 by the U.S. Air Force Association to spark young students' interest in cyber security or other science, technology, engineering, and mathematics fields. A network defense competition, CyberPatriot challenges students to find vulnerabilities (e.g., malware, weak passwords, unnecessary services) within a set of virtual images that represent Windows or Linux operating systems while maintaining critical network services, such as email. Each image contains anywhere from 10 to 20 flaws; the teams that discover the most flaws within a six-hour time limit advance to subsequent rounds. Although the format of the rounds and the scoring system have evolved over the years to support the growing number of registered teams (eight to start and more than 2000 in the 2014–2015 season),



Helping a student prepare for the CyberPatriot competition, Robert Cunningham, leader of the Secure Resilient Systems and Technology Group, explains how to configure a Windows 7 system to ensure strong passwords.

the basic advancement process has remained the same, with teams competing at the state, regional, and national levels.

In its first two years of participation in the CyberPatriot program, the Laboratory sponsored a single team; for the past two years, three teams have been sponsored. Teams typically consist of five to six students, many of whom compete in multiple CyberPatriot seasons. Veteran members are often paired with rookies, according to

Chiamaka Agbasi-Porter of the Communications and Community Outreach Office, who coaches the teams and recruits Laboratory volunteers to serve as mentors. From September through March, the students and mentors meet once a week for two hours at the MIT Lincoln Laboratory Beaver Works facility near the MIT campus in Cambridge, Massachusetts. During these weekday sessions, students learn and practice the computer and teamwork skills they need to



For two years in a row, the first Laboratory-mentored CyberPatriot team, DoNot Hack Us, was one of 12 finalists selected to compete in the national championship held in Washington, D.C. More than 1000 teams entered the competition in each of those years. Seen above left are three of the five team members racing against the clock to detect vulnerabilities in the areas of policy, patch, configuration, and third-party management during the 2013 finals. After graduating high school, three CyberPatriot alumni from the team spent their summer interning in the Cyber Systems and Technology Group (above right). All three have chosen to pursue computer science in their undergraduate studies.

compete in CyberPatriot. Throughout the season, technical staff from the Laboratory give presentations on relevant topics, including cryptography, networking, Windows internals, and Linux security. On some weekends early in the season, all CyberPatriot teams participate in online qualifying rounds from their home base, finding vulnerabilities within virtual machine images downloaded onto laptops. These rounds could also include a Cisco Networking quiz or a Cisco Packet Tracer (a network simulation program for students to experiment with network behavior) challenge—one of the mechanisms through which teams can gain points beyond those acquired by fixing vulnerabilities. Points are also awarded for answering forensics questions about the steps taken to remediate the vulnerabilities. Teams lose points if they take any actions that make a system less secure (e.g., reintroducing a previously fixed vulnerability). Scores are automatically recorded by a centralized scoring system.

Jorge Coll, a technical staff member in the Secure Resilient Systems and Technology Group, is one of the CyberPatriot mentors. A previous Microsoft employee, Coll focuses on the Windows operating system, helping students identify misconfigured settings; configure their machines with policies, such as those for password restrictions; and ensure software patches are up to date. One of Coll's major contributions has been in the area of competition strategy: How can students maximize their time to gain as many points as possible?



CyberPatriot team members collaborate on finding malware and locking down a Windows virtual machine during one of the online weekend competitions.

“The two largest time sinks students struggle with during the competition are discovering what is wrong with any given system and applying security best practices to lock down their machines,” explains Coll. To reduce the time spent on such tasks, Coll introduced the students to various automation tools, including Windows PowerShell (a command-line interface and scripting language), security policy templates, and techniques for recognizing configuration drift (i.e., changes to a system's hardware or software environments). “For example, with PowerShell, students can automatically query login records to see when the last time a particular user accessed his or her account, instead of having to manually sift through these records,” says Coll.

The track record of the Laboratory teams has been impressive.

For the 2011–2012 and 2012–2013 seasons, the one Laboratory-sponsored team advanced to the national competition in Washington, D.C., where they placed 7th among 11 finalist teams both times. At the end of the 2013 season, most of the team members graduated from high school. New team members were recruited for the following season (2013–2014), resulting in three teams, all of whom came very close to qualifying for the national finals. In 2014–2015, all three teams competed at the highest level in the statewide competition, and one went on to complete its season at the Northeast regional competition.

While CyberPatriot is at its core a competition, with scholarship money given to the top three teams, it is more than a game. “CyberPatriot gives students an

## Lab Notes

early window into cyber security, a field that most students do not encounter until college,” says Sophia Yakoubov, one of the mentors and a technical staff member in the Secure Resilient Systems and Technology Group. Yakoubov taught the team members about classical cryptography and cryptanalysis. “I showed them how, just by looking at an encrypted message or file, they can figure out which encryption scheme was used and then how to apply various techniques to crack it,” she explains.

With the help of colleagues Emily Shen and David Wilson, Yakoubov served as the lead instructor for a new cyber security-focused outreach program, LLCipher, in summer 2015. Held at Beaver Works, this one-week cryptography workshop provides an introduction to modern cryptography—a math-based, theoretical approach to securing data. Lessons in abstract algebra, number theory, and complexity theory provide students with the foundational knowledge needed to understand theoretical cryptography. Students then construct provably secure encryption and digital signature schemes. On the last day, the students learn about two techniques that enable multiple entities to exchange data without disclosing to one another more data than necessary to perform a particular function: zero-knowledge proofs (proving a statement is true without revealing any information beyond the truth of the statement) and multiparty computation (computing a function over multiple parties’ inputs while keeping the inputs private).



Workshop designer and lead instructor Sophia Yakoubov (standing) makes her way through the classroom as the students work on a physical secret communication challenge. Teams of three, an all-girls one of which is pictured above, assumed the roles of Alice, Bob, and Eve—common archetypes in the cryptography literature. The premise of the challenge is as follows: Alice is trying to securely communicate a secret to Bob; Eve is trying to eavesdrop. Alice and Bob are both given individual locks to affix to a writing notebook, which contains the secret, and corresponding keys. To solve the challenge, teams must figure out how the lock-key systems can be applied to the notebook so that Bob can read the secret but Eve cannot.

The idea for LLCipher came from Bradley Orchard, a technical staff member in the Advanced Sensor Systems and Test Beds Group and a part-time teacher at the Russian School of Mathematics in Lexington, Massachusetts. While teaching at this enrichment school for the past four years, Orchard encountered several remarkably bright students who were just entering high school yet were ready to take calculus—a course typically reserved for the senior-year curriculum. “These students are often two to three years ahead of their classmates in regular school,” explains Orchard. Recognizing these students’ need for learning opportunities beyond those offered

in schools, Orchard set to work to design an introductory summer course for advanced students. With his academic training as a mathematician, he naturally thought theoretical cryptography would be the ideal subject matter for the course: “Theoretical cryptography combines beautiful mathematics with powerful, useful, and fun techniques and, most importantly, aspects of cryptography are very accessible to advanced students.” Orchard proposed his idea to John Wilkinson, leader of the Cyber System Assessments Group, who reached out to cryptography experts within the Laboratory’s Cyber Security and Information Sciences Division to help design



and teach the course. Knowing how much she enjoyed teaching the CyberPatriot students about cryptography, Yakoubov was eager to get involved.

According to Yakoubov, the pilot program was a huge success: “The class was very interactive, with students asking questions that demonstrated they understood the material. The feedback we received from the students indicates they really enjoyed LLCipher and learned a lot.” When asked about the most interesting thing he learned, one student replied, “Zero-knowledge proofs, as they seemed impossible. The idea of proving knowledge without sharing it is fascinating.”

As Orchard had hoped, the subject matter of the course piqued student interest. “My favorite thing about this program was learning about cryptography, as it was dif-

ferent from traditional math and required a different type of thinking,” another student commented. Among students, the most common suggestion was to extend the length of the program. On the basis of this feedback, the instructors will increase the sessions from two to eight hours per day next year.

CyberPatriot and LLCipher are two of the Laboratory’s outreach programs dedicated to cyber security education. At the college level, a Capture the Flag competition based on an attack-defend approach seeks to equip students with the skills needed for real-world network security (see Lab Note titled “Can a Game Teach Practical Cyber Security?” for more information). The Laboratory’s Science on Saturday demonstrations have made topics, such as computer authentication, accessible to the younger K-12 crowd.

By reaching out to students at different levels of their education, the Laboratory hopes to, at some point, incite their interest in cyber security—a field that will only expand in the coming years. “Every day, attackers break into computers holding sensitive information. The need to secure these data is great, but there is a shortage of people with the right knowledge and experience to meet this need. Currently, the Department of Defense is seeking to hire 6000 cyber security personnel but so far has only hired half of that,” explains Robert Cunningham, one of the CyberPatriot mentors and leader of the Secure Resilient Systems and Technology Group. “Programs like CyberPatriot and LLCipher help grow the base of those who are knowledgeable about computer security while also teaching students about leadership and critical thinking.”



Students in the LLCipher program gathered for class in the morning at Beaver Works. Here, Yakoubov provides a lesson on the EIGamal algorithm for public key encryption.

# Advanced Tools for Cyber Ranges

Timothy M. Braje

In response to the growing number and variety of cyber threats, the government, military, and industry are widely employing network emulation environments for cyber capability testing and cyber warfare training. These “cyber ranges” have been increasing in size and complexity to model the high-volume network traffic and sophisticated attacks seen on the Internet today. For cyber ranges to operate effectively and efficiently, organizations need tools to automate range operations, increase the fidelity of emulated network traffic, and visualize range activity. Lincoln Laboratory has developed a variety of such tools.



**With the recent high-profile cyber** attacks on government agencies, such as the U.S. Office of Personnel Management [1], and on companies, including Target, Home Depot, and Sony [2], the dangers of cyber attacks have gained national prominence. Cyber attacks threaten not only the security of personal data but also the national critical infrastructure, for example, power grids and transportation systems [3]. To mitigate the cyber threat, researchers are actively developing cyber defense tools. Before these tools can be deployed in corporate or military networks, they must be tested and validated in realistic environments. Simple tests conducted on developers’ computers are insufficient because these tests do not have the required level of realism. Needed are high-fidelity, surrogate networks (i.e., cyber ranges) in which we can introduce attackers, defenders, and defensive and offensive capabilities, and measure the performance of these capabilities in the hands of skilled network defenders pitted against realistic adversaries. To help create and operate these cyber ranges, tools are needed to (1) automate the configuration and generation of complex network environments; (2) create high-fidelity emulated user traffic on these networks; and (3) effectively operate and visualize the rich traffic environment being executed on the range during an event, i.e., a scenario to test capabilities or train personnel.

## Cyber Ranges

At the crudest level, cyber ranges are racks of computer hardware. What makes them interesting, however, is their ability to be reconfigured into essentially endless complex

network topologies and overlaid with different network traffic profiles. Because cyber ranges are typically disconnected from external networks (and thus have no access to the Internet or to any network resources) to prevent disruptions or damage to live networks during testing and training exercises, all network conditions and activity must be generated from scratch.

As a result, cyber ranges are a scarce, expensive resource. Teams of information technology (IT) staff are required to maintain the hardware and support the events that are executed on the range (e.g., assessing the effectiveness of a software or hardware system). Properly configuring the range for an event can be a daunting task: network machines need to be built, network routing and defensive tools need to be installed, services to support an event need to be deployed, event-specific traffic generation and applications need to be set up, and, finally, this entire infrastructure needs to be configured. The range community has been scaling the size and capabilities of cyber ranges to more realistically depict the network environment (e.g., by increasing the number of network machines, generating more traffic, configuring additional applications), only complicating the aforementioned tasks. What we gain from this expense and complexity is the ability to perform assessments, experimentation, and training that would not be possible without cyber ranges. It is within this context that Lincoln Laboratory has developed a tool suite to help ease the workload burden on IT staff and to drive costs to a manageable level.

### Range Tools

As cyber ranges become larger and more complex and their use becomes more prevalent, the importance of automation and sophisticated tools increases; we need to be able to quickly and accurately build and configure networks and to describe the ranges and events we would like to execute. Once the networks are configured and operational, we need to overlay virtual users that automatically perform the activities of real users to generate simulated network traffic. Finally, we need analysis infrastructure so that we can monitor events as they execute and can examine in great detail the results of those events. Our tools extend automation capabilities, increase environment fidelity, and scale to cyber ranges of both high complexity and very large size. In this article, we discuss the tools we

have developed, beginning with our efforts to develop a standard event-description language—an enabling technology for our entire tool suite.

### Standardization

In the cyber range business, the data used to describe how the range should be built and configured are typically separate from the data used to describe how the traffic generator should operate. These inconsistent descriptions result in traffic generators having an inaccurate understanding of the range's layout. Consequently, significant time and effort are wasted on reconciling discrepancies. One straightforward way to avoid this inefficiency is to create a single description language that can be used by all of the tools that participate in a cyber range event. This description language needs to be precise, machine readable, portable, and comprehensive. Lincoln Laboratory has been developing ontologies (called the Common Cyber Event Representation) to describe the network (e.g., hosts, subnets, routing infrastructure, firewall rules, virtual local area networks). We feed data derived from these ontologies into all of our tools, from our Automatic Live Instantiation of a Virtual Environment (ALIVE) application for range build-out to our Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) [4] application for traffic generation and range control.

Using a common cyber event data source offers more benefits than just a consistent view of the configuration data; it also allows us to perform integrity analysis on our cyber event data before we use any range time. Because cyber ranges are costly to operate and maintain and are relatively scarce, cyber range time is expensive, so catching data integrity issues before the event begins is very important. We have developed many rules and validation checks that we perform on the event description while it is being developed, giving us a high degree of confidence that, when we deploy the described range, it will operate as expected.

Of course, a standard description language that is only used by the tools of the organization that developed it is not as useful as it could be. If shared by multiple organizations, the description language can enable tool interoperability and reuse. We are actively working with industry partners to develop a standard language that could be adopted by organizations in the cyber range business.

### Range Automation

A cyber range useful for a variety of purposes potentially needs to be configured differently for every event. While the hardware often remains the same for each scenario executed on the range, the network topologies, services, and traffic patterns layered on top of that hardware change. Typically, we use virtualization technologies like VMware to build out custom networks for every event. This network churn is a burden on cyber range administrators, who maintain the range hardware and set up custom environments for different events as they are scheduled. Automation tools are essential to relieve this burden. While each vendor (e.g., VMware, HP) has custom software solutions to help build virtual networks, these solutions are usually designed around a single use case with needs that significantly differ from those of a cyber range. As such, these solutions are optimized to repeatedly “stamp out” identical copies of the same network or virtual machine. Tools for rapid network design and reconfiguration are currently lacking.

Lincoln Laboratory has developed ALIVE to fill this gap. ALIVE ingests configuration files from the Common Cyber Event Representation and then automatically and reliably builds out the necessary virtual machines and networking infrastructure to make the network function. ALIVE can create virtualized networks within VMware Elastic Sky X (ESXi),<sup>1</sup> automating most of this network build-out, including the creation of end hosts (clients), routers, firewalls, and many of the servers needed to support interesting traffic generation (e.g., Microsoft Exchange Server, Active Directory). After the operating systems are installed and networking is configured, ALIVE can install on each host other software packages, from web browsers to office applications to email clients and other user software. ALIVE also creates the user accounts that are required for the traffic generators to operate. A typical enterprise network would have its own procedures for generating credentials for new users on the system, but for range events, the virtual users that will be operating on the environment are already known. User accounts are an essential component of the range enterprise environment, and ALIVE can create them in bulk (including Active Directory credentials and Microsoft Exchange mailboxes) as part of the range build-out and configuration.

<sup>1</sup>In the future, additional virtualization backends may be supported.

### Emulation Environment

Cyber ranges are disconnected from the Internet; however, most of what we do with computers requires Internet connectivity. Users connect to Facebook, Google Mail (Gmail), and corporate intranet sites, and send email to each other through webmail services or other email hosts (like Exchange). Without access to these services, we cannot make the range come to life with virtual users interacting with dynamic content, applications, and each other as real Internet users would.

To emulate the Internet, we leverage several techniques. We sample 10s of 1000s of sites very shallowly to scrape their content and efficiently and realistically rehost this scraped content by using our custom-written software. Through a similar process, we closely mirror sites so that the emulated users can browse deeply into the sites' content. This content is rehosted with Microsoft's Internet Information Services (IIS) or the Apache HTTP Server. Because the rehosted content is inherently very static, we periodically collect new content. Emulating rich web applications, which constitute the majority of the Internet traffic we see today, is not as straightforward as emulating content. Although we would like to emulate users' interactions with webmail servers like Gmail or Yahoo! Mail, Google and Yahoo are not going to give us their proprietary software and, without an Internet connection, we cannot access these servers directly. Instead, we must choose “surrogate” servers and then carefully model interactions with those surrogates. An open-source alternative, Zimbra Collaboration, allows us to build models for users that interact with a webmail server that we can call Gmail or Yahoo! Mail. While the modeled network traffic will not exactly match real network traffic, the interaction model will be very similar, and for most scenarios, the interactions are the important part of the traffic model. Lastly, we emulate the root Domain Name System structure of the Internet to provide the link between website names and their numeric addresses.

The Internet is not the only service users expect to have. Users access corporate email servers, directory services, websites, and file shares. Within the description of the environment we are building, we include all of these services. ALIVE is able to automatically build and configure many of them. The number and types of services that we deploy are constantly being expanded so that we can create environments of ever-increasing fidelity.

Given a high-fidelity emulation environment, we need to overlay virtual users onto the network so that the network appears as if it is being used by real people. On an actual network, users interact with applications, services, and each other, ultimately producing a rich network traffic environment. It is within this traffic environment that we need to test our tools and capabilities.

### Background Traffic

Background traffic is the term we use to describe the normal, random-looking traffic that you would see if you were to inspect the network. It is the by-product of everyday network activities: sending and receiving emails, interacting with content on the Internet, and chatting with friends and coworkers. This traffic affects the way tools work. For example, a network intrusion-detection tool has a much more difficult time detecting malicious traffic within background traffic environments (normal traffic is commonly misidentified as malicious) than it does within “clean” environments in which only malicious traffic is present. To create high-fidelity testing environments for cyber range tools, we need to emulate the constant network activity that normal users produce. This background traffic also covers malicious traffic that is introduced onto a network, as oftentimes attackers hide their activity within the background.

There are several techniques for generating network traffic. Commercial solutions, such as Ixia’s BreakingPoint, create realistic, packet-level traffic (i.e., streams of bits on the network) [5]. These techniques involve either replaying network packets or generating streams of bits on the network that emulate specific protocols. They are highly scalable, are relatively simple to add new traffic types to, and have sufficient fidelity for many scenarios, including those in which you want to push as many bits as possible across a link or through a piece of software. BreakingPoint is designed to efficiently generate this high-bit-rate traffic with a variety of network protocols, and we have found it useful for augmenting our background Internet traffic to increase traffic volume and protocol variety.

Instead of building a protocol emulator, Lincoln Laboratory is building a different kind of traffic generator—one that generates traffic that is tailored to real, specific user-application interactions. We hook into (i.e., programmatically control) existing installed applications on behalf of each virtual user in the emulated network,

making them automatically perform their actions and, as a by-product, produce network traffic similar to that produced by a real user. This approach has several advantages over protocol emulation:

1. Each and every user interaction generates traffic in the same way a real user would, including second- and third-order effects (e.g., a Domain Name System look-up caused by a website visit).
2. Because our virtual users are interacting with real applications, they can click on malicious links, download compromised files, and carry out other actions that real users will inevitably perform on a network.
3. Unlike packet generators, traffic generators can provide real targets for malicious code propagation and endpoints for attackers to leverage for further attacks within the network.

This level of fidelity comes at the costs of increased complexity and smaller network sizes. For every traffic generator, the need for a fully configured operating system reduces the amount of traffic that can be produced for a given set of hardware. The events that we have designed LARIAT to support (e.g., red team [offense]/blue team [defense] exercises, evaluations of complex network tools) require this level of fidelity to allow for realistic attack propagation [6].

### BLUE TRAFFIC

A significant part of LARIAT is its actuation capability, which allows the system to realistically interact with applications that real users would have installed on their computers. For blue users, LARIAT contains actuators (i.e., application emulations) for standard user software, such as office applications, mail clients, and web browsers. Using these kinds of software, virtual users can generate and edit documents, send emails to each other, and interact with web content and web applications. By finding programmatic hooks into user applications, LARIAT builds a model of the software and automatically executes the actions that a user would perform when interacting with the software. These same programmatic hooks that are used to control the applications’ behavior also allow LARIAT to receive feedback from the software with which it interacts.

Many applications, however, are not controllable in this way. For those cases, we use image-processing techniques on the video output from the virtual user’s machine

to recognize available actions that can be performed on an application. Then, keyboard or mouse commands are sent to that application to make it perform its actions. For example, in order to browse to a website, we would use image-processing techniques to find the location of the URL bar, send mouse move commands to position the cursor at the correct place on the screen, send a mouse click command to bring the URL bar into focus, and then send keyboard click commands to type the URL. We have developed an actuator that works remotely by interacting with keyboard, video, and mouse (KVM) devices or through a virtual network computing connection. Using either of these connection types, this actuator (KVM-based O Artifact LARIAT Actuator, or KOALA) interacts with applications in much the same way a real user would by recognizing relevant images from a video stream and then performing keyboard or mouse actions at those image locations. In many ways, this means of interacting with the application provides an even more realistic application interaction model than the one produced by typical LARIAT actuators.

Realizing we will be unable to build all actuators of interest to the cyber range community, we are also building a platform into which actuators can be plugged. Our actuation system in no way requires upfront knowledge of all the actuators that may be used within an event. We provide hooks for programmers to dynamically register their custom actuators to seamlessly work within our environment. In fact, we build our own actuators in this way so that we can refine our processes and application program interfaces. In particular, KOALA provides a visual scripting language with which range developers who are interested in building interactions with applications can capture the necessary images and register the appropriate actions against those images; these actions can then be assembled into larger scripts that describe the application interaction model.

#### RED TRAFFIC

Many uses of cyber ranges involve testing offensive and defensive tools, or running red-on-blue exercises (Figure 1). Adversarial traffic is absolutely essential for creating a realistic environment for these events. This traffic is used not only as a cover for live red teams to help assess the stealth of their teams or their tools but also as a base level of attacks that the defensive tools must



**FIGURE 1.** During a red/blue exercise held at Lincoln Laboratory, members of the blue team look through data gathered by their defensive tools to tease out signatures of network attackers—both LARIAT virtual users and members of the live red team. The network defenders are from different Cyber Protection Teams, which are being created by the U.S. Cyber Command to help companies and government agencies defend their networks from cyber attacks.

protect against. Malicious traffic has a different character from that of blue traffic. In many ways, it can look like normal system administrator traffic, with attackers scanning computer ports, creating accounts, changing passwords, and installing software. Attackers also engage in more obviously malicious actions, such as creating botnets, performing network reconnaissance, and pivoting from host to host. Lincoln Laboratory has been developing an automated capability, the Lincoln Laboratory Attack Framework, to generate these kinds of malicious activities, including many of the exploits provided in Metasploit, a network-penetration testing software suite [7]. Generating coordinated attacks against blue networks, this framework provides a relatively large-scale, fairly sophisticated array of attacks that would be encountered in real environments.

#### User Modeling

To emulate real network users, we need models for many kinds of users with different behaviors; at the same time, we need a modeling engine that is both simple and powerful so that general user behaviors can be described and easily encoded within the system. Fulfilling both of these requirements is particularly challenging because the behavior descriptions must be distributed across poten-

tially 10s to 100s of 1000s of virtual users on a large network; thus, the description language must characterize many user behaviors in a succinct but precise manner. Additionally, the execution of these models needs to be mostly self-contained and autonomous. We will be unable to scale a modeling architecture that requires a single master server to dole out actions to each virtual user; once a size threshold is met, the single master server cannot keep up with the workload. We must find other ways to build models in which the users coordinate actions to achieve a common goal.

#### USER MODELING BASICS

LARIAT comes with a modeling engine that is provided separately from the actuators. The modeling engine is a language that allows us to aggregate our actuator actions into simple models, aggregate those simple models into larger models, and then build virtual users that are configured to use different aggregations of these interaction models. We decouple the modeling capability from our actuators, keeping us from mixing modeling and actuation logic and providing us with the ability to more easily integrate actuators written by others and to build single models that mix actions from different actuators. For example, we can combine actuator actions and build simple models of what it means to compose a Microsoft Word document or to randomly surf the Internet. We can take those models and aggregate them into more interesting models for surfing the Internet for some interesting facts on a particular topic and then feed those facts into the document we are creating. We could then vary how we combine these actions to make different models of what we could call an analyst, intelligence officer, or other type of user.

In addition to having these aggregation and composition capabilities, the modeling language can automatically interact with the environment, detecting and responding to failures. Consider the case of a corporate Microsoft Exchange Server going down: a user who had intended to use the server to send an email could use a webmail service instead. The modeling language also automatically handles the selection of specific applications needed to accomplish tasks (e.g., choosing Chrome, Firefox, or Internet Explorer when given a model of a web browser). Perhaps most importantly, the engine provides several developer conveniences, such as automatic handling of error propagation.

#### MISSION MODELING

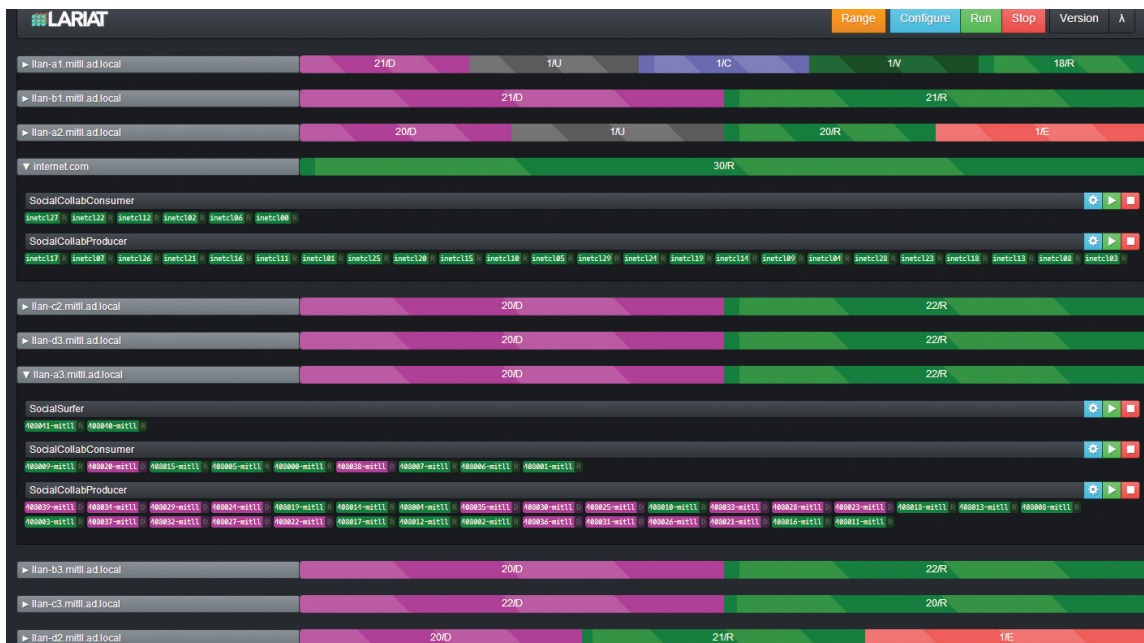
Once we have established a modeling capability that supports random (but semi-intelligent) background traffic, the next level of interesting user behavior is mission modeling. Missions are coordinated actions among several virtual users that, in aggregate, achieve one large goal—for example, several agents at an air operations center are working to produce a portion of the daily air tasking order,<sup>2</sup> which needs to be sent to a commander for assembly into the final order [8]. We have just begun to model these kinds of missions and are researching ways to express coordinated actions within the modeling engine. We can already model simple coordinated tasks like the one above, but we are interested in expanding the fidelity and increasing the complexity of the models we can build.

For missions that need to be very precisely controlled, we have prototyped a scripting capability that allows the author of a model to specify actions that should occur at a given time or within a certain time interval of another action. This scripting capability is currently fairly limited, but already we have used it to describe models of malicious actors working within an organization to sell the secrets of that organization.

#### Event Operations

Given tools to precisely specify an event, automatically build out the cyber range based on the specification, and generate realistic network traffic, we still need to execute the event. LARIAT provides a graphical user interface (Figure 2) that helps with this task. This interface guides the range operator through the workflow of configuring the virtual users with the data needed to execute their behaviors, validating that the configuration is correct, and then starting and stopping traffic. While necessary, these functions are clearly not sufficient for comprehensive situational awareness of an event. Range operators running the event need to be able to build and maintain an accurate understanding of the current states of potentially many 1000s of machines, users, and traffic flows. An easy-to-understand visualization of the virtual user (or even of the host that the virtual user executes its actions on) states can help range operators understand their events to the level necessary. Additionally, event opera-

<sup>2</sup> An air tasking order is a document created by an air operations center that has command and control of a particular theater. The document outlines how airpower will be used over a 24-hour period.



**FIGURE 2.** Each gray bar (most of which are collapsed) on LARIAT’s graphical user interface represents a subnet (e.g., llan-c2.mitll.ad.local). Roll-up summaries show the statuses of the virtual users within that subnet; on the top row, the fuchsia bar (21/D) indicates that 21 users are currently unresponsive, the gray bar (1/U) specifies the one user that has never been heard from, the purple bar (1/C) represents one user in the configured state, the dark green bar (1/V) shows one host that is ready to start running, and the light-green bar (15/R) represents 15 users that are running as expected. Expanding out a subnet view shows details at the user type or individual host level. For example, two user types are shown in the expanded view of the internet.com subnet: SocialCollabConsumer and SocialCollabProducer, with the individual users listed below them. The play, stop, and send configuration buttons allow the operator to control the operation of virtual users by sending them configuration data or commands to start or stop traffic.

tors want to perform analyses of the event either during its execution or afterwards in order to measure the effectiveness of the event.

**COMMAND AND CONTROL**

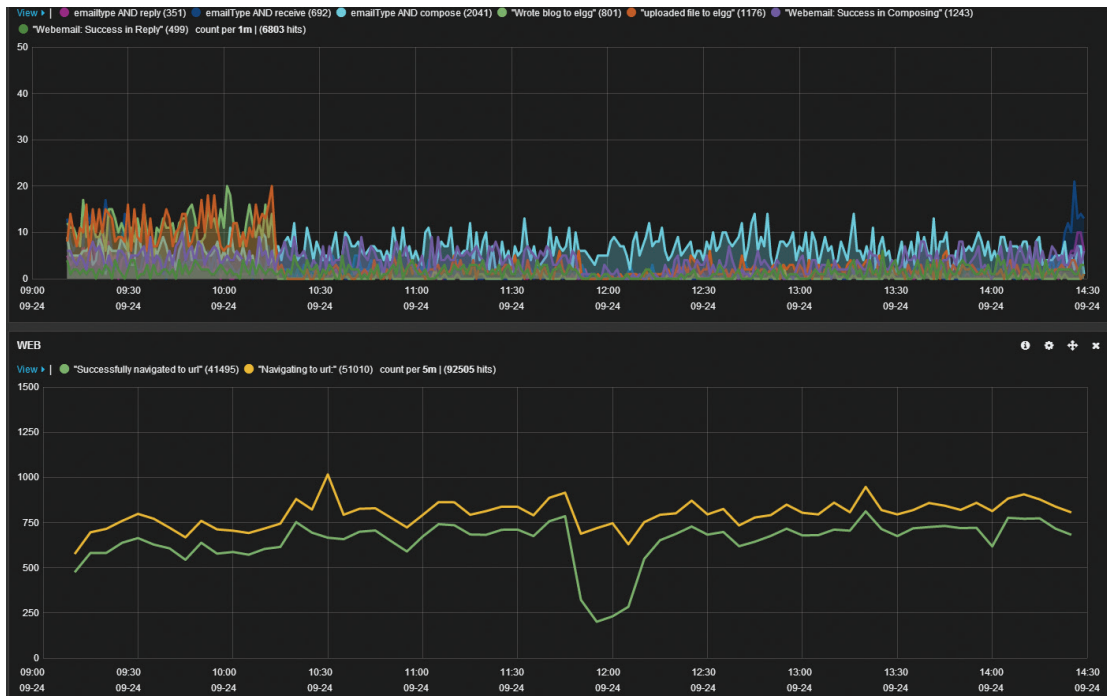
As ranges become larger, more intricate, and further distributed, we need a lightweight, scalable command-and-control (C2) system to operate the traffic generators. Simultaneously, we need to monitor in real time and with high accuracy how these traffic generators are performing and fix any errors that may arise. To avoid the latency introduced by the request-response cycles of synchronous C2 systems and to help us achieve the scalability requirements, we have built an asynchronous C2 system. However, because the asynchronous system does not provide immediate feedback from the virtual users under a range operator’s control, the status monitoring problem is more difficult. We are able to achieve near-real-time reporting on the health and status of the traffic

generator by using a messaging protocol, which analyzes messages as they periodically arrive from the virtual users. When we detect that a virtual user is unresponsive, we can take steps to fix the issue or, at the very least, notify the range operators that there is a problem.

Our C2 system works by pushing data to the virtual users when they need the information. The server “knows” what these users need for configuration and state changes (i.e., whether they should be running traffic or not). Virtual users continuously report to the server a signal that indicates whether they (a) have received the correct configuration and (b) are in the correct execution state. As the server detects inconsistencies, it may send out either updated configurations or other C2 messages to transition the virtual user into the appropriate state.

Because this C2 system is built around a loosely coupled, asynchronous messaging protocol, it is easy for organizations other than Lincoln Laboratory to augment LARIAT’s capabilities by adding their own components





**FIGURE 3.** The LARIAT network traffic seen in the above visualization was produced during one day of a red/blue exercise hosted at the Laboratory. The top graph plots the counts of virtual users' actions as a function of time. For example, several users were uploading images to a social networking site (orange line) at the beginning of the exercise, but this activity drops off drastically after an hour or so. Other actions include replying to an email (fuchsia), composing an email (light blue), and writing a blog post on the social networking site (green). Shown in the lower plot are counts over time of successful (green) and attempted (yellow) website navigation instances. About halfway through the plot, the number of successful navigations to the website plummets, perhaps because the web server became overloaded or a router was misconfigured.

(e.g., actuators) into LARIAT. A very near-term goal for the LARIAT development team is to break out the necessary components of this C2 system into a separate module that has very clear integration points for third parties. Then, a simple integration path could be created for traffic generators that are not built at Lincoln Laboratory.

#### VISUALIZATION AND ANALYTICS

To help range operators build the necessary mental model of the entire range, we provide a visualization of the range state. The visualization shows the virtual user workflow states so that range operators understand if and when the virtual users are ready to start execution. These workflow states progress as follows:

1. There is no indication that the virtual user is available (i.e., before LARIAT installation).
2. The virtual user checks in at some point in time.
3. The virtual user is configured with a behavior model and ready to start executing.

Additionally, separate from the workflow state, virtual users are either responsive or unresponsive, determined by whether they have checked in recently. We give range operators a way to quickly determine how traffic is running and what, if any, parts of the range need to be fixed.

The fairly high-level status reporting and visualization described above is for a single virtual user. We have also built aggregate visualizations of large portions of the virtual users within the network so that the range operator can, for example, see where network traffic is flowing. The process for building visualizations begins with each actuator logging its actions as it performs them. These logs are then sent to a centralized server that stores them and makes them available for analysis. Using these data, we can create real-time graphs of, for example, the number of successful and unsuccessful website navigation attempts (Figure 3). Too many failed navigation attempts could indicate to the range operator that there is a problem with the web servers or the routers that allow traffic

to flow through them. We provide a range of out-of-the-box queries and visualizations for actuator data but also allow users to write custom queries against the same data so they can monitor the activities that are most relevant to their events.

### Future Work

The LARIAT technology has recently been licensed to SimSpace Corporation ([www.simspace.com](http://www.simspace.com)) for commercial use in their products and services. Lincoln Laboratory intends to continue driving toward increased range fidelity and to build more sophisticated tools for range operators to monitor the health and status of the range. Specifically, we will enhance our modeling engine with features that allow for more complex interactions with the environment, such as responding to dynamic stimuli (e.g., messaging windows popping up on the screen). Ultimately, we want to create mission activities that describe coordinated user actions and are woven into the normal background traffic. We will also be supporting additional actuator types so that we have more variation in our virtual users. Finally, we will augment our range introspection capabilities, provide better analytics, and develop more visualizations of the emulated-user log data to make the jobs of range operators and event analysts easier.

### Acknowledgments

The author would like to thank the many people who were instrumental to the design and development of the tools mentioned in this article. I would like to recognize the LARIAT support team who designed and built the ALIVE tool and deploy these tools within organizations across the country; the LARIAT team, in particular Sarah Chmielewski, the current project lead and one of the first developers to start the LARIAT redesign with me, and Mark Mazumder, who was instrumental in driving us toward better designs and tools; and the original creators and developers of LARIAT without whom the project wouldn't have existed, specifically Lee Rossey and Douglas Stetson, who were each involved, at different times, with managing the range tools projects. Numerous others have supported this program over the years, including the Common Cyber Event Representation and Attack Framework teams. The hard work of everyone involved does not go unrecognized. ■

### References

1. J. Sciutto, "OPM Government Data Breach Impacted 21.5 Million," CNN website, 10 Jul. 2015, available at <http://www.cnn.com/2015/07/09/politics/office-of-personnel-management-data-breach-20-million/>.
2. K. Granville, "9 Recent Cyberattacks Against Big Businesses," *The New York Times*, 5 Feb. 2015, available at [http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?\\_r=0](http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=0).
3. "Cyber Warfare: Sabotaging the System," CBS News website, 6 Nov. 2009, available at <http://www.cbsnews.com/news/cyber-war-sabotaging-the-system-06-11-2009/>.
4. L.M. Rossey, R.K. Cunningham, D.J. Fried, J.C. Rabek, R.P. Lippmann, J.W. Haines, and M.A. Zissman, "LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed," *Proceedings of the 2002 IEEE Aerospace Conference*, vol. 6, pp. 2671-2682.
5. "Network Testing with Simulated Traffic. Does Realism Matter?" An Ixia BreakingPoint Case Study, White Paper 915-3128-01, Rev. C., Aug. 2014.
6. C.V. Wright, C. Connelly, T. Braje, J.C. Rabek, L.M. Rossey, and R.K. Cunningham, "Generating Client Workloads and High-Fidelity Network Traffic for Controllable, Repeatable Experiments in Computer Security," *Proceedings of the 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2010, pp. 218-237.
7. Metasploit company website, available at <http://www.metasploit.com/>.
8. J. Mathieu, J. Melhuish, J. James, P. Mahoney, L. Boiney, and B. White, "Multi-scale Modeling of the Air Operations Center," The MITRE Corporation, Technical Papers, November 2007, available at [http://www.mitre.org/sites/default/files/pdf/06\\_1497.pdf](http://www.mitre.org/sites/default/files/pdf/06_1497.pdf).

### About the Author



**Timothy M. Braje** is a technical staff member in Lincoln Laboratory's Secure Resilient Systems and Technology Group, where he works on adaptive computing platforms and quantum computing algorithms. He also has interests in functional programming, formal methods for verifying software systems, programming languages, and constructive theorem proving. Between 2009 and 2015, he led the effort to architect and build the Laboratory's next-generation advanced cyber tools platform, including tools for range control, visualization, and traffic generation. Prior to joining the Laboratory in 2009, he worked at MyVest, Solidware Technologies, and Coverity, building systems to help automate financial investment management and to help software developers analyze and improve the quality of their products. He holds a bachelor's degree in physics from the University of Florida and a doctoral degree in physics from Stanford University.

# Threat-Based Risk Assessment for Enterprise Networks

Richard P. Lippmann and James F. Riordan

Protecting enterprise networks requires continuous risk assessment that automatically identifies and prioritizes cyber security risks, enables efficient allocation of cyber security resources, and enhances protection against modern cyber threats. Lincoln Laboratory created a network security model to guide the development of such risk assessments and, for the most important cyber threats, designed practical risk metrics that can be computed automatically and continuously from security-relevant network data.



**Computer networks are under constant** cyber attack. In 2013, in one of the most historically devastating insider attacks, Edward Snowden exfiltrated 1.7 million documents from the National Security Agency [1]. That same year, the security company Mandiant released a report on the likely Chinese government-sponsored cyber espionage group APT 1 (for advanced persistent threats), who stole 100s of terabytes of proprietary information from at least 141 organizations by maintaining a long-term presence in the victims' networks [2]. More recently, the widespread Heartbleed [3] and Shellshock [4] attacks exploited vulnerabilities in common Internet web and encryption services, and the U.S. Office of Personnel Management announced the theft of sensitive information, including the Social Security numbers of 21.5 million individuals, from the background investigation databases for persons seeking government clearances [5].

Assessing important security risks at large enterprises to make sure that risks from all current threats are addressed is costly, takes time, requires trained security specialists, and involves a high degree of organizational accountability. As a result of these factors, many organizations adopt a best-practices approach by installing popular baseline security controls, such as antivirus software and email spam filters, and scanners that find and patch software vulnerabilities. Because this approach is not tailored to meet the unique security needs of individual organizations, resources may be wasted on implementing unnecessary controls while important threats go unaddressed. Metrics that result from this approach (e.g., counts of files scanned or of high-severity vulnerabilities)

are difficult to interpret because their relationship to risk from modern threats is unknown.

Many organizations also perform some type of qualitative risk assessment in which a list of threats is considered and the likelihood and impact of each threat is rated on an ordinal scale from low to high. Threat management involves addressing those threats for which the likelihood and impact are both high. This approach can be effective when performed by skilled security practitioners who understand an enterprise network, can enumerate all threats and their likelihoods, and can accurately assess the effectiveness of controls against the threats and their expected impacts. Unfortunately, it is difficult to find such trained practitioners, and even skilled security experts can miss key threats or misunderstand the impact of breaches.

Our goal is to automate and improve the current state of the art in risk assessment. Using a list of important modern threats, we describe how to compute risk for each threat and also how to specify the data required to compute risk. We provide an initial list of threats that can be updated over time to capture recent threat types. The required data can be gathered online in real time to provide continuous risk assessment. The resulting risk values can be compared across threats, time, and different enterprises. The accuracy of this method should approach that of the best skilled security experts because we provide a carefully selected list of threats and specify how to compute risk objectively without relying on unsupported qualitative human judgments.

### Modeling and Mitigating Modern Threats

Recently, security experts from companies, government agencies, and academia joined forces to create a set of security controls and adversary models specifically focused on modeling and preventing advanced persistent threats and other current threats. Their work led to the 20 critical security controls shown in Table 1 and described by the Council on CyberSecurity [6]. These controls are prioritized by their capability to provide a direct defense against attacks. Subcontrols from the first four critical controls constitute most of the so-called “five quick wins” that have the most immediate impact on preventing common attacks [6]. The remaining controls provide additional protection against attacks. These widely used controls are the most effective and specific set of tech-

nical measures available to detect, prevent, respond to, and mitigate damage from current threats. We have used these critical controls to prioritize the threats that should be addressed in enterprise networks and to recommend and model controls that should be in place to mitigate those threats. Metrics we have developed focus on automatically computing risk for the most important critical control threats, and they directly model the effectiveness of critical controls that should be in place.

### Security Metrics

We have developed 9 security metrics, each of which is associated with a specific cyber threat and critical control(s) from the critical controls document [6]. Table 2 lists these metrics, the threat each metric addresses, and the control(s) from Table 1 that mitigate each threat. Each metric is assigned a Lincoln Risk (LR) number ranging from LR-1 to LR-9. The LR-3 metric, for example, is concerned with attackers who search for and exploit known software vulnerabilities in a network. The risk of these attacks is reduced when the durations of software vulnerabilities (i.e., the time between when a vulnerability is first published to when it is removed) are shortened. These durations can be shortened by performing continuous vulnerability assessment and remediation as suggested by critical control 4, which includes detecting and patching vulnerabilities more frequently. The LR-3 metric is discussed further in the “Two Example Metrics” section.

The first four metrics we developed (LR-1 to LR-4) focus on the same management areas as the “five quick wins” mentioned previously. These metrics prevent common attacks, such as gaining access to devices via well-known default passwords and accessing computers remotely by using previously published exploits. They also support higher-numbered metrics by providing important baseline observations concerning the presence and characteristics of devices, software, and configurations that exist in a network. Detailed descriptions of LR-1 to LR-4 are available in Lippmann et al. [7]. The next three metrics (LR-5 to LR-7) focus on users’ roles, credentials, and accounts, and they cover insider attacks, credential theft, and attacks that require physical access to victim devices. LR-8 concerns user behaviors that enable attacks, such as providing passwords over the phone or in response to an unverified email.

LR-9 addresses the boundaries added to networks to prevent outside attacks. These 9 metrics cover the highest-priority attacks and controls listed in the Council on CyberSecurity document [6] that can be automated and computed using continuous measurements.

**Table 1. Twenty Critical Security Controls**

1	Inventory of authorized and unauthorized devices
2	Inventory of authorized and unauthorized software
3	Secure configurations for hardware and software on mobile devices, laptops, workstations, and servers
4	Continuous vulnerability assessment and remediation
5	Malware defenses
6	Application software security
7	Wireless access control
8	Data recovery capability
9	Security skills assessment and appropriate training to fill gaps
10	Secure configurations for network devices such as firewalls, routers, and switches
11	Limitation and control of network ports, protocols, and services
12	Controlled use of administration privileges
13	Boundary defense
14	Maintenance, monitoring, and analysis of audit logs
15	Controlled access based on the need to know
16	Account monitoring and control
17	Data protection
18	Incident response and management
19	Secure network engineering
20	Penetration tests and Red Team exercises

### Metrics Development

To develop security metrics, we first develop simple but realistic attack models to guide the four steps of the processing loop shown in Figure 1. Attack models establish

1. what security conditions must be observed to determine the risk of an attack;
2. how to compute the risk of an attack on the basis of observed security conditions;
3. how to prioritize the risk of an attack across network entities, such as persons, devices, and accounts; and
4. how to design the network so it is easy for network administrators to take actions that mitigate risk and to eliminate security conditions that enable attacks.

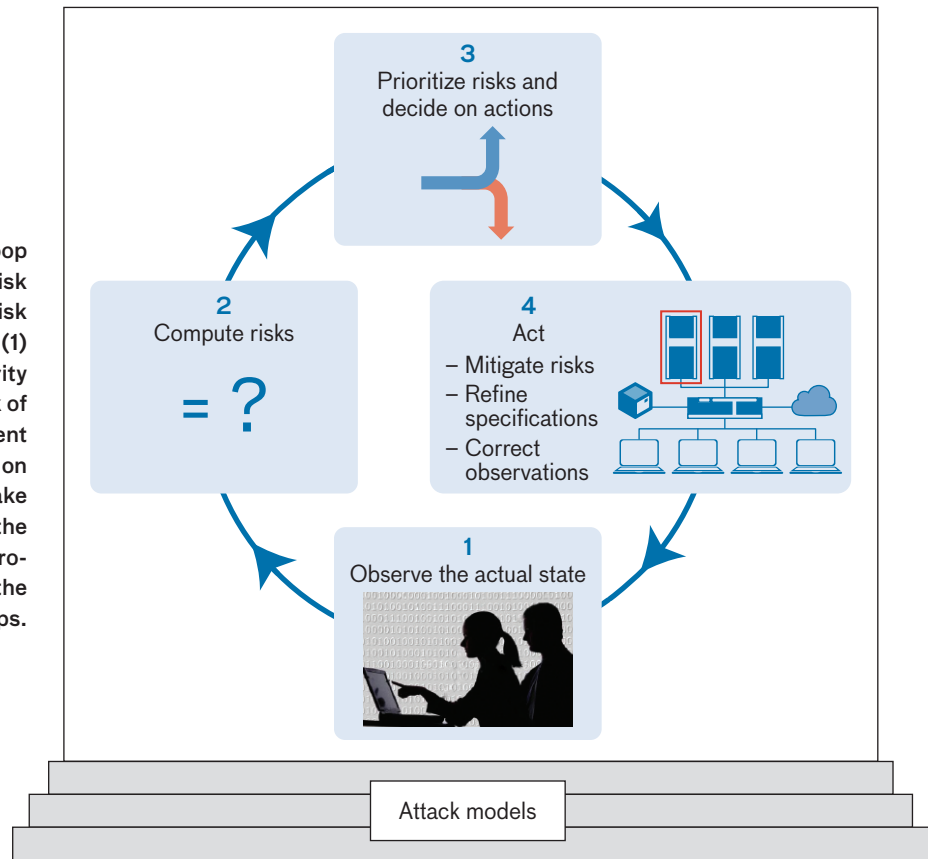
The first step of the processing loop in Figure 1 is to observe relevant security conditions in a network. For managing software vulnerabilities (LR-3), network vulnerability scanners could be used to find vulnerable servers. Observation techniques for managing persons (LR-5 to LR-8) include accessing personnel records for logins (and attempted logins) and for screening, indoctrination, and training to determine granted trust levels and user roles.

The second step of the processing loop is to use these observations to compute risk. For LR-3, this step involves determining the duration of known vulnerabilities and the probability that attackers observe and exploit these vulnerabilities to compromise devices. For managing trust (LR-5), this step consists of analyzing user trust levels, role assignments, accounts, and approaches that improve the security of user authentication (such as two-factor authentication) to compute the overall systemic risk of insider attacks.

The third step of the processing loop is to prioritize risks according to their risk values calculated in step two and to prioritize mitigation actions on the basis of their effectiveness and other practical concerns (e.g., the cost of the mitigations). Finding the most effective approach to mitigate risk involves performing offline analyses using the risk computation capability of step two to compare the effectiveness of different actions. Mitigations range from immediate rapid fixes, such as patching software, to longer-term changes, such as adding separation of duties in which two persons are required to complete a task that provides access to a high-value asset (e.g., a bank vault).

The fourth step of the processing loop is to mitigate the risks prioritized in step three. Mitigations can be dis-

**FIGURE 1.** A processing loop required to measure and reduce risk for the threat from each Lincoln Risk (LR) metric requires four steps: (1) observe relevant network security conditions, (2) compute the risk of threats to the network in its current state, (3) prioritize risks and decide on actions to mitigate risks, and (4) take action to mitigate risks or improve the risk computation and mitigation processes. Attack models provide the foundation for each of these steps.



played on a local dashboard containing counts of defects or security conditions that must be remediated to reduce risk. In addition, specifications, such as lists of computers allowed on a network, may have to be refined because of the deployment of new computers. Similarly, observations, such as lists of computers actually on a network, may have to be corrected as a result of inaccurate automated measurements.

**A Maturity Model with Three Metric Components**

The maturity model shown in Figure 2 is essential to our overall approach of improving enterprise security. This model allows (1) the gradual introduction and use of processes and tools required to assess risk, (2) the capability to observe security-relevant data, and (3) the capability to estimate risk and apply mitigations to reduce risk. The security metrics listed in Table 2 can only be used to accurately compute risk after the first two levels of metric development shown in Figure 2 are completed.

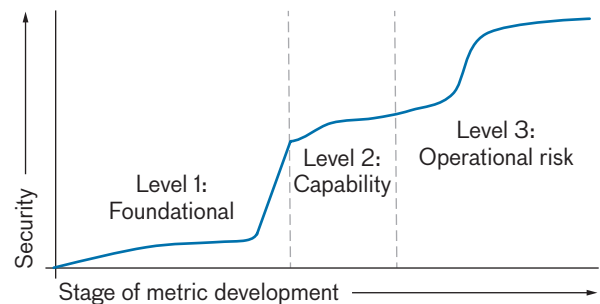
For each metric in Table 2, we develop three metric components during three maturity metric phases as seen in Figure 2. In the Level 1 maturity phase, foundational or checklist metric components are developed. These components determine whether all essential tools and procedures are in place to support continuous monitoring. During this phase, system administrators develop an understanding of their systems and the most potentially damaging threats. They begin to implement security control processes as described in Table 1, add tools to gather data, and develop mitigations. For LR-3, administrators would continuously monitor all devices and identify known software vulnerabilities on those devices. A control process, such as patching, would be initiated to eliminate the vulnerabilities. Even if security measures are not directly implemented in this stage, there is a significant security impact because improved network hygiene improves administrators’ understanding of the network topology and enhances their visibility of network security conditions.

**Table 2. Lincoln Laboratory Security Metrics**

LINCOLN RISK (LR) NUMBER	THREAT	MITIGATING CRITICAL CONTROL(S)
LR-1	Attackers compromise unauthorized devices	1
LR-2	Attackers compromise unauthorized or prohibited software	2, 5, 11
LR-3	Attackers exploit known software vulnerabilities	4
LR-4	Attackers exploit insecure configurations	3, 7, 11
LR-5	Attackers launch insider attacks	9, 13, 15, 16
LR-6	Attackers steal credentials and exploit weak authentication	7, 10, 12, 13, 16
LR-7	Attackers exploit account and physical access privileges	3, 12
LR-8	Users perform actions that enable attacks	9
LR-9	Attackers penetrate network boundaries; sensitive information exits network boundaries	7, 11, 13

In the Level 2 maturity phase, capability metric components are developed. These components determine whether the coverage, frequency, and accuracy of observations are sufficient to estimate risk. Specifications, such as lists of the types of software allowed on each device or of the correct configurations of each device, are also created in this phase. Until the values of capability metrics are low (indicating good capabilities), risk cannot be computed accurately. Security improves slowly during this stage, as indicated by the slope of the graph in Figure 2. This slow progression is due to further discovery and repair of security issues as coverage improves across the entire network, specifications are developed, previously missed short-duration security conditions are identified because of more frequent observations, and security conditions are accurately measured.

In the final maturity phase, Level 3, operational risk metric components are developed. These components compute the actual risk associated with a given threat. They can be used to determine which devices, software packages, misconfigurations, vulnerabilities, persons, or other security conditions are responsible for the greatest increase in risk. With this information, network personnel can take actions that reduce risk. Operational risk metrics continuously assess the risk of the most import-



**FIGURE 2.** The notional curve in this security maturity model suggests that network security increases (and risk of attacks decreases) as a particular metric is further developed.

ant threats in real time by estimating the impact caused by attackers directly compromising assets (e.g., proprietary information, hardware, services).

The risk score for any threat is calculated by multiplying the value of assets under attack by the probability that an insider or outsider attack succeeds (see equations in “A Metric for Software Vulnerabilities” section). In cases of fraud and theft, the asset value is easy to assign because it is simply the total value of money or goods stolen. In most cases, however, the asset value is assigned subjectively and is related to how an attack would impact

the performance of an organization or persons that rely on or are affected by that organization. Values must be assigned for attacks that compromise an asset's confidentiality, integrity, or availability. To achieve low operational risk scores, critical assets must be assigned high values and be provided with strong protective controls. For example, the confidentiality of details supplied by persons to obtain Top Secret clearances might be violated by foreign governments who exfiltrate this information and use it to identify undercover U.S. agents posted to their countries [8]. Databases containing such details should qualitatively be assigned a high value because stolen data could compromise an agent's usefulness or life. In other cases, network services should be assigned a high value. Denial-of-service attacks are often used to render network services inaccessible. The Department of Veterans Affairs, the Social Security Administration, the Federal Emergency Management Agency, and other agencies that provide government services directly to citizens need to implement strong protective controls against these attacks to ensure that network services can be accessed, especially during emergencies.

### Metric Components Design Guidelines

Three key principles guide metric development: each metric must (1) be simple to understand and implement, (2) practically estimate the risk of one specific important threat, and (3) motivate actions to reduce the risk of that threat.

Maintaining simplicity and practicality and estimating the risk of a specific threat are fairly straightforward tasks. Simple risk prediction models that utilize existing security tools to gather data are used when possible. The order in which we develop the metrics is chosen with practicality and effectiveness in mind: earlier metrics provide situation awareness and baseline information, such as lists of devices and their software, required by later metrics. For example, metric LR-1 provides a device list that is used by all other higher-numbered metrics; LR-2 provides a device software list used by metrics LR-3 and LR-4.

To motivate system administrators to improve security controls, metrics must be objective, well defined, and visible to all involved in the security process so the metric scores can be understood to be fair. We adopted the convention that high scores for metrics are bad and low scores (near 0%) are good; when continuous vulnerability monitoring was implemented at the U.S. Depart-

ment of State, this scoring system was shown to be more likely to encourage administrators to improve their performance than the 0% (bad) to 100% (best) test scoring traditionally used in schools [7]. Two other motivating features are (1) incremental improvements in security controls lead to incremental improvements in metrics and (2) the overall difficulty of obtaining a low (good) metric score increases slowly over time as metric parameters change. Initially, it can be relatively easy to get a good score; however, as an enterprise's capabilities improve and as response times to mitigate insecure conditions shorten, obtaining a good score can become more difficult. Slowly increasing the difficulty of obtaining a low metric score should lead to long-term overall security improvements because system administrators will have to continually improve security controls and processes in order to maintain a low score.

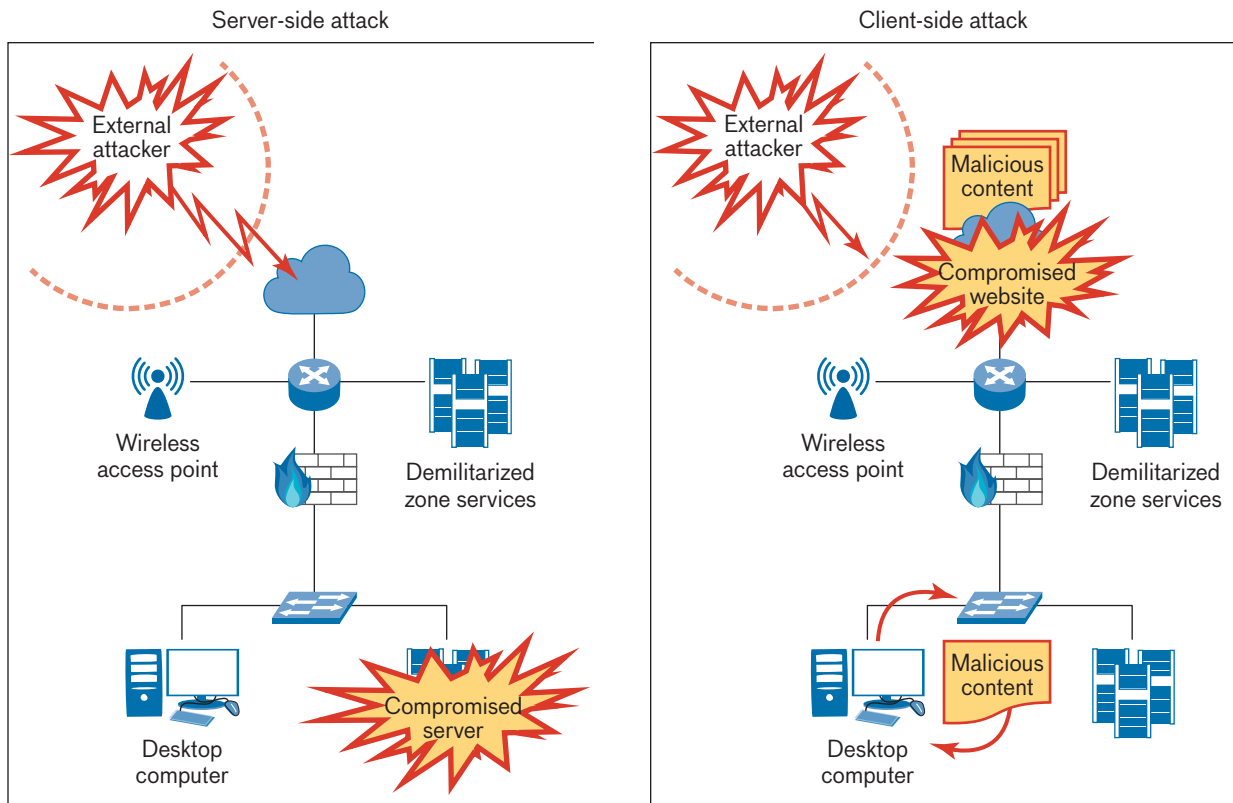
### Two Example Metrics

The following sections provide detailed examples of two metric types. One metric (LR-3) focuses on attackers who detect and exploit known software vulnerabilities. The risk of attack that the metric computes could, at least conceptually, be reduced to zero if all known vulnerabilities are immediately patched. Obtaining good low scores for this metric requires continuously observing all known vulnerabilities and eliminating them as soon as possible. The second metric (LR-5) focuses on insider attackers who use allowed privileges to exfiltrate data. The risk of insider attacks can never be eliminated because any person can decide to act maliciously at any time. Obtaining good low scores for this metric involves monitoring security screenings, roles, and privileges for all persons with access to a network and then computing the expected risk. This computation takes into account security measures, such as compartmentalization (a network is broken into separate "compartments" that can be accessed by users only on a need-to-know basis) and separation of duties.

### A Metric for Software Vulnerabilities

LR-3 is concerned with managing known software vulnerabilities. Figure 3 shows the two attack models that we developed for this metric. In server-side attacks, external attackers scan for vulnerabilities in web, database, email, and other servers open to the Internet. Once found, these vulnerabilities are exploited by attackers





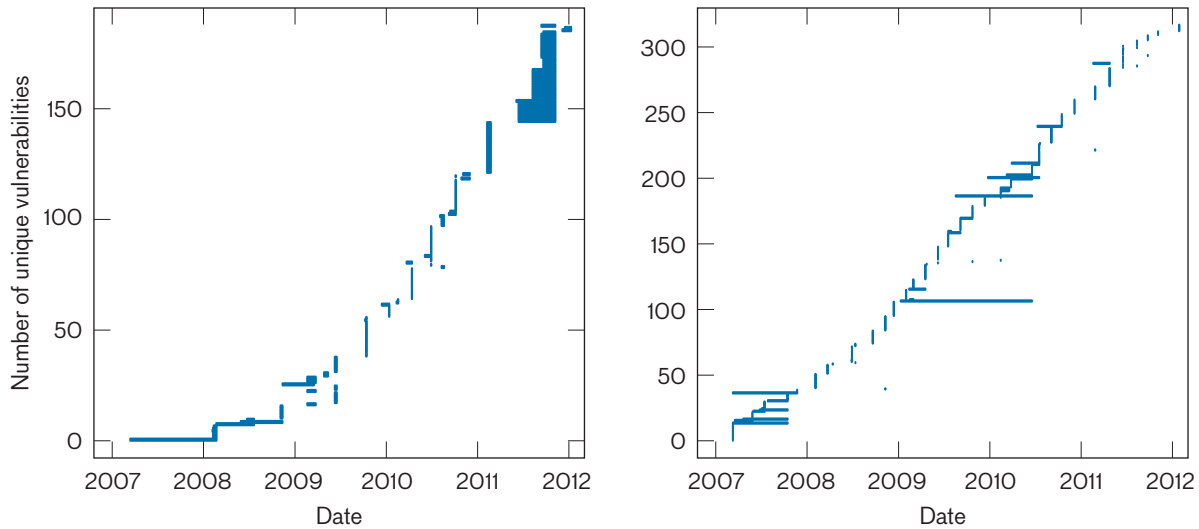
**FIGURE 3.** Attack models for server- and client-side attacks were developed in support of the LR-3 metric. In a server-side attack (left), remote attackers find and compromise internal servers with known vulnerabilities. In a client-side attack (right), users download remote content with embedded malware (often from a website) and view the content with a browser or other client software with known vulnerabilities that are exploited by the malware.

to gain control of the servers. In the more common client-side attacks, attackers embed malware in webpages, images, documents, movies, or other content and place that content on websites or transmit it via email or social media networks. When users visit websites infected with malware or view content with embedded malware, their computers are compromised because of vulnerabilities in the client-side software used to display the remote content. Persons can be lured to malicious websites owned by attackers (spear-phishing attacks), or attackers can infect websites that persons are known to visit (watering-hole attacks).

Client-side attacks depend on the occurrence of client-side vulnerabilities in web browsers and other client software used to view remote content. The risk of attacks is reduced when windows of vulnerability (i.e., the durations the vulnerabilities are present) are minimized. This minimization can be accomplished by patching vulnerabil-

ities whenever a patch is released or by rapidly detecting vulnerabilities and performing targeted patching of those vulnerabilities. Figure 4 illustrates the large number of client-side vulnerabilities in a popular PDF viewer (Acroread) and web browser (Firefox) that were discovered between 2007 and 2012. Persons browsing the Internet during this time could have been vulnerable to compromise if they encountered malware exploiting vulnerabilities that had not been rapidly detected and patched by defenders. In general, there are fewer server-side vulnerabilities per year because server-side software is generally more mature and less complex than are modern browsers and other content-viewing client software.

The foundational and capability metric components of LR-3 are low when all devices, client software, and servers are identified and when there is an up-to-date mechanism for rapidly detecting and then mitigating vulnerabilities. Instead of providing detailed equations



**FIGURE 4.** These cumulative five-year histograms plot the client-side vulnerabilities for the popular PDF viewer Acroread (left) and the web browser Firefox (right). Per year, there were roughly 40 to 60 vulnerabilities in each software product. Each vulnerability is represented by a unique horizontal line, the length of which represents the time interval from when the vulnerability is publicly announced to when a patch is made available by the software developer. Longer horizontal lines indicate instances when vulnerabilities are announced without patches, but a patch is subsequently made available. Narrow vertical lines correspond to vulnerabilities that are simultaneously announced with patches.

for foundational and capability metric components, we instead focus on the operational metric and assume defenders know precisely when vulnerabilities are first present and when they are eliminated. As noted above, the operational or risk metric for LR-3 requires that each device be assigned an asset value related to the impact of a successful attack on that device. The operational metric ( $OM$ ) for each device is then the product of the asset value ( $AV$ ) times the probability that the device is compromised over a specified time window ( $P_{DeviceCompromised}$ ):

$$OM = AV \cdot P_{DeviceCompromised} \tag{1}$$

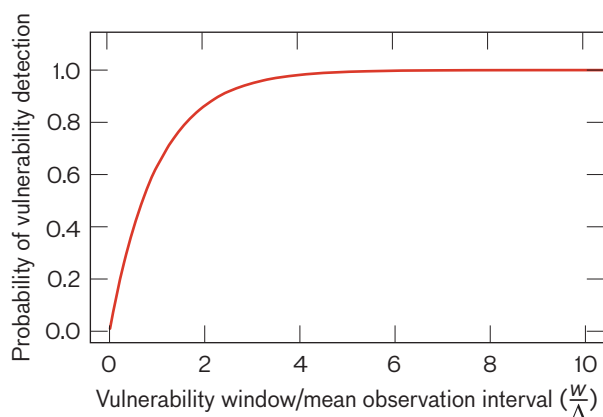
Focusing only on server-side vulnerabilities for simplicity, we say the probability that a device with a single vulnerability is compromised is equal to the probability that the vulnerability is observed or discovered by an adversary ( $P_{Observed}$ ) times the probability that the device is compromised by the adversary, given that the vulnerability is observed when the adversary has an exploit for that vulnerability, as shown in Equation (2):

$$P_{Compromised}(v) = P_{Observed}(v) \cdot P_{Compromised|Observed}(v) \tag{2}$$

We assume the probability that a device with vulnerability is compromised is related to the Common Vulnerability Scoring System (CVSS) score [9] assigned by the National Institute of Standards and Technology to each vulnerability listed in the National Vulnerability Database [10]. Specifically, we assume the probability that an attacker compromises a device with an observed vulnerability  $v$  is equal to the squared value of the CVSS score for that vulnerability, which ranges from 0.0 (low severity) to 10.0 (high severity), after it is normalized to range from 0 to 1:

$$P_{Compromised|Observed}(v) = \left\{ \frac{CVSS(v)}{10} \right\}^2 \tag{3}$$

Because the low range of CVSS scores is not frequently used, this computation leads to a more uniform and realistic distribution of compromise probabilities than the distribution obtained by simply normalizing CVSS scores. We understand that the CVSS score was not intended for this purpose, but it is the only widely available measure available across vulnerabilities, and the single highest-weighted term used to compute each CVSS score does, in fact, directly assess the exploitability of each vulnerability [9].



**FIGURE 5.** The probability that an attacker who scans server software every  $\Delta$  days with exponential Poisson interarrival times detects a vulnerability present for  $w$  days increases slowly to 1.0 as the average interval between scans becomes much less than the duration of the vulnerability.

The probability that an attacker observes a vulnerability present on a server for a window of duration  $w$  depends on how often the attacker scans the server. If attackers scan a server every  $\Delta$  days with exponential Poisson<sup>1</sup> interarrival times, then the probability that a vulnerability is observed is given by Equation (4) and shown in Figure 5. This probability is low when the average interarrival times between scans are large relative to the vulnerability window and increases to 1.0 as the average interarrival times become much smaller than the vulnerability window.

$$P_{Observed}(v, w) = 1 - e^{-\frac{w}{\Delta}} \quad (4)$$

Equation (2) can be used to compute the probability of device compromise only when there is one vulnerability on a server. Computing the probability of device compromise when multiple vulnerabilities are present, as is often the case, requires an extended attacker model that specifies the number of exploits an attacker has attempted and successfully implemented. In the stealthy attacker model, an attacker attempts to exploit only the

vulnerability with the highest probability of compromise (i.e., the vulnerability with the highest CVSS score). Another attacker model is a noisy attacker who tries an exploit for every vulnerability on the device until an exploit succeeds. We can compute risk for both of these attacker models and variations of them. If we assume that a noisy attacker tries an exploit for every vulnerability on a device and that the probabilities of success for every exploit are independent of each other, then the probability of compromising a device with multiple vulnerabilities ( $Vulns$ ) is given by Equation (5). This probability rises as the number of vulnerabilities increases and as the probability of device compromise for individual vulnerabilities increases.

$$P_{DeviceCompromised} = 1 - \prod_{v \in Vulns} \{1 - P_{Compromise}(v)\} \quad (5)$$

Equations (1) through (5) support the computation of the LR-3 operational metric for server-side attacks on one device. Across a network, the operational metric is simply the sum of the individual operational metrics for each device on the network. Computations are similar for client-side attacks, except the observation interval is the interval between exposures to client-side exploits, and the equations use client-side instead of server-side vulnerabilities.

A simulation experiment demonstrated the effect of rapid patching on the LR-3 operational metric. The simulation contained 100 hosts, each with an asset value arbitrarily set to 1.0 and running only the Firefox web browser. We made the rather pessimistic assumptions that persons browse an infected website once every 30 days and that every infected website contains exploits for all known Firefox vulnerabilities. We also assumed that attackers require one week after the publication of a vulnerability to develop an exploit and place the exploit on websites. We used actual vulnerabilities announced in 2012 [10], including their dates and CVSS scores, to populate the simulation. The results of the simulation are shown in Figure 6; each plot shows the windows for all vulnerabilities. Vulnerabilities in the left plot are for an enterprise in which all current Firefox patches are applied every 5 days, while those in the right plot are for an enterprise in which patches are applied only every 30 days. The vertical bars represent several vulnerabilities that were announced on the same day, with the width of

<sup>1</sup> More formally, we assume that attacker observations form a Poisson process in which (1) the time between each pair of consecutive observations has an exponential distribution with parameter  $\lambda$  and (2) each interarrival time is independent of all the others. This assumption is true of many Internet phenomena, such as the times between user-initiated bursts of requests from a web browser and between requests for a particular document at a web server.

the bars indicating the vulnerability window. The operational risk metric represents the expected number of hosts compromised during 2012 and can be calculated by applying Equations (1) to (5) to client-side vulnerabilities as described in Lippmann et al. [7]. When patches are applied every 5 days, the expected number of hosts compromised is 4.9 over the year. Because each host is valued as 1, the operational risk metric has the low value of 4.9. When patches are applied only every 30 days, the expected number of hosts compromised is 98.6, a very high value. These results illustrate how sensitive LR-3’s operational risk metric is to vulnerability windows and patching frequency.

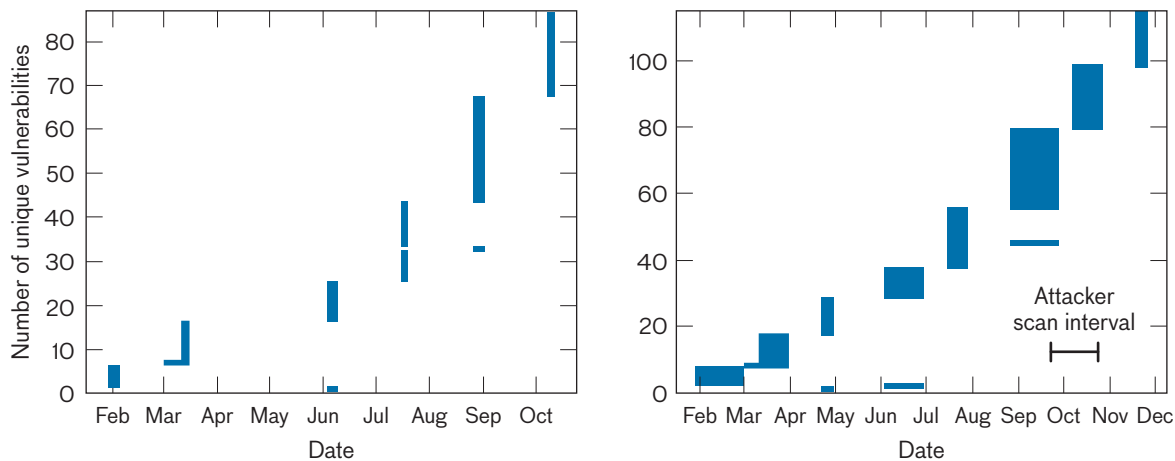
**A Metric for Insider Attacks**

Metric LR-5 computes the intrinsic risk of insider attacks, given trust levels granted to individuals, role assignments, and the controls in place to restrict access to assets. Computing LR-5 requires an estimate of the intrinsic condition of a person’s untrustworthiness, which cannot be detected by security tools but can be modeled. We assume that organizations create roles, set granted trust levels to persons, assign persons to roles, and assign privileges (to access assets) to roles as shown in Figure 7. Roles simplify management of privileges because privileges are assigned to roles rather than to individual persons and persons are assigned to roles rather than directly to privileges. For example, in Figure 7, person A is assigned to both role 1 and role 3, person B is assigned to only role 2, and person

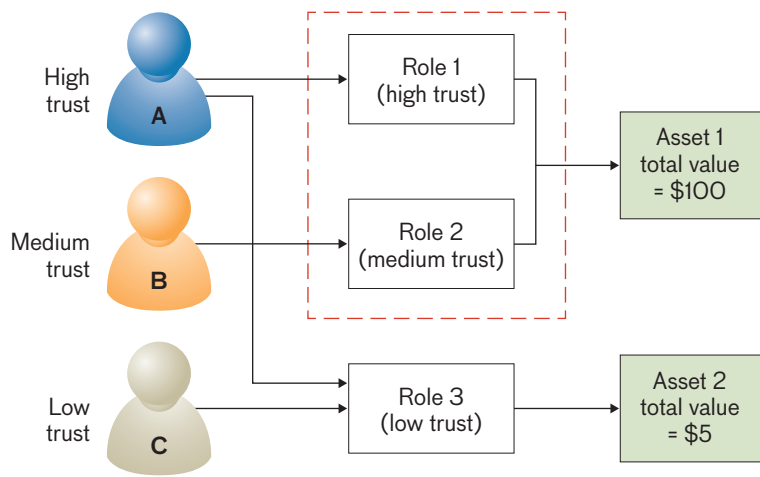
C is assigned to only role 3. These roles can be hierarchical (roles with higher trust levels inherit all the privileges from lower levels).

Each person has a granted trust level. A person with a low level of trust should not be assigned to a role that comes with privileges to access assets of high value. Figure 7 provides an example of an appropriate role assignment: person C has low granted trust and is assigned to role 3, which provides access to an asset of relatively low value (asset 2 with a value = \$5), while persons A and B have higher trust levels and thus together can access an asset of much higher value. The dotted box in Figure 7 indicates that there is a separation-of-duties rule for roles 1 and 2: to access asset 1, two different persons have to separately perform roles 1 and 2. In this example, person A and person B must simultaneously assume roles 1 and 2, respectively, to access asset 1 (akin to a double-key lock or double-password system). This multiperson procedure makes it more difficult for one malicious insider to access the highly valued asset 1.

To compute the risk from insider attacks, we need to model how persons become untrustworthy. We model persons using a Markov process with two states, trustworthy and untrustworthy, as shown in Figure 8. This model assumes that persons are either trustworthy and will never perform an insider attack or that they are untrustworthy and will perform an insider attack. After an initial screening, a fraction of persons are untrustworthy ( $P_{Untrust}$ ) and the remainder are trust-



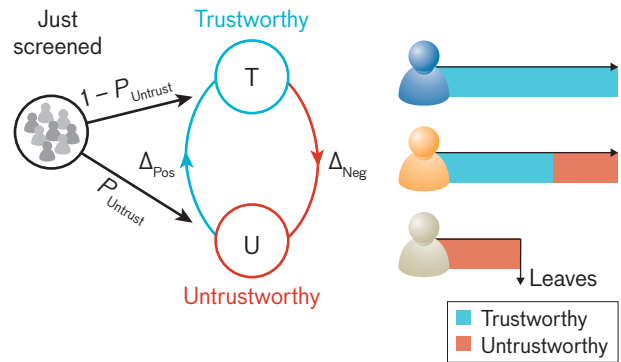
**FIGURE 6.** The LR-3 operational risk metric was computed with 5-day (left) and 30-day (right) patching policies for vulnerabilities announced in 2012. As indicated by the width of the vertical bars, the vulnerability windows are much larger when patches are applied only every 30 days as opposed to every 5.



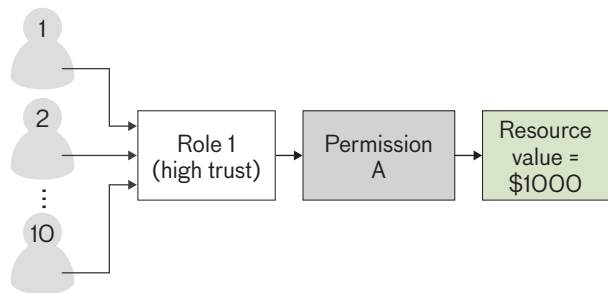
**FIGURE 7.** Three users (A, B, and C) have been granted trust levels of high, medium, and low. Each person is assigned different roles (which also have required trust levels) that provide access to assets of varying value. Two users (A and B) are required in order for either to obtain access to asset 1 because of a separation-of-duties rule.

worthy ( $1 - P_{Untrust}$ ). Over time, negative life events, such as incurring a large debt or being demoted, can cause a person to become untrustworthy ( $\Delta_{Neg}$ ). Positive life events, such as receiving a raise or recognition at work, can cause an untrustworthy person to become trustworthy ( $\Delta_{Pos}$ ). The timelines in Figure 8 show some examples of how persons are trustworthy or untrustworthy over time. Using the Markov model, we compute the long-term steady-state probabilities of persons being trustworthy and untrustworthy to compute the probability that persons with different granted trust levels are untrustworthy.

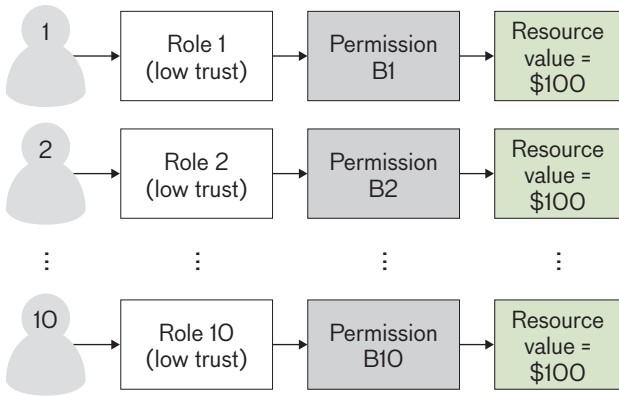
Computing insider attack risk can be extremely complex in large enterprises because there are so many combinations of user roles, ways users can be untrustworthy, and instances of separation of duties. Here, we will use a simple example to illustrate some of the important aspects of risk computation and the ways in which mitigations can reduce risk. Consider a small company started by one person who has been given access to a university professor’s intellectual property valued at \$1000. If this person has a 0.05 probability of performing an insider attack and stealing the intellectual property in one year, then the expected insider attack risk is  $\$1000 \times 0.05$ , or \$50. If the company grows to 10 employees and the probability that any one of the employees is untrustworthy is again 0.05, then the expected loss per year is roughly \$400, assuming the employees operate independently (Figure 9). This amount will certainly lead to loss of the intellectual property after a few years.



**FIGURE 8.** A Markov process is used to calculate how persons become untrustworthy or trustworthy over time. The timelines (top to bottom) show examples of a person who (1) is always trustworthy; (2) is initially trustworthy but becomes untrustworthy; and (3) is untrustworthy from the start and leaves an organization after launching an insider attack.



**FIGURE 9.** All 10 employees in an organization can access the total intellectual property worth \$1000.

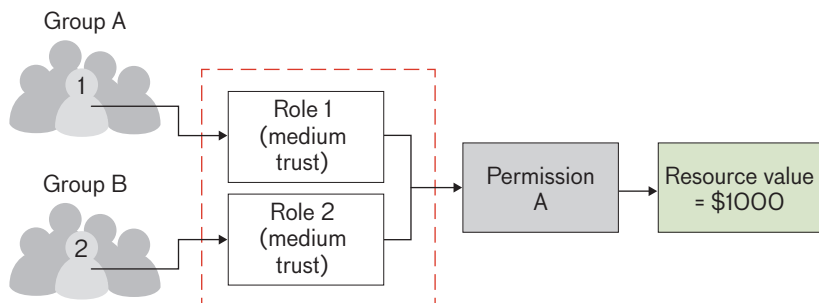


**FIGURE 10.** Compartmentalization of resources can reduce the risk of an insider attack. Persons use only the part of the resource necessary to perform their roles, and no single user accesses the resource in its entirety.

Compartmentalization is one approach that can reduce the risk of insider attacks (Figure 10). It involves breaking resources into separate components that are each only accessed by one user or role. Compartmentalization is possible if individuals’ roles only require access to parts of a high-value resource. With compartmentalization, the \$400 risk can be reduced back down to \$50 per year—the same risk as that when only one person accesses the total resource value.

As shown in Figure 11, separation of duties is another approach that can reduce the risk of insider attacks. It involves requiring approval from multiple users to access a high-value resource, such as administrative access to a central database, to all computers in a company, or to a machine capable of writing data to a USB storage key or DVD disk. Separation of duties can also be used to reduce the \$400 risk to roughly \$50 per year. In this figure, access to the resource can only be provided when two persons agree to give permission, preventing a single insider

**FIGURE 11.** Separation of duties can reduce risk of untrustworthy individuals gaining access to a high-value resource. In this example, users are separated into two groups (A and B), each containing five persons. One person from group A who can access role 1 and one person from group B who can access role 2 must be present to obtain permission A, which is needed to access the resource.



from gaining unauthorized access. Separation of duties is often used to reduce fraud but is also being applied in networks because of heightened concerns about insider attacks and data exfiltration.

**Future Directions**

Vulnerability risk analysis needs to expand to include the analysis of complex multistage attacks and of approaches that can be used to discover the most effective network-wide defensive strategies. Attack graphs can help in the analysis of multistage attacks, in which attackers gain an initial foothold on the network and proceed to take over the entire network by compromising more and more devices. We have already developed tools that perform attack graph analysis on large enterprise networks (e.g., Ingols et al. [11]) and have begun to construct attack graphs with data from LR-1 to LR-4 metrics. Such analyses can identify key insecure network conditions that enable attacks (e.g., a firewall with outdated filtering rules that permits Internet access to internal databases) and can be used to explore the effectiveness of defensive measures.

Future work will also involve modeling the risk reduction made possible by using approaches described in critical controls 8 and 18–20 [6]. We also need an approach that simultaneously estimates the overall risk from all types of attacks and accurately determines the effectiveness of complex defense strategies. Network simulations that model multiple types of defenses and attacks have been initiated. So far, we have modeled only a few attacks and mitigations. Our goal is to scale this modeling until all important attacks and mitigations are included in the network simulations and we can estimate overall risk over long time intervals and study dynamic attacker and defender models. Such simulations can inform strategic decisions in a rapidly varying adversarial environment. ■

## Acknowledgments

We would like to thank Kimberly Watson, George Moore, and John Streufert for their support and insights into cyber security. We would also like to thank the many Lincoln Laboratory staff members who have contributed to this effort, including David Bigelow, Thomas Hobson, Robert Lychev, Sebastian Neumayer, Shannon Roberts, Era Vuksani, Neal Wagner, David Weller-Fahy, and Tamara Yu.

## References

1. G. Greenwald, *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State*. London: Hamish Hamilton, 2014.
2. "APT1: Exposing One of China's Cyber Espionage Units," Mandiant Intelligence Center Report, 2013, pp. 1-74.
3. H. Kelly, "The 'Heartbleed' Security Flaw That Affects Most of the Internet," CNN website, 9 Apr. 2014, available at [www.cnn.com/2014/04/08/tech/web/heartbleed-openssl/](http://www.cnn.com/2014/04/08/tech/web/heartbleed-openssl/).
4. N. Perlroth, "Companies Rush to Fix Shellshock Software Bug as Hackers Launch Thousands of Attacks," *The New York Times*, 26 Sept. 2014, available at [www.bits.blogs.nytimes.com/2014/09/26/companies-rush-to-fix-shellshock-software-bug-as-hackers-launch-thousands-of-attacks/?\\_r=0](http://www.bits.blogs.nytimes.com/2014/09/26/companies-rush-to-fix-shellshock-software-bug-as-hackers-launch-thousands-of-attacks/?_r=0).
5. U. S. Office of Personnel Management, "Cybersecurity Resource Center, Cybersecurity Incidents," available at <https://www.opm.gov/cybersecurity/cybersecurity-incidents/>.
6. "The Critical Security Controls for Effective Cyber Defense," version 5.0, Council on CyberSecurity, SANS Institute, available at [www.sans.org/media/critical-security-controls/CSC-5.pdf](http://www.sans.org/media/critical-security-controls/CSC-5.pdf).
7. R.P. Lippmann, J.F. Riordan, T.H. Yu, and K.K. Watson, "Continuous Security Metrics for Prevalent Network Threats: Introduction and First Four Metrics," MIT Lincoln Laboratory Project Report IA-3, 22 May 2012, available at [https://www.ll.mit.edu/mission/cybersec/publications/publication-files/full\\_papers/2012\\_05\\_22\\_Lippmann\\_TechReport\\_FP.pdf](https://www.ll.mit.edu/mission/cybersec/publications/publication-files/full_papers/2012_05_22_Lippmann_TechReport_FP.pdf).
8. M. Mazzetti and D.E. Sanger, "U.S. Fears Data Stolen by Chinese Hacker Could Identify Spies," *The New York Times*, 24 July 2015, available at <http://www.nytimes.com/2015/07/25/world/asia/us-fears-data-stolen-by-chinese-hacker-could-identify-spies.html>.
9. P. Mell, K. Scarfone, and S. Romanosky, "CVSS: A Complete Guide to the Common Vulnerability Scoring System Version 2.0," National Institute of Standards and Technology, June 2007, pp. 1-23, available at [www.first.org/cvss/cvss-v2-guide.pdf](http://www.first.org/cvss/cvss-v2-guide.pdf).
10. "National Vulnerability Database," National Institute of Standards and Technology, 2015, available at <https://nvd.nist.gov/>.
11. K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling Modern Network Attacks and Countermeasures Using Attack Graphs," *Proceedings of the Annual Computer Security Applications Conference*, 2009, pp. 117-126.

## About the Authors



**Richard P. Lippmann** is a Lincoln Laboratory Fellow working in the Cyber Analytics and Decision Systems Group. He joined Lincoln Laboratory in 1981. His research interests include aids for the hearing impaired, speech recognition, pattern classification, neural networks, and cyber security. He has taught three courses on machine learning; has been an IEEE Distinguished Lecturer; won the first IEEE Signal Processing Magazine Best Paper Award for an early article on neural networks; and has authored or coauthored more than 100 papers on topics including speech recognition, machine learning, and cyber security. He served as the program chair of the Research in Attacks, Intrusions, and Defenses Workshop; the Neural Information Processing Systems (NIPS) annual conference; and the NIPS Workshop on Machine Learning in Adversarial Environments. He has participated in four national-level government studies on cyber security. He received a bachelor's degree in electrical engineering from the Polytechnic Institute of Brooklyn in 1970 and a doctoral degree in electrical engineering from MIT in 1978.



**James F. Riordan** is a technical staff member in the Cyber Analytics and Decision Systems Group. His research interests include operational security, applied cryptography, risk assessment, resilient computing, and the semantic web. Prior to joining the Laboratory in 2009, he was a researcher at the IBM Research Laboratory in Zurich, Switzerland, for 12 years. During this time, he led numerous security-related projects on topics ranging from mobile computing to intrusion detection to web-centric trust enhancement, was named an IBM Master Inventor, and served on the executive board of the European Union's Resilient Computing Network of Excellence. He received a bachelor's degree in mathematics from the University of Massachusetts, Amherst, in 1990 and a doctoral degree in mathematics from the University of Minnesota in 1997. While pursuing his doctoral studies, he was a member of the Architecture, Research, and Technology Group at the Secure Computing Corporation and a consultant to Counterpane Internet Security.

# Finding Malicious Cyber Discussions in Social Media

Richard P. Lippmann, William M. Campbell, David J. Weller-Fahy,

Alyssa C. Mensch, Giselle M. Zeno, and Joseph P. Campbell

Today's analysts manually examine social media networks to find discussions concerning planned cyber attacks, attacker techniques and tools, and potential victims. Applying modern machine learning approaches, Lincoln Laboratory has demonstrated the ability to automatically discover such discussions from Stack Exchange, Reddit, and Twitter posts written in English.



## **Criminal hackers often use social media**

networks to discuss cyber attacks, share strategies and tools, and identify potential victims for targeted attacks. Analysts examining these discussions can forward information about malicious activity to provide system administrators with an advance warning about attacker capabilities and intent. As described in the February 2016 Federal Cybersecurity Research and Development Strategic Plan [1], system administrators must deter, protect networks from, and detect cyber attacks and then adapt after successful attacks (Figure 1). To enable system administrators to be more successful at these four tasks, advance warnings let system administrators focus on specific attack component types, time intervals, and targets. For example, prior to the anticipated cyber attacks on Israeli government websites by the hacking group Anonymous, government analysts were monitoring hackers on Facebook and in private chat rooms. As a result, system administrators were prepared to counter distributed denial-of-service attacks and defacement of government websites. Israel temporarily suspended some international traffic to these sites and advised employees to not open emails for five days. Teams were available to respond to successful attacks and repair or restore websites. Because of Israel's careful preparation, this cyber assault only succeeded in bringing down a few websites for a short period of time [2].

Monitoring social media networks is a valuable method for discovering malicious cyber discussions, but analysts currently lack the automation capabilities needed





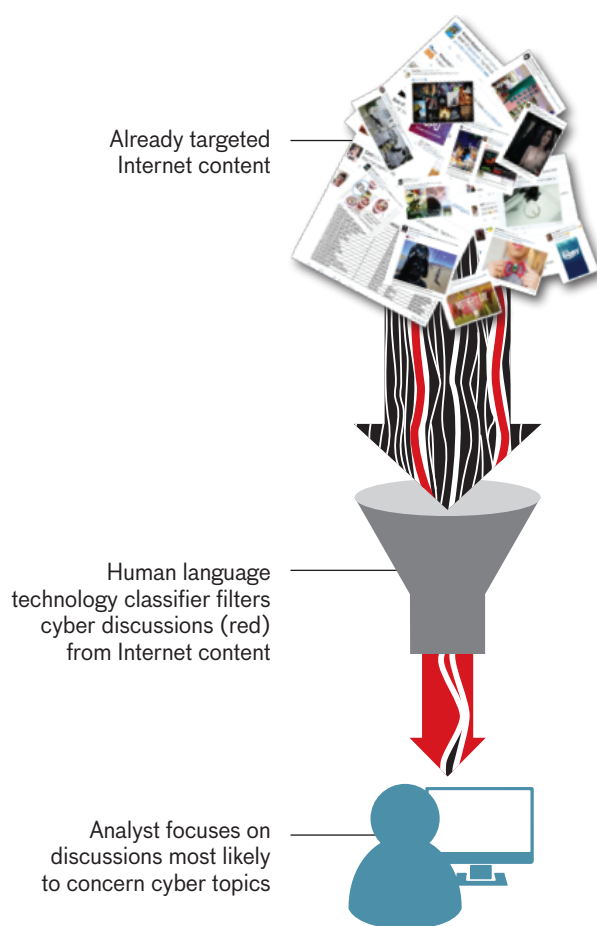
**FIGURE 1.** The four components pictured above must be present in any security process [1]. Anticipating an attack enhances the ability to deter, protect from, and detect new cyber attacks and makes it easier to recover from successful attacks.

to sift through vast amounts of data. Analysts try to discover and track cyber discussions by manual searches, often using metadata, such as thread or discussion topics, sources and destinations of social media discussions, and account names. This process is labor intensive, particularly when non-English cyber discussions must be manually translated, and sometimes ineffective because attackers can easily change metadata to hide malicious conversations by adopting innocuous-sounding names for Stack Exchange topics, Reddit threads, or Twitter hashtags. A more efficient and effective method is to supplement metadata analysis with direct mining of the discussion text via machine learning and human language technology (HLT) approaches. Such approaches can be applied to English and non-English content without requiring manual translation.

Although great bodies of published work focus on either HLT or cyber security, surprisingly few publications discuss the application of HLT to the cyber domain. The application appears to have been first proposed by Klavans in 2013 [3]. More recently, Lau et al. analyzed interactions between known cyber criminals on social media to distinguish between transactional interactions, in which cyber attack tools are bought or sold, and collaborative interactions, in which cyber criminals share tools or information without any monetary exchange [4]. However, their analysis requires manual extraction of cyber discussions before automated transaction analysis can be performed.

### An Automated Solution

Under the Cyber HLT Analysis, Reasoning, and Inference for Online Threats (CHARIOT) program, Lincoln Laboratory is developing HLT classifiers to automatically detect cyber discussions concerning attack methods, defense strategies, and tools' effectiveness through the examination of online forums. Our aim is to leverage



**FIGURE 2.** An automated process for extracting cyber discussions from online forums reduces the amount of time an analyst needs to spend on eliminating content that is irrelevant to his or her investigation.

available techniques, such as topic classification, entity recognition, and sentiment analysis (i.e., opinion mining), which have only begun to be applied to the problem of detecting and analyzing malicious cyber discussions.

### Concept of Operations

Among the large number of online discussions, few are on cyber topics. Our goal is to utilize modern HLT approaches to automatically filter out those cyber discussions for analysts (Figure 2).

We identified two concepts of operations (CONOPS) for using an HLT machine learning classifier to determine if a discussion concerns malicious cyber topics:

1. An analyst has already discovered Internet content, such as lists of topics in Reddit or lists of users in Twit-

ter, to examine. Instead of an analyst manually examining all discussions grouped under these topics or all tweets posted by these users, a classifier trained to determine whether a discussion/tweet was about cyber topics could identify which content an analyst should focus on first. This ranking is necessary because discussions may drift from topics of interest (malicious cyber topics) to topics that are not of interest (non-malicious cyber topics and noncyber topics) and vice versa, or they may move to users who do not discuss malicious cyber topics.

2. An analyst is trying to discover Internet forums (e.g., Stack Exchange communities) that contain cyber discussions of interest. This scenario is more difficult—the search is not focused on known forums and is thus wider. When exploring new Internet discussion areas, the classifier can rank the forums by their probability of containing cyber content, prioritizing discussions for an analyst’s investigation. For best performance, the classifier should be trained to find new discussions that are similar to past ones of interest.

**Classifier Development**

Before an HLT classifier can filter out cyber discussions, it must first be trained on cyber and noncyber discussions. In the sections below, we describe how training and testing were performed for our HLT classifiers. We also describe how data were gathered and labeled to support classifier development and how a previously developed keyword classifier was used as a reference for performance evaluations.

**Training**

The first training phase required to create an HLT classifier involves selecting both cyber and noncyber social media discussions to be fed into the classifier. To ensure that highly ranked discussions are actually the discussions of most interest to analysts, cyber examples used for training should be representative of those that were of most interest in the past. Training data should contain noncyber discussions that cover many topics and should capture words and phrases that distinguish cyber from noncyber content in many subjects to prepare the classifier for the diversity of content it will encounter once operational.

After an HLT classifier is trained, it can be fed input text from a discussion occurring on a social media network and provide as output the probability that the discussion is on a cyber topic (Figure 3). An output probability supports both CONOPS: conversations in forums of interest can be ranked by probability, and analysts can examine those with the highest probabilities first, or many new forums can be scanned to identify those with the greatest number of high-probability cyber conversations.

**Social Media Corpora**

Initially, we are training and testing our classifiers using three social media networks that analysts may monitor: Stack Exchange, Reddit, and Twitter (Table 1). Stack Exchange is a well-moderated question-and-answer network with communities dedicated to diverse topics. Answers can be quite comprehensive, long, and well written. Reddit is a minimally moderated set of forums

**Table 1. Characteristics of Social Media Posts**

SOCIAL MEDIA CORPUS	POST CHARACTERISTICS	EXAMPLE POST
Stack Exchange	Long, curated posts	“Every time I try even a simple stack smash on a 64bit machine, I run into issues. An address I am trying to write always contains null bytes.”
Reddit	Medium-length, not-well-curated posts	“What is a hack that you know that is awesome or mind blowing?”
Twitter	Short (140 characters), noncurated posts	“Cyber attack creates temporary disruption in Hawaii’s thirty-meter telescope website <a href="http://bit.ly/1OXOdce">http://bit.ly/1OXOdce</a> #cybersecurity #infosec”

with main topics called sub-Reddits and many individual threads or discussions under each topic. Twitter data consist of short tweets with at most 140 characters each. Tweets can be followed via usernames, hashtags that identify tweets on a similar topic, or Twitter lists (i.e., curated groups of Twitter users).

These three corpora were selected because they

- contain text with at least some cyber content;
- span a range of social media types; and
- offer a history of prior posts over a long time span.

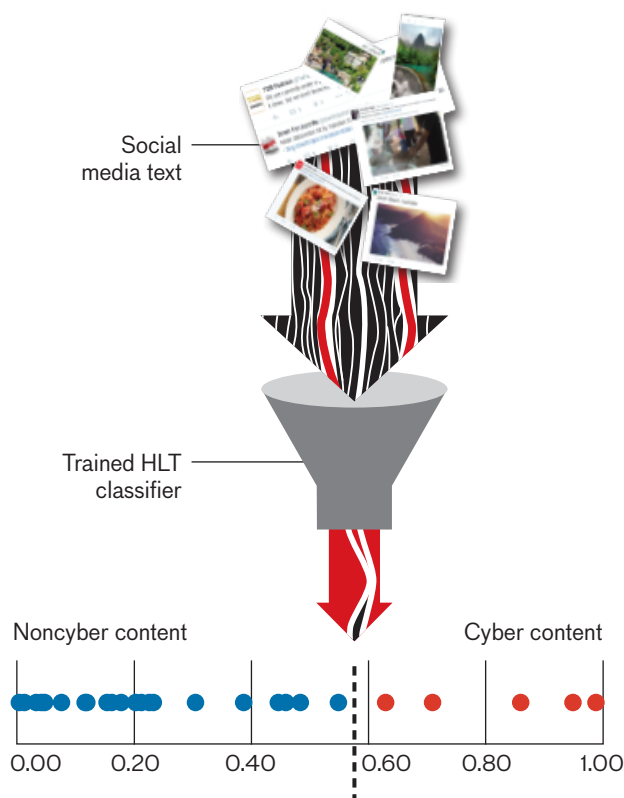
For each of these corpora, original posts and comments were gathered to generate cyber and noncyber “documents” to be fed into our classifiers for training and testing.

#### DOCUMENT LABELING

Documents refer to a collection of all posts concerning discussions on a specific question for Stack Exchange, all posts for a specific sub-Reddit thread in Reddit, and all collected tweets from a specific Twitter user. In practice, we required a Twitter document to have more than 20 tweets but less than 300 tweets to create a balanced set of training data, as Twitter users, particularly spammers, may have 1000s or 10,000s of tweets.

Preprocessing eliminated dates, thread titles, hashtags, usernames, and other metadata so that the classifier would be trained using only the discussion text (when a trained classifier is put into operational use, metadata may not be available to provide context for a discussion). Documents for Stack Exchange and Reddit were labeled with topic titles and tags set by the users of each corpus. All posts under cyber-related topics (e.g., *reverse engineering*, *security*, *malware*, *blackhat*) were labeled as cyber, and posts on other topics (e.g., *astronomy*, *electronics*, *beer*, *biology*, *music*, *movies*, *fitness*) were labeled as noncyber. For Stack Exchange, we further restricted cyber discussions to posts with lower-level tags (e.g., *penetration test*, *buffer overflow*, *denial of service*, *Heartbleed*<sup>1</sup>). For Twitter, tweets from 127 users identified as cyber experts by Lincoln Laboratory researchers were labeled cyber, while tweets from 500 other randomly selected users were labeled noncyber. Table 2 shows for each corpus the number of cyber and noncyber topics, the number of documents, the

<sup>1</sup>Made public in April 2014, Heartbleed is a vulnerability in the OpenSSL cryptography library that allowed attackers to steal servers’ private keys and users’ passwords.



**FIGURE 3.** Text from a social media discussion is fed into a trained human language technology (HLT) classifier. The classifier then outputs the probability that the discussion is about cyber topics. This probability ranges from zero (not about cyber) to one (almost certainly about cyber). Output probabilities for different discussions are shown above, with a cyber content threshold (dashed line) that may be manually set by an analyst. An analyst would examine all discussions with probabilities above the threshold (red dots) and ignore remaining discussions with probabilities below the threshold (blue dots).

median number of words in each document, the time period covered by the collection, and a summary of how documents were labeled as cyber or noncyber.

#### Reference Keyword Detector

To compare the performance of our classifier with that of previously used classifiers, we implemented a tool that detects cyber discussions via keywords and phrases. It searches for 200 cyber keywords and phrases in a document, counts the number of occurrences, and normalizes the count by dividing the total number of occurrences by the total number of words in the document. Higher counts indicate documents that are more likely about

cyber topics. Cyber discussion keywords (e.g., *rootkit*, *infected*, *checksum*) and phrases (e.g., *buffer overflow*, *privilege escalation*, *distributed denial of service*) had been selected by trained linguists.

**Processing and Classification**

As shown in Figure 4, the classification pipeline requires preprocessing each document, generating features (term frequency-inverse document frequency [TF-IDF] ratios) for each word in each document, and training a classifier to distinguish between cyber and noncyber documents on the basis of these generated features. The preprocessing step employs stemming<sup>2</sup> to normalize word endings and text normalization techniques, such as the removal of words containing numbers and the replacement of URLs with a token indicating a URL was used, to ensure that the feature inputs are standardized. The TF-IDF ratios were created by counting the number of occurrences of words in documents and normalizing these counts by using the number of documents in which the words occur. In our research, and in the HLT community’s research in general, TF-IDF ratios have provided good performance when used in text classification. Our experiments used the TF-IDF ratios of unigrams (individual words) to create features. To classify the documents on the basis of these features, logistic regression and linear support vector machine classifiers were used; both classifiers train rap-

idly, require little computation to analyze a document, and provide an output score proportional to the probability that the input document contains cyber content.

**Initial Results**

Figure 5 shows initial results for classifiers trained and tested on Stack Exchange, Twitter, and Reddit data. Each classifier outputs the probability that each document discusses cyber topics; this probability is based on a set threshold (the minimum probability required for the classifier to label a document as cyber). The document labels then make it possible to determine the number of false alarms (i.e., noncyber documents that are classified as cyber) and misses (i.e., cyber documents that are classified as noncyber). We present our results in the form of detection error tradeoff (DET) curves that show how false-alarm and miss probabilities vary as the threshold on the classifier’s output probability varies as plotted on normal deviate scales [5]. Our goal is to provide good detection of cyber documents (e.g., a low miss rate) and limit the number of noncyber documents that are labeled as cyber (i.e., a low false-alarm rate). As shown by the gray box in Figure 5, a false-alarm rate below 1% and a miss rate below 10% is the performance target. Within this target range, our pipeline provides good filtering of Internet content as long as the portion of cyber documents relative to all documents presented to a classifier is 5% or greater.

The curves shown in Figure 5 indicate that the classifiers we developed for each social media corpus do meet the performance target—they miss less than 10% of cyber-la-

<sup>2</sup>Stemming is the reduction of a word to its root form, e.g., stemming “hacks” or “hacked” produces “hack.”

**Table 2. Social Media Corpora Document Labeling**

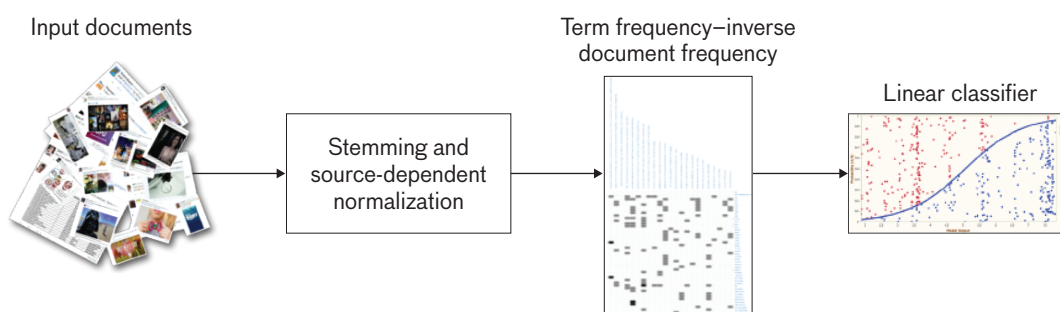
CORPUS	TOPICS		DOCUMENTS		TIME COVERED	DOCUMENT LABELING METHOD
	CYBER	NONCYBER	NUMBER OF DOCUMENTS	MEDIAN NUMBER OF WORDS		
Stack Exchange	5	10	~200K	245	Years	Cyber-related topics and tags
Reddit	10	51	~59K	152	Months	Cyber-related sub-Reddits
Twitter	127	500	627	546	Months	Expert cyber users’ tweets

beled documents and classify less than 1% of the noncyber-labeled documents as cyber. Before obtaining these results, we first had to understand the minimum number of words in each document, amount of training data, and types of preprocessing necessary to provide good performance.

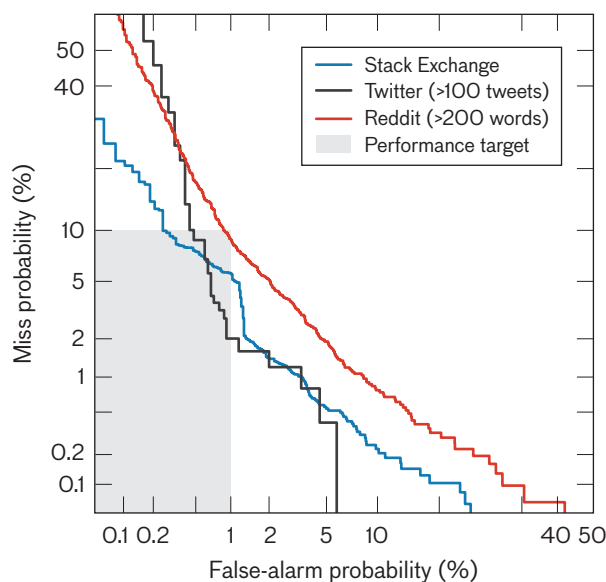
### Comparative Analysis of Classifiers

Figure 6 compares the performance of the baseline keyword classifier to the logistic regression classifier on Stack Exchange data. The logistic regression classifier (blue

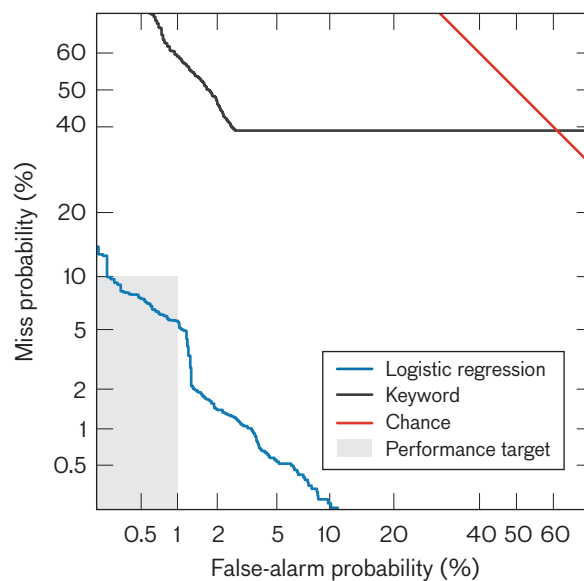
curve) passes through the performance target region, meaning it misses less than 10% of cyber documents with a false-alarm rate of less than 1%. The baseline keyword system (black curve) performs substantially worse than the logistic regression classifier. At a false-alarm probability of 10%, the system fails to detect roughly 40% of the cyber documents; at a false-alarm probability of 1%, the miss probability is roughly 60%. To determine the cause of this poor performance, we examined the Stack Exchange documents that corresponded with the false



**FIGURE 4.** The flow of documents through the classification pipeline requires preprocessing to ensure the text is ready to use in feature generation, calculation of term frequency-inverse document frequency ratios for each word in the document, and classifier training using the features generated for each document.



**FIGURE 5.** Initial detection error tradeoff results indicate that the classifiers perform well for all three social media corpora; all curves overlap with the performance target region (gray box).



**FIGURE 6.** The detection error tradeoff curves for Stack Exchange documents show that the logistic regression classifier significantly outperforms both the baseline keyword system and chance guessing.

**Table 3. List of Most Important Cyber and Noncyber Words Used by Our Logistic Regression Classifier Trained on Stack Exchange Data**

TOP 50 CYBER WORDS	TOP 50 NONCYBER WORDS
HTTP, SQL, Secur, URL, Window, access, address, app, application, attack, authenticate, browser, bug, certificate, client, code, crack, detect, encrypt, execute, exploit, file, firewall, hash, infect, inject, install, key, malicious, malware, network, obfuscate, overflow, packet, password, payload, request, risk, scan, script, secure, server, site, test, tool, traffic, user, virus, vulnerability, web	Arduino, Christian, God, LED, The, and, bank, board, buy, cell, chip, chord, circuit, clock, credit, current, datasheet, design, electron, film, frac, frequency, fund, graph, hi, invest, microcontroller, motor, movie, music, note, output, part, pin, play, power, rate, resistor, serial, signal, simulate, state, stock, tax, the, time, tree, two, voltage, wire

alarms. We found that false alarms were often caused by one or more occurrences of cyber keywords in documents with topics unrelated to cyber. For example, the keyword *infected* appeared in documents referring to bacterial infection. Similarly, the keyword *checksum* appeared in many documents on technical topics. Simply counting occurrences of keywords without considering the context of the documents led to the false alarms. Worst-case performance, shown by the chance-guessing curve (red), is obtained by randomly assigning a label to each document.

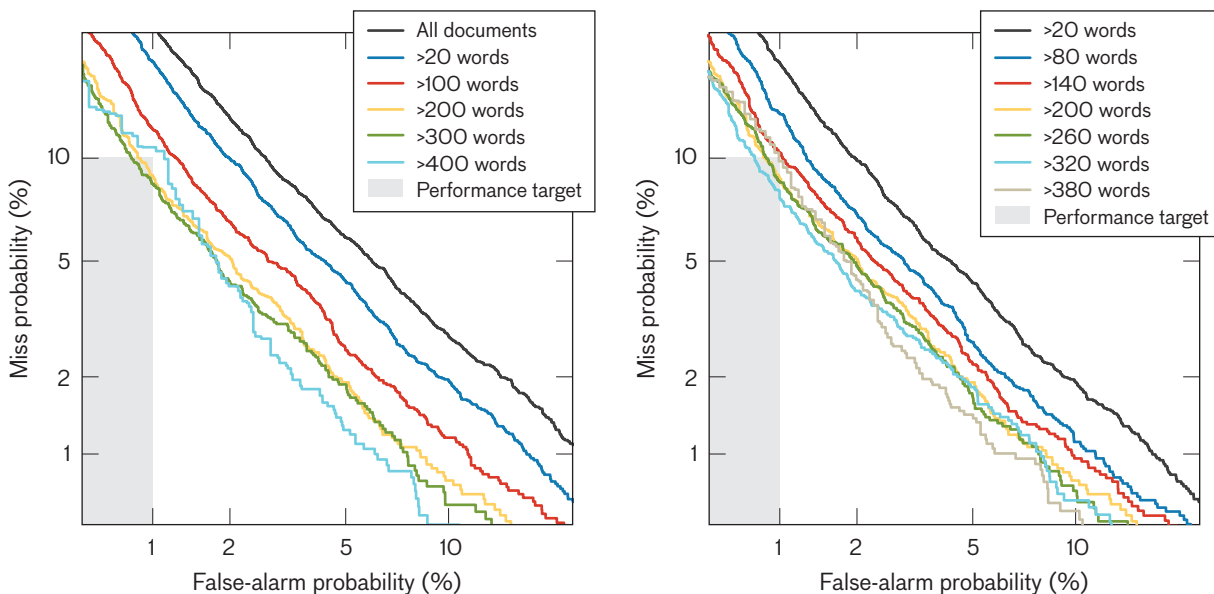
Table 3 provides some insight into why our logistic regression classifier performs better than the keyword system. On the left are the 50 words that receive the highest positive weights (i.e., the words that are most useful to our classifier in identifying cyber documents) and thus contribute more than other words to causing a document to be classified as cyber. These words span a wide range of cyber discussions on several topics. Many of these words and other positively weighted cyber words used by this classifier are highly likely to be present in cyber documents. While there is some vocabulary drift with time, experiments suggest that most terms remain stable for up to one year (see section titled “Stability in Performance over Time”). Unlike the keyword system, our classifier strongly indicates cyber only if many of the 50 cyber words are combined in one document. Multiple instances of one word will not yield a strong cyber indication. The right side of this table lists the 50 words that receive the highest magnitude negative weights (i.e., the words that are most useful to our classifier in identifying noncyber documents) and thus contribute more than others to causing a document to be classified as noncyber. These words indicate the breadth of topics that

noncyber documents cover. This diversity suggests that a large set of noncyber documents needs to be fed into the logistic regression classifier during training.

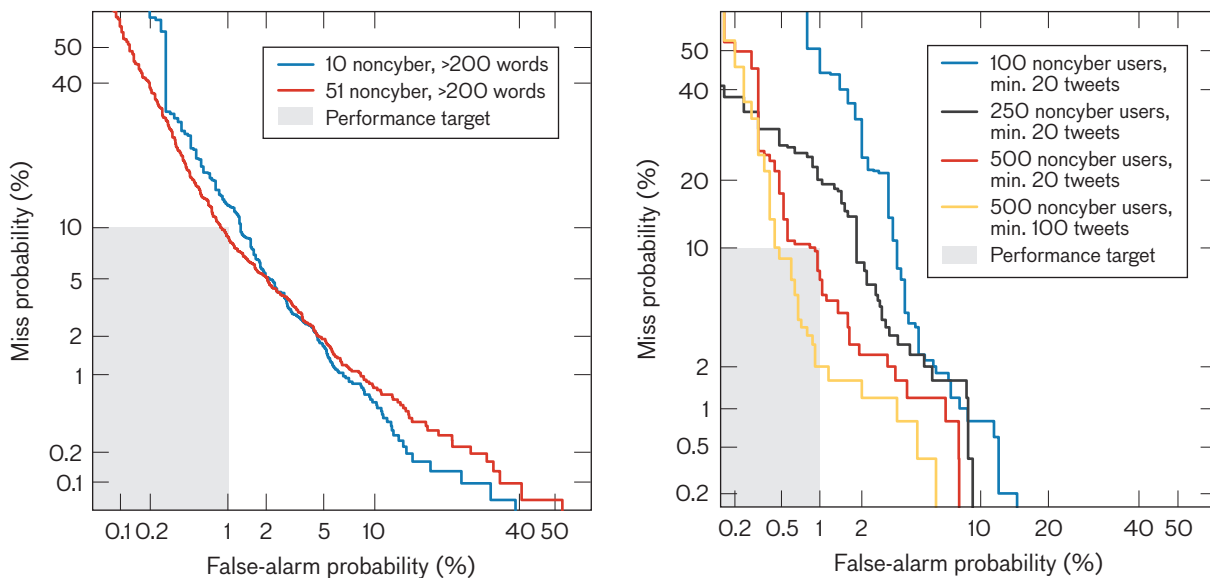
**The Effect of Document Length and Amount of Training Data**

The DET curves in Figure 7 show how our classifier’s performance depends on the number of words in a Reddit document. For comparison purposes, the left plot shows how poorly the classifier performs when all documents (no minimum word count, many short threads with no responses) are included (black curve). The right plot shows the classifier performance with minimum word counts in smaller increments, allowing a better view of the performance improvements. As seen in both plots, performance initially increases rapidly as the number of words increases. However, the rate of performance increase slows as the minimum number of words increases, and classifier performance enters the target range when the minimum number of words is above 200. Our results thus suggest that 200 or more words in an Internet conversation are required to provide accurate classification of cyber and noncyber documents. To examine the effect of the amount of noncyber Reddit data on performance, the number of noncyber topics was increased from 10 to 51 (Figure 8). A small performance improvement is seen for this increase in the number of noncyber topics.

Classifier performance also improves for Twitter as the number of words per document and the amount of noncyber training data are increased while the number of cyber users (127) remains constant (Figure 9). For Twitter, a document is composed of all the tweets from



**FIGURE 7.** As the minimum number of words in each Reddit document is increased, the classifier’s performance improves.



**FIGURE 8.** Increasing the number of noncyber sub-Reddit topics from 10 to 51 reduces the gap to the performance target. Both experimental runs were performed with a minimum word count of 200. Note that the performance increases by approximately 3% in the desired false-alarm range.

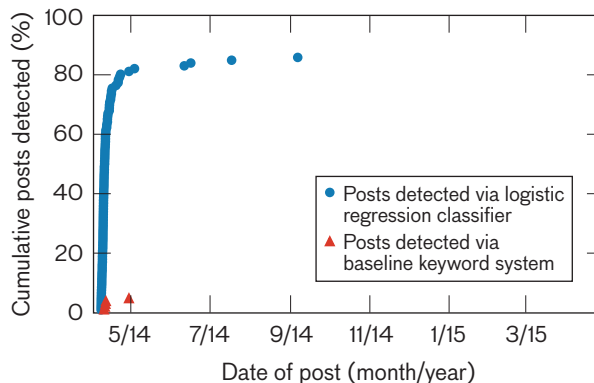
**FIGURE 9.** As the number of noncyber Twitter users and words per document (i.e., tweets per user) is increased, performance improves. For example, at 1% false alarms, the miss rate is 40% with 100 noncyber users (blue curve), 20% with 250 users (black curve), and only 6% with 500 users (red curve). By adding additional tweets from the 500 users, the miss rate is reduced to 2% at 1% false alarms (yellow curve).

a single user, so the number of words per document is increased by including more tweets per user. The number of noncyber training documents is increased by randomly sampling users and collecting their tweets in additional documents. Because we assume that there is a very low probability of a randomly sampled user discussing cyber topics, no extra labeling or cost is incurred by incorporating additional training data. On average, there are 10 words per tweet after preprocessing, so in each of the results with a minimum of 20 tweets, there are 200 words per document. Performance was further improved by collecting additional tweets and increasing the average number of words per document to 1000. These results are consistent with the Reddit results showing improved classifier performance as more words are added to the documents and with the Reddit and Stack Exchange results showing improved classifier performance as more noncyber training data are provided.

#### Stability in Performance over Time

Another test of our logistic regression approach determined whether a classifier trained before the Heartbleed vulnerability was made public could detect social media discussions concerning Heartbleed. Such discussions could only be found if they included words that were used in prior social network cyber discussions because the classifier would have never seen the word *Heartbleed*. Figure 10 plots the cumulative percentage of Stack Exchange threads detected by a logistic regression classifier trained on 3924 cyber and 7848 noncyber documents posted before the Heartbleed attack was announced on 8 April 2014. The classifier immediately detects the flurry of posts on 8 April and in the following days. Of the 106 *Heartbleed*-tagged threads, 86% were detected and only 14% were missed at a false-alarm rate of 1%. Our logistic regression classifier performed much better than the keyword baseline system, which only detected 5% of the Heartbleed discussions, because ours detects words related to the protocols affected by Heartbleed (e.g., *SSL*, *TLS*) and other words associated with cyber vulnerabilities (e.g., *malware*, *overflow*, *attack*). Because the keyword system lacked such keywords used in Heartbleed discussions, it suffered from a high miss rate.

A system to detect cyber documents is most useful if it does not require frequent retraining to match possible changes in cyber vocabulary over time. We performed



**FIGURE 10.** Our logistic regression classifier, which was trained on data before the Heartbleed vulnerability was known, was still able to detect 86% of Stack Exchange posts discussing Heartbleed (blue dots). By comparison, the baseline keyword system only detected 5% of posts discussing Heartbleed (red triangles).

experiments in which a classifier was trained on Stack Exchange data up to a given date and then tested every month after that date without retraining. Figure 11 plots the miss percentage (averaged over false-alarm rates ranging from 0.25% to 1.0%) for a classifier that was trained on data before June 2012 and then tested each month for a year on new data appearing within each respective month. The results indicate that the miss rate increases little over the year and is always below roughly 20%. The experiment was repeated over multiple time periods from 2012 through 2014, producing similar results each time. Classifiers thus do not require frequent retraining—once a year or at most every six months is adequate.

#### Filtering and Concentrating Cyber Documents

One of our goals with the cyber classifiers we are developing is to have them filter or concentrate documents from social media sources so an analyst is presented mainly with cyber documents. We assume that our classifiers will be applied to preselected Internet data that are known to have more than 1% cyber documents and that a 90% detection rate for cyber documents is sufficient to discover important long-standing cyber discussions. As previously discussed, the target performance we have been using as a reference is a miss percentage below 10% for a false-alarm percentage below 1%. Figure 12 shows the filtering or concentration effectiveness of our classifiers with performance in this target range when the classifiers are



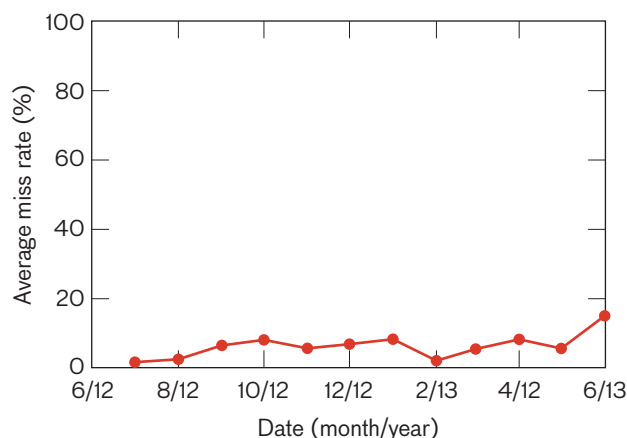
applied to Internet sources with different initial concentrations of cyber documents. The vertical axis in this figure is the fraction of cyber documents remaining after filtering the documents; the horizontal axis is the fraction of cyber documents in the Internet source. The upper curve (red) is for a classifier that misses 10% of the cyber documents with 0.25% false alarms, and the lower curve (blue) is for a classifier that misses 10% of the cyber documents with 1% false alarms. If only 1 in 100 of the Internet documents examined are cyber (1% on the horizontal axis), then our classifiers that provide performance between these curves present between 50% (1 in 2) and 80% (4 in 5) cyber documents to an analyst. This ability to enrich output of cyber documents is a large improvement in concentration over the existing keyword classifier, which presents 30% (3 in 10) cyber documents to an analyst at a 1% false-alarm rate. If the fraction of cyber documents increases to only 5% (1 in 20), our classifiers present between 83% (5 in 6) and 95% (19 in 20) cyber documents to an analyst. These results motivate the performance target we are reaching with our classifiers and suggest that our classifiers are useful even if there is only 1 cyber document in each 100 documents from an Internet source.

## Related Work

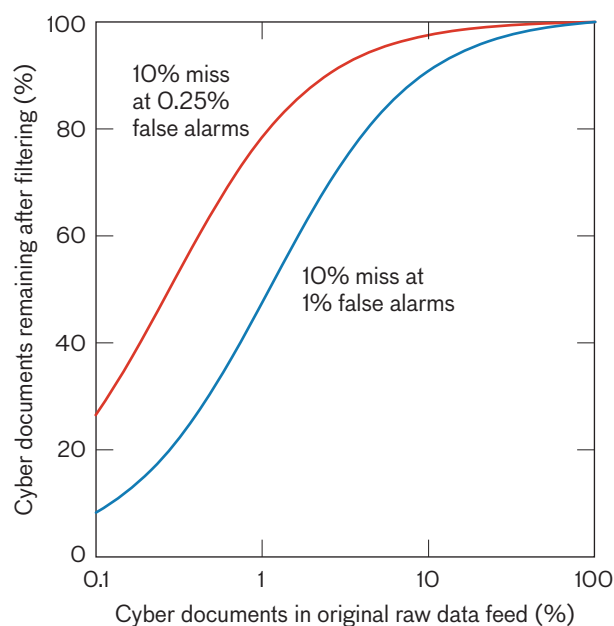
### Relational Classification Methods

Up to this point, we have focused on extracting the language content within social media posts to perform classification. Certain social media networks, such as Twitter, include rich metadata (e.g., user, content, messaging information) that can be leveraged to build a social network of entities describing the relations and activities between these entities [6]. Entity types may include groups, individuals, and even hashtags. Because of homophily (“birds of a feather flock together”), we expect that finding one cyber user on Twitter will lead to finding other cyber users who follow or retweet each other. Homophily is part of a more sophisticated set of relational classification methods [7] that combine social network metadata and machine learning techniques to establish connections and interactions among users and content on the network.

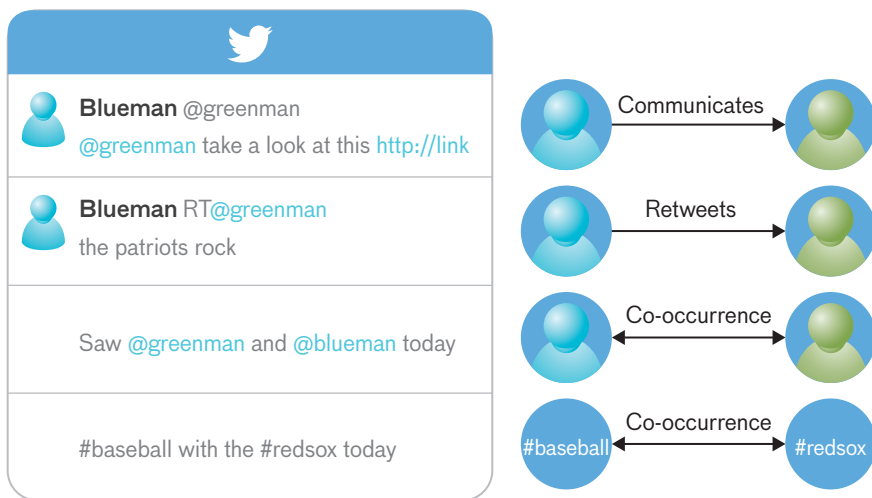
The steps for relational cyber classification are as follows: First, text and metadata of a single message are processed to produce entities and the relations between them [6]. For example, a tweet by @cyberuser, such as



**FIGURE 11.** A logistic regression classifier was trained on Stack Exchange data before June 2012 and then tested every month after that for a year on new data. Despite the classifier not being retrained, its miss percentage increased little over the year and stayed below 20%.



**FIGURE 12.** The two performance curves for our classification pipeline show the percentage of cyber documents that would be presented to an analyst after classification (y-axis) compared to the percentage of cyber documents in the input documents before classification (x-axis).



**FIGURE 13.** A Twitter graph is constructed by using multiple edge types (communications between users, retweets, co-occurrence of users, and co-occurrence of hashtags) and two types of nodes: users (@greenman, @blueman) and hashtags (#baseball, #redsox).

“@cyber01 Look at this #malware exploit,” shows a relation between the two Twitter users, @cyberuser and @cyber01. It also shows a relation between the two users and the hashtag #malware. Second, the entities and relations are combined in a database that stores graphs and optimizes graph operations (i.e., a graph database), such as finding all the neighbors of a node (an entity). Computed graph features, such as the number of nodes connected to a given fixed node, can be added to the graph along with attributes on relations and entities (e.g., full names, email addresses). The final step is to apply relational learning to the problem of classifying entities as cyber/noncyber, a process that consists of finding relational features for both entities and related entities; then, labels of nodes representing known cyber users and homophily are used to boost performance of classifying nodes as cyber or noncyber. This relational learning technique is referred to as collective classification or semisupervised learning in the literature [8–10].

**INFORMATION EXTRACTION AND GRAPH CONSTRUCTION**

The first two steps, information extraction and graph construction, are performed by using multitype nodes and edges (relations between entities). Figure 13 shows the basic process, with four different types of relations and two types of entities being used to construct a Twitter graph.

Relations and entities capture a significant amount of the activity on Twitter. Applying the method described in Figure 13 on 10% of the tweets posted for a typical

month on Twitter in 2014 yields a graph with the following characteristics:

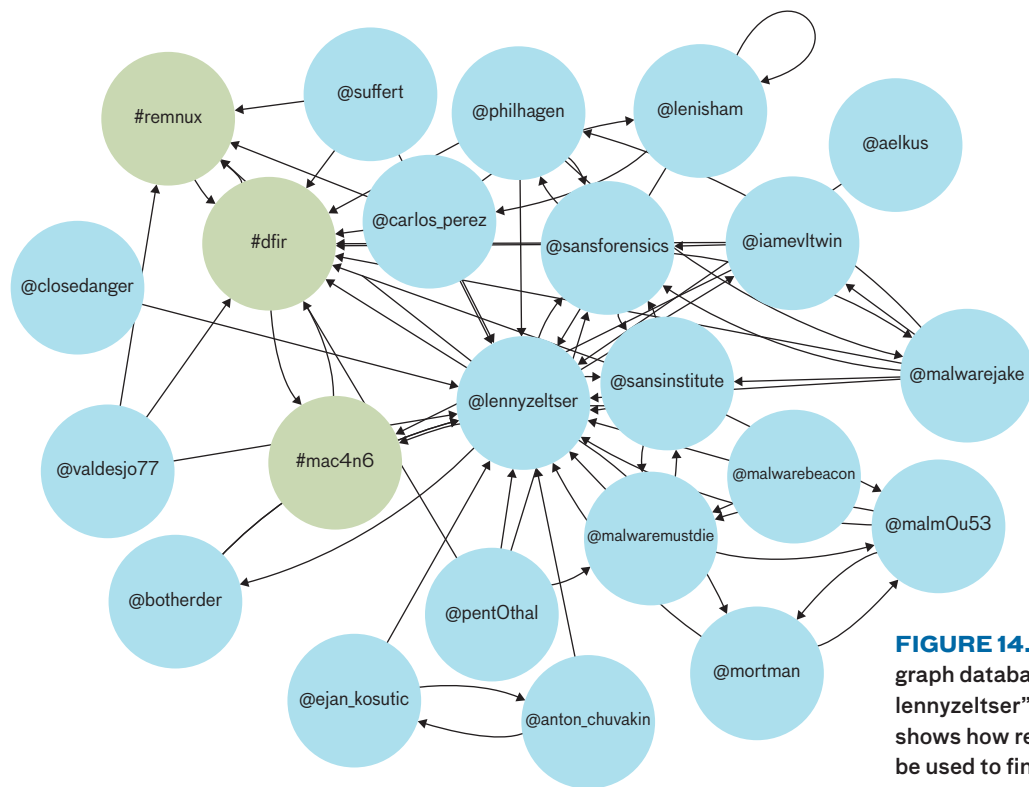
- 52.3 million nodes (6.7 million hashtags and 45.6 million users)
- 361.7 million edges

This large graph can be stored in a graph database (e.g., Neo4j) and explored using graph queries. A typical example of querying for the user “@lennyzeltser” and all of his neighbors in the graph is

```
match (n:user {name:'@lennyzeltser'})-[r:rel]-(m)
return n,r,m;
```

This query yields the result shown in Figure 14. In the center of the graph is the user we queried. Hashtag neighbors (green circles) are #mac4n6 (Mac Forensics), #dfir (Digital Forensics and Incident Response Summit), and #remnux (A Linux Toolkit for Reverse Engineering and Analyzing Malware)—all cyber forensics-related hashtags. Many of the user neighbors (blue circles) are also cyber related (e.g., @malwaremustdie, @malware-jake, @sansforensics), but some are more generally named, for example, @closedanger. This network of neighbors of @lennyzeltser shows the power of relational homophily—neighbors of a cyber user have a strong tendency to also be cyber users.

After constructing the Twitter graph, we can then utilize relational methods for classification. A standard baseline for relational classification is collective inference [8], which uses the cyber/noncyber probability of a



**FIGURE 14.** An example query in the graph database Neo4j of the user “@lennyzeltser” and of all his neighbors shows how relational homophily can be used to find other cyber users.

user node and that of its neighbors to iteratively estimate the probability of a user being cyber or not cyber. Thus, collective inference is a natural algorithmic implementation of relational homophily in social networks. Some well-known methods for collective inference are relaxation iteration, Gibbs sampling, iterative classification, and relational dependency networks [8, 11]. Exploring these methods will be an area of future experimentation at Lincoln Laboratory.

### Future Work

Our results demonstrate that

- our HLT classifiers performed well for all corpora;
- roughly 200 words in a discussion provide good detection of cyber conversations;
- a classifier trained before the major Heartbleed vulnerability was announced could accurately detect discussions relating to this vulnerability; and
- performance of a classifier is maintained even when tested on discussions occurring six months to a year after it was trained.

However, preliminary experiments suggest that performance degrades when a classifier is trained on one corpus (e.g., Reddit) and tested on another (e.g., Stack Exchange). We are currently exploring three approaches to improve cross-domain performance: (1) constructing a generative probabilistic model of cyber documents that can be used to determine if a new document has a high probability of being cyber without referencing noncyber data; (2) using neural network word embeddings to take advantage of the syntactic and semantic relationships between words; and (3) using features derived from graph analysis of social networks. Feature selection, phrase selection, n-gram analysis (i.e., considering words that occur together in documents), and cross-domain training and adaptation will also be further explored.

We have also begun collecting non-English social media content to test our approaches with other languages. Future relational-learning experiments using social network structure to perform cyber classification are expected to yield information that should be useful to

improve content-based methods of classification (such as the TF-IDF and logistic regression methods discussed in this article). Analysts could leverage relational learning to explore the neighbors of a user in a prioritized manner, investigating closely related users, organizations, events, and topics. Follow-on work also includes efforts to automatically extract entities and relationships and to model cyber threats. This automated extraction and modeling will enable us to categorize documents according to the “Diamond Model” of intrusion analysis (so named for how the model organizes the basic aspects of malicious activity in the shape of a diamond) to assess the capabilities, available infrastructure, and victims of cyber adversaries so we can understand how to observe, understand, and defend against them [12].

### Acknowledgments

We are grateful to Julie Dhanraj, Beth Richards, Jason Duncan, Peter Laird, and their subject-matter experts for their support. Vineet Mehta’s initial contributions to the Cyber HLT Analysis, Reasoning, and Inference for Online Threats (CHARIOT) program are appreciatively acknowledged. Thank you to Clifford Weinstein, Kevin Carter, and Kara Greenfield for initiating the program and providing helpful insights. ■

### References

1. U.S. Executive Office of the President, National Science and Technology Council, Federal Cybersecurity Research and Development Strategic Plan, 5 Feb. 2016, available at [https://www.whitehouse.gov/sites/whitehouse.gov/files/documents/2016\\_Federal\\_Cybersecurity\\_Research\\_and\\_Development\\_Strategic\\_Plan.pdf](https://www.whitehouse.gov/sites/whitehouse.gov/files/documents/2016_Federal_Cybersecurity_Research_and_Development_Strategic_Plan.pdf).
2. J. Chartaff, “Announced Cyber Attack on Israel Fizzled,” *Homeland Security Today*, 15 April 2014, available at <http://www.hstoday.us/briefings/daily-news-analysis/single-article/announced-cyber-attack-on-israel-fizzled/a34d782076af7d-0053ded523054619c9.html>.
3. J.L. Klavans, “Cybersecurity—What’s Language Got to Do with It?” Technical Report, University of Maryland Language and Media Processing (LAMP) Laboratory, Institute for Advanced Computer Studies (UMIACS), UMIACS-LAMP-TR-158, 2013, available at <http://lampsv02.umiacs.umd.edu/pubs/LampTRs.php>.
4. R. Lau, Y. Xia, and Y. Ye, “A Probabilistic Generative Model for Mining Cybercriminal Networks from Online Social Media,” *IEEE Computational Intelligence Magazine*, vol. 9, no. 1, 2014, pp. 31–43.
5. A.F. Martin, G.R. Doddington, T.M. Kamm, M.L. Ordowski, and M.A. Przybocki, “The DET Curve in Assessment on

Detection Task Performance,” *Proceedings of Eurospeech*, vol. 4, 1997, pp. 1899–1903.

6. W.M. Campbell, E. Baseman, and K. Greenfield, “Content + Context Networks for User Classification in Twitter,” Neural Information Processing Systems (NIPS) 2014 Workshop, available at [snap.stanford.edu/networks2013/papers/net-nips2013\\_submission\\_3.pdf](http://snap.stanford.edu/networks2013/papers/net-nips2013_submission_3.pdf).
7. L. Getoor and B. Taskar, eds. *Introduction to Statistical Relational Learning*. Cambridge, Mass.: MIT Press, 2007.
8. S.A. Macskassy and F. Provost, “Classification in Networked Data: A Toolkit and a Univariate Case Study,” *Journal of Machine Learning Research*, vol. 8, 2007, pp. 935–983.
9. M. Szummer and T. Jaakkola, “Partially Labeled Classification with Markov Random Walks,” in *Advances in Neural Information Processing Systems 14*, T.G. Dietterich, S. Becker, and Z. Ghahramani, eds. Cambridge, Mass.: MIT Press, 2001.
10. X. Zhu, “Semi-Supervised Learning Literature Survey,” University of Wisconsin–Madison, Computer Sciences Technical Report 1530, 2007.
11. J. Neville and D. Jensen, “Collective Classification with Relational Dependency Networks,” *Proceedings of the 2nd International Workshop on Multi-Relational Data Mining*, 2003, pp. 77–91.
12. S. Caltagirone, A. Pendergast, and C. Betz, “Diamond Model of Intrusion Analysis,” Center for Cyber Threat Intelligence and Threat Research, Hanover, Md., Technical Report ADA586960, 2013, available at [http://www.threatconnect.com/files/uploaded\\_files/The\\_Diamond\\_Model\\_of\\_Intrusion\\_Analysis.pdf](http://www.threatconnect.com/files/uploaded_files/The_Diamond_Model_of_Intrusion_Analysis.pdf).

### About the Authors



**Richard P. Lippmann** is a Lincoln Laboratory Fellow working in the Cyber Analytics and Decision Systems Group. He joined Lincoln Laboratory in 2001. His research interests include aids for the hearing impaired, speech recognition, pattern classification, neural networks, and cyber security. He has taught three courses on machine learning; has been an IEEE Distinguished Lecturer; won the first *IEEE Signal Processing Magazine* Best Paper Award for an early article on neural networks; and has authored or coauthored more than 100 papers on topics including speech recognition, machine learning, and cyber security. He served as the program chair of the Research in Attacks, Intrusions, and Defenses Workshop; the Neural Information Processing Systems (NIPS) annual conference; and the NIPS Workshop on Machine Learning in Adversarial Environments. He has participated in four national-level government studies on cyber security. He received a bachelor’s degree in electrical engineering from the Polytechnic Institute of Brooklyn in 1970 and a doctoral degree in electrical engineering from MIT in 1978.



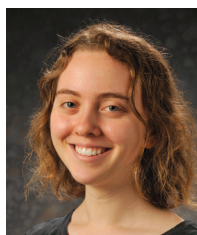
**William M. Campbell** is a senior technical staff member of the Laboratory's Human Language Technology Group. He provides leadership and technical contributions in the areas of speech processing, machine learning, and social networks. His speech processing work has resulted in advances in speaker and language recogni-

tion, including the development of algorithms that have been widely cited in published papers and operationally implemented. He has made numerous contributions in social network graph analysis as it relates to simulation of social networks, machine learning involving social networks, and construction of networks from multimedia content. Prior to joining Lincoln Laboratory in 2002, he worked on speech processing and communication systems at Motorola. An active contributor to the speech and machine learning research community, he has served as a reviewer and scientific committee member for several conferences: IEEE Odyssey: The Speaker and Language Recognition Workshop; Annual Conference on Neural Information Processing Systems; International Conference on Acoustics, Speech, and Signal Processing; INTERSPEECH; and IEEE Spoken Language Technology Workshop. He is the author of more than 100 peer-reviewed papers, including multiple book chapters; a recipient of the Motorola Distinguished Innovator Award; the holder of 14 patents; and a senior member of the IEEE. He received three bachelor's degrees—in computer science, electrical engineering, and mathematics—from South Dakota School of Mines and Technology in 1990 and master's and doctoral degrees in applied mathematics from Cornell University in 1995.



**David J. Weller-Fahy** is a technical staff member in the Cyber Analytics and Decision Systems Group and a principal investigator on the CHARIOT program. His research interests include machine learning applications to language problems, programming languages, computer-to-computer communications

protocols, and computer and network security. He joined Lincoln Laboratory in 2014. He holds a bachelor's degree in computer science from the University of Illinois Springfield and a master's degree in cyber operations from the Air Force Institute of Technology. For exceptional academic achievement and outstanding research during his graduate studies, he received the Secretary James G. Roche Award.



**Alyssa C. Mensch** is a technical staff member in the Human Language Technology Group. Since joining Lincoln Laboratory in 2015, she has focused on natural language processing for cyber applications. She received a bachelor's degree in mathematics with computer science from MIT and a master's degree in computer

science from the University of Pennsylvania.



**Giselle M. Zeno** is a doctoral student in the Department of Computer Science at Purdue University. She contributed to the CHARIOT program as a 2015 Lincoln Laboratory Summer Research Program participant in the Human Language Technology Group, applying relational machine learning methods to the cyber domain. At

Purdue, she works with Professor Jennifer Neville in the Network Learning and Discovery Lab, studying machine learning in relational domains. She has a bachelor's degree in computer science from the University of Puerto Rico at Bayamón. As a graduate student, she has received two fellowships: one from the National Graduate Degrees for Minorities in Engineering and Science (GEM) Consortium and the other from Purdue University.



**Joseph P. Campbell** is the associate leader of Lincoln Laboratory's Human Language Technology Group, where he directs the group's research in speech, speaker, language, and dialect recognition; word and topic spotting; speech and audio enhancement; speech coding; text processing; natural language processing;

machine translation of speech and text; information retrieval; extraction of entities, links, and events; cross-language information retrieval; multimedia recognition techniques, including both voice and face recognition for biometrics applications; advanced analytics for analyzing social networks on the basis of speech, text, video, and network communications and activities; and recommender systems. He specializes in the following for government applications: research, development, evaluation, and transfer of speaker recognition technologies; design of speaker recognition and biometrics evaluations; design of corpora to support those evaluations; and development and evaluation of biometrics technologies and systems. He joined Lincoln Laboratory in 2001 as a senior staff member after serving 22 years at the National Security Agency. He was an IEEE Distinguished Lecturer and is an IEEE Fellow. He earned bachelor's, master's, and doctoral degrees in electrical engineering from Rensselaer Polytechnic Institute in 1979, The Johns Hopkins University in 1986, and Oklahoma State University in 1992, respectively.

# Cloudbreak: Answering the Challenges of Cyber Command and Control

Diane Staheli, Vincent F. Mancuso, Matthew J. Leahy, and Martine M. Kalke

Lincoln Laboratory's flexible, user-centered framework for the development of command-and-control systems allows the rapid prototyping of new system capabilities. This methodology, Cloudbreak, effectively supports the insertion of new capabilities into existing systems and fosters user acceptance of new tools.



As the number and size of networks maintained by the Department of Defense (DoD) continue to grow, concerns about the complexity of providing cyber security for these networks have mounted. In 2012, then Secretary of Defense Leon Panetta established the Joint Cyber Centers (JCC) at U.S. geographic combatant commands (COCOMs) to coordinate cyber activities within each command's area of responsibility (AoR) and to apprise combatant commanders of the impacts of the cyber landscape to their missions [1]. The JCCs were instituted to resolve the lack of coordinated cyber security within and across all the COCOMs.

The JCCs charted their own paths for defining the structure of their organizations, determining their work processes, and procuring the tools and capabilities necessary to accomplish their missions. To help address the COCOMs' capability needs and improve upon their model for technology delivery, leadership at the JCCs turned to Lincoln Laboratory's Cloudbreak<sup>1</sup> initiative, which had been sponsored by the Assistant Secretary of Defense for Research and Engineering. During its four-year tenure, the Cloudbreak program successfully filled critical gaps in COCOMs' cyber situational awareness by utilizing an iterative user-centered design process to rapidly deploy cyber capabilities to the warfighter.

The Cloudbreak process is designed to address near-term capability gaps once for all COCOMs rather than once for each COCOM. The overall goal of Cloudbreak

---

<sup>1</sup>The name Cloudbreak was selected as the next in a series of program names inspired by weather terms; it does not imply a connection to cloud computing.

is to rapidly deliver technologies to confront emerging and unanticipated threats. By allowing operators to drive technology development rather than giving them predefined solutions, the Cloudbreak approach aims to provide agile, interoperable, and reusable applications. This article describes Cloudbreak's genesis and its successful technology development and insertion process. Case studies demonstrate how the Cloudbreak process was applied to the implementation of two cyber security tools: the Cyber Analytical Station and Cyber Dashboard.

### **Cyber Challenges for Combatant Commands**

The COCOMs are responsible for maintaining command and control of U.S. forces in their AoR during military operations, in times of conflict and peace, and during crisis interventions, such as humanitarian relief or disaster response activities. Two critical ingredients to any successful military operation are timely, reliable situational awareness and efficient, secure communication of that information to all participants in the operation. The cyber challenges to the realization of those ingredients fall into two main categories: mitigating difficulties caused by the inability of multiple users to share information over disparate computing systems and addressing problems caused by either a lack or overabundance of data relayed to COCOMs during operations.

As an illustration of these challenges, consider the difficulties faced by the U.S. disaster relief operation launched in response to the 11 March 2011 Great East Japan Earthquake, which led to a tsunami with waves higher than 40 m that traveled up to 10 km inland and that caused a major nuclear meltdown at the Fukushima Daiichi Nuclear Power Plant [2]. Operation Tomodachi, under the control of the U.S. Pacific Command (USPACOM), spanned nearly two months and involved multiple organizations responsible for the 24,000 U.S. service members, 189 aircraft, and 24 naval ships deployed in the mission [3]. During Operation Tomodachi, USPACOM found that existing military network resources were inadequate to keep pace with evolving situations and activities. Because the existing software tools and computing procedures were stove-piped (designed for specific organizations' needs) and not interoperable, they did not enable USPACOM to efficiently gain sufficient situational awareness of the mission and the environment, and did not support

on-the-fly acquisition or development of software tools better suited to the tasks at hand. Situational awareness also suffered because the information sent to command varied in quantity ("drought or deluge") and tools varied in their ability to process data.

The solution to the problem of stove-piped, incompatible tools is not simply providing access to more tools, and the ready availability of data is not necessarily an advantage. With the advancement of sensor systems for gathering data and the expansion of computing resources for processing, storing, and distributing data, operators have more access to more information than ever before. With this deluge of information comes the risk of information overload. The vast amounts of diverse information (e.g., text, video, imagery) that are disseminated daily throughout DoD commands and organizations strain the ability of analysts to develop a comprehensive picture of evolving situations. When the current tool set does not support the goals of the command or the individual operators, these drawbacks may become greater than the benefits of the expanded toolsets and datasets.

### **Challenge of the COCOM Acquisition Process**

Currently, COCOM acquisitions are conducted through Integrated Priority Lists (IPL). These lists represent an individual COCOM's most important capability needs prioritized across military service and function lines, risk areas, and long-term strategic planning issues [4]. These IPLs are then used to inform the programming and budgeting processes about COCOM needs. Each IPL represents the needs of an individual COCOM (e.g., USPACOM, U.S. Southern Command [USSOUTHCOM]) and is developed to satisfy the particular requirements and procedures of each COCOM's branches. This compartmentalization can lead to a lack of awareness of the overall capability needs across COCOMs, tools that are not generalizable across COCOMs, and redundant functionalities. Current tools are often stove-piped for individual threats and organizations, and updates are infrequent and difficult. Optimally, COCOM development and acquisition should provide agile, user-centered decision support tools that are (1) composable capabilities that can be built and modified on the fly and (2) interoperable, reusable applications that are generalizable across commands and threats.

### Cloudbreak and User-Centered Design

The Cloudbreak process has origins in both user-centered design and agile software development. Human-centered design, defined in the International Organization for Standardization's standard ISO 9241-210 [5], is an approach to interactive system development. Typically, this process uses the characteristics of relevant stakeholders (e.g., users) and their environment to define a set of requirements for design solutions; the tools developed to meet those requirements undergo user evaluations that then inform subsequent iterations of the tools. User-centered design requires significant upfront research and analysis of user needs, resulting in a longer time to deliver a working product. Agile methods, on the other hand, focus on rapidly delivering small sets of features onsite to customers, iteratively updating using a feedback loop between the developers and the users.

Traditionally, user-centered design has been seen as incompatible with the agile development process [6]. However, if the two are aligned, user-centered and agile methods can be used to maintain a close connection to users while rapidly iterating on system design and requirements [7, 8]. This hybrid strategy is flexible and holistic, taking into account the entirety of the problem space and allowing for incremental development that can make system modifications based on evolving circumstances.

While many developers in industry and academia have been reluctant to combine the two approaches to system design, researchers at Lincoln Laboratory have championed taking an agile, user-centered approach to aid in building effective, practical tools and visualizations that satisfy the requirements of their users [9]. In their review of user-centered design in cyber visualizations, Staheli et al. found that in the majority of visualization developments described in the published research, users were not even consulted during the design process [10]. Additionally, in the efforts discussed in that research, post-design evaluation of the visualizations was mainly limited to high-level qualitative analyses, such as surveys. During the development of the Extreme Malicious Behavior Viewer, Yu et al. interviewed users to understand how they interact with cyber data [11]. While the geographic locations of malicious cyber events may not seem to yield adequate information for cyber defense (most attacks will likely be clustered in populous locations), the team found that geolocation was a simple, intuitive option for con-

veying relevant information to users with limited cyber knowledge. The team's interviews revealed that a map displaying network activity in relation to geopolitical entities was helpful to users' decision making in identifying threats that target specific regions, employ language or culture-specific social engineering, or exploit localization or pirated software. When developing MacroScope, a network-based intrusion-detection system, Cunningham et al. based their design of the system display, RapIDisplay, on interviews with intrusion-detection analysts [12]. These interviews led to the incorporation of display features that are not common in many intrusion-detection systems: a presentation that allows rapid access to documentation and report generation, and a visualization of the confidence of an attack happening.

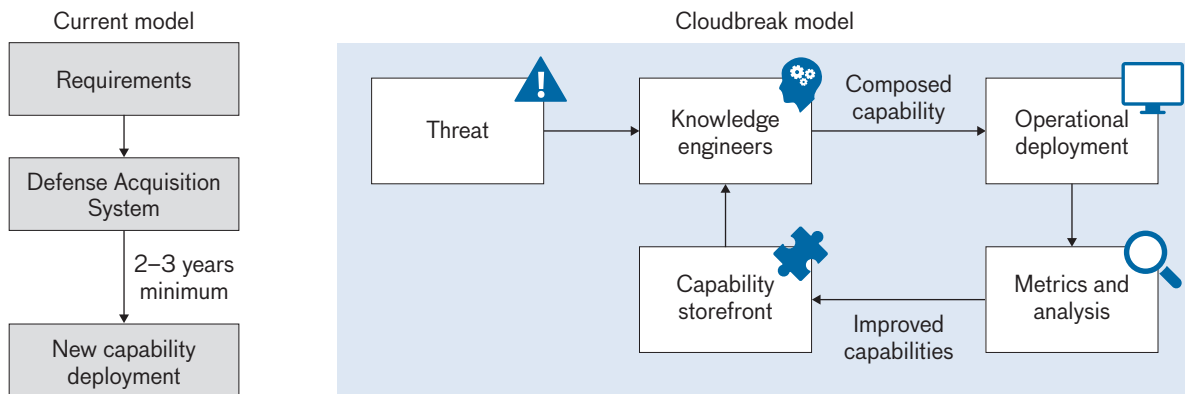
The Cloudbreak program applies methods used in previous successful system implementations to the development of rapid, composable designs and software. Cloudbreak's efficient, flexible iterative process is better suited to quickly and effectively providing tools to meet the complex and emergent needs of COCOMS than are the current models for technology delivery (Figure 1).

### Cloudbreak Process

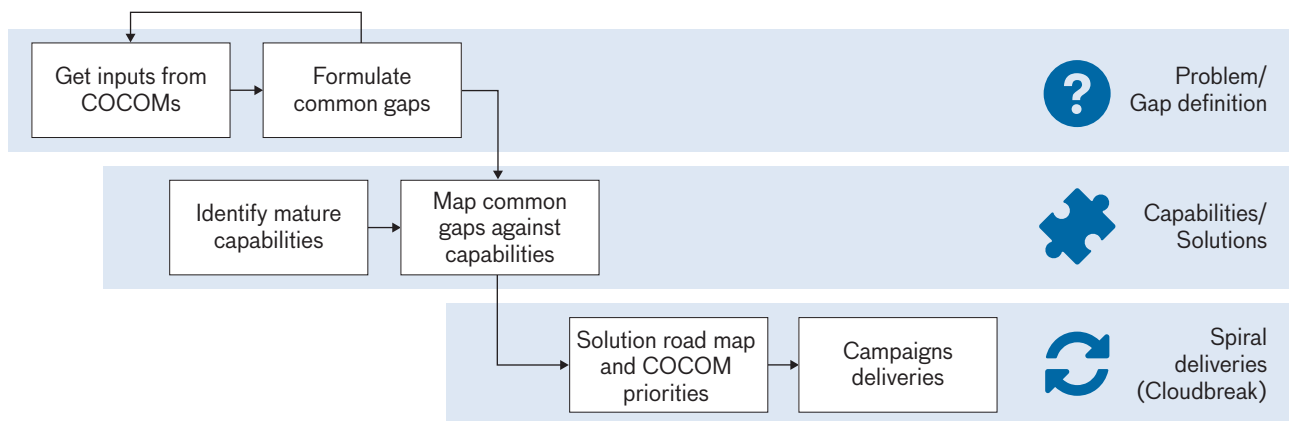
The Cloudbreak process leverages current capabilities to quickly deploy to operators newly composed software solutions for responding to emerging threats. Cloudbreak's approach is a cycle of problem definition, identification of relevant existing capabilities and solutions, and deliveries of new tools in spirals, each of which is informed by ongoing observations of the tools' productivity (Figure 2). The outcome of the mission for which the new capabilities were created and the lessons learned from their implementation are used to determine if the new capabilities can be applied, perhaps with modifications, at other COCOMs. This focus on post-deployment assessments enables a successful reuse of newly designed software across multiple COCOMs and mission areas. For example, a capability deployed for unclassified information sharing to support nation building in one COCOM's AoR could be repurposed to coordinate a response during a humanitarian assistance crisis or a disaster recovery operation.

The Cloudbreak process allows for activities to be completed concurrently and in various orders to address changing needs. The first step, defining the problem and software gaps, starts with assessing each COCOM and





**FIGURE 1.** This comparison of the Cloudbreak model to the current model for technology delivery shows the iterative nature of Cloudbreak: knowledge engineers assess the nature of a threat, compose from available technologies a solution to mitigate the threat, provide the solution capability to operational users, analyze the capability’s effectiveness, make improvements to the capability on the basis of the analysis, and add the new capability to the storefront for future use. The current Department of Defense acquisition cycle, which can take a minimum of two to three years, is a complete design and build of a new system for supplying the requisite functions.



**FIGURE 2.** The three-tiered Cloudbreak process begins with defining the needs of combatant commands (COCOMs). Existing software and mature capabilities that may provide solutions to those needs are evaluated against the gaps in the command’s current software tools. Developers propose a plan for creating new tools and deploy usable solutions that may require multiple development spirals to fully answer all requirements.

using the results to formulate a set of common gaps, which are then presented back to each COCOM to ensure that the descriptions of the gaps adequately summarize the COCOM’s specific problems.

Next, Cloudbreak practitioners identify a set of mature capabilities that map against the common gaps. These capabilities are essentially building blocks that can be acquired from a de facto “storefront” and can be composed into a new unified capability. Available capabilities include systems from other domains; previously matured technologies, including government and commercial off-

the-shelf (GOTS and COTS, respectively) systems; and additional data sources.

Once technology capabilities are mapped against needs, COCOMs are provided with a solution road map that links needs to available services and capabilities in the storefront. This plan provides information on how many and which COCOMs have a specific capability gap and what the implementation of the plan will cost and involve. Knowledge engineers, i.e., multidisciplinary engineers armed with an understanding of the needs and operational environment of the COCOMs and experi-

enced with diverse technology solutions available through the storefront, use common architectures to synthesize and tailor a capability that supports the operational need.

During the composition of a new capability, it is critical that the knowledge engineers involve the operators in an iterative development strategy to promote strong acceptance of the new tool. Formal and informal assessments of the new software can generate feedback that allows the engineers to remain responsive to operator needs. When the knowledge engineers insert the new technology into the operational environment, they can concurrently evaluate the technology's utility. Finally, the engineers transfer the newly composed capability, with improvements developed on the basis of lessons learned, back into the storefront to be used to address other gaps and emerging incidents across multiple commands.

### Elements of Successful Technology Insertion

The Cloudbreak process is a general framework for inserting new technology into previously composed systems. Through our work with the COCOMs, we have identified several elements critical to a successful technology insertion.

- Collaborating across organizations. Exchanging new, useful services and applications is a critical aspect of the Cloudbreak model. However, many command centers that have similar needs and missions do not currently take advantage of each other's capabilities. Cloudbreak provides a platform on which COCOMs can build an awareness of the capabilities and current gaps of other commands. Once COCOMs are cognizant of each other's systems and technology gaps, knowledge engineers can work across the commands to ensure that work is not repeated or further resources are not spent on solutions already available. COCOMs can leverage the previous work from other commands and work together to develop unified capabilities.
- Leveraging existing capabilities. COCOMs do not have access to unlimited resources; therefore, it is important that they maximize their resources. Currently, COCOMs are required to invest in GOTS and COTS systems, but these systems may not fully meet their requirements. Cloudbreak offers a process and platform for COCOMs to pull previously developed, known capabilities from one command to another; thus, individual COCOMs can insert high-quality solutions into their systems while expending significantly less time and resources.

- Customizable tools and composable architectures. Cloudbreak allows tools to be adapted to individual problems and commands. From the storefront, knowledge engineers can obtain capabilities to reach a 90% solution and then tailor those capabilities to attain a COCOM's goals. Each COCOM can utilize its own data feeds and develop other, low-level customizations to achieve a 100% solution. These customizations can then be integrated back into the storefront for use by other commands that may have similar requirements.

### Operational Deployment Case Studies

As a part of the Cloudbreak program, researchers from Lincoln Laboratory visited USPACOM, USSOUTHCOM, and the Defense Information Systems Agency to interact with users and understand the current technology needs and deficiencies across their organizations. During these visits, which have occurred regularly since 2012 and typically last at least a week, researchers interview numerous analysts to better understand their command-level technology gaps and analyst-level needs.

Once the problems were identified, the Cloudbreak team focused on cataloging capabilities and potential solutions available from Lincoln Laboratory, and COTS and GOTS providers. As a part of the initial assessment of the COCOMs' cyber programs, an exhaustive list of available capabilities was compiled and organized according to the mission area utilizing the capabilities, COCOM deploying the capabilities, and utility accruing from the capabilities. Tools identified as important for the cyber mission were cataloged on the basis of their applicability in improving situational awareness and the analytical process. In addition to identifying the tools and their primary usage, all tools were cataloged according to their current usage across COCOMs, estimated costs, designations as enterprise-level software, composability, availability in the Cloudbreak storefront, and maturity.

By identifying available technologies and aligning them with the current needs, the Cloudbreak methodology can support the combination of the latest technologies with existing tools, datasets, and capabilities. The key to effectively compiling capabilities is to understand the operational relevance of a technology.

The following case studies from the Cloudbreak program illustrate the practical implementation of the process.

### Case Study: Cyber Analytical Station

Operators at JCCs had employed commercial cyber defense tools to log network activity, monitor the network for anomalies, and generate alerts upon detection of potentially malicious activity. However, the information assurance policies that dictated monitoring on a wide range of activities resulted in millions of alerts each day. This abundance of data far exceeded the capacity of the JCC teams to effectively monitor network traffic. The JCCs needed a way to prioritize the incoming alerts so that operators could direct resources to processing the most relevant data.

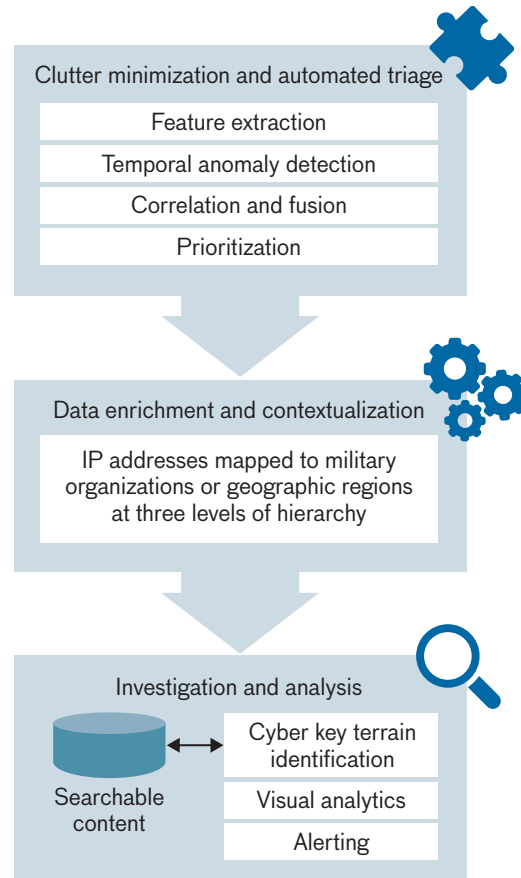
The Cloudbreak initiative identified seven requirements for an improved system for the JCCs:

1. Enable users to analyze cyber event data via a prioritized dashboard of critical and/or alarming events
2. Support the analysis of tens of millions of cyber events daily
3. Quickly identify the most important vulnerabilities
4. Facilitate timely creation of remediation plans to prevent escalation of cyber threat activity
5. Provide an interface with event data organized for easy exploration
6. Be easily and quickly learned
7. Enable forensic analysis of cyber event trends

These requirements were mapped against the current capabilities for operator situational awareness so knowledge engineers could identify technologies to integrate into a new tool—the Cyber Analytical Station. Typically, only mature capabilities are considered for integration, but in the case of the Cyber Analytical Station, no suitable capabilities existed, necessitating a custom development effort.

From the gap analysis, three predominant requirements for the system were apparent: the technology would have to (1) perform automated triage, (2) enrich data and apply context, and (3) support the investigation and analysis process. Each of these requirements contains several subrequisites as delineated in Figure 3. These requirements were then mapped to the available Lincoln Laboratory and COTS and GOTS technologies.

The Cyber Analytical Station aims to provide operations centers at military commands with the cyber situational awareness necessary for monitoring and managing the performance, security, and integrity of computer networks. To compose the final operational prototype, knowledge engineers integrated several mature research-grade capabilities:



**FIGURE 3.** The above data analytical features required by a system to achieve satisfactory situational awareness were identified by the Cloudbreak researchers. During clutter minimization and triage, the system should be able to extract key features, perform anomaly detection, correlate and fuse data from multiple sources, and prioritize alerts. Next, the system should map Internet protocol (IP) addresses to known entities and/or geographic regions to contextualize the incoming data. Finally, the data should be stored in a way that allows the cyber analyst to interact with the data either in a raw form or through a visual analytic.

- Ingest and enrichment. The final composed application was able to ingest and enrich necessary cyber data with geolocation and organizational data from known sources, execute automated analyses, and deliver the outcomes through visualizations on a user-friendly web interface. The ingest capability is responsible for loading, enriching, and storing cyber data, which are principally acquired from network intrusion-detection systems. Because the Cyber Analytical Station can perform an automated enrichment while loading and storing data, it can help cyber operators by providing the context they need to more quickly categorize and interpret the data.

- Event detection and prioritization. This automated data analysis capability provides users with rapid detection and prioritization of anomalies in the data. The station proactively identifies significant events rather than just offering alerts that are based on pre-determined, rigid heuristics, also known as “trip-wire” conditions. This capability was easily addressed via a prototype application previously developed at Lincoln Laboratory for anomaly detection [13]. Finally, outcomes of the ingestion, enrichment, and analysis are presented to users through an interactive interface that enables analysts to review prioritized anomalies and drill down to the underlying data (Figure 4).

Even though several of the capabilities within the Cyber Analytical Station were already mature, through the Cloudbreak technology insertion process, we were able to deliver a solution that was customized for the needs of the COCOM. Early on, while Lincoln Laboratory staff were working to solicit specific requirements for transitioning the Cyber Analytical Station to COCOM operational use, JCC operators had a great many firm requirements and new questions. These questions and requirements were then spiraled back into the prototype application in an iterative strategy, supporting the development of a system that met the evolving needs of the JCCs and attained strong user acceptance.

During the initial stages of Cloudbreak, primary efforts centered on developing the detailed measures of a system’s effectiveness and performance that would be used in conducting formal assessments. For the Cyber Analytical Station project, formal evaluations helped provide overall impressions on the station, but the most useful feedback came from simply observing operators and recording their behaviors and subjective commentaries as they used the system. For example, during one informal observation period, we watched an operator manually perform a copy-and-paste operation. To capture the details of an anomaly for inclusion in a cyber-incident report, the operator used the mouse to highlight 10s of table rows and paste them into a spreadsheet. The development team captured this behavior as a new requirement, executed a feature “quick turn” to implement a new functionality for converting tables to spreadsheets, tested this software update, and delivered it to multiple COCOMs within three days.<sup>2</sup>

<sup>2</sup>The update did not include security-relevant changes and did not trigger an information assurance reaccreditation.

## RESULTS OF THE CYBER ANALYTICAL STATION TECHNOLOGY INSERTION

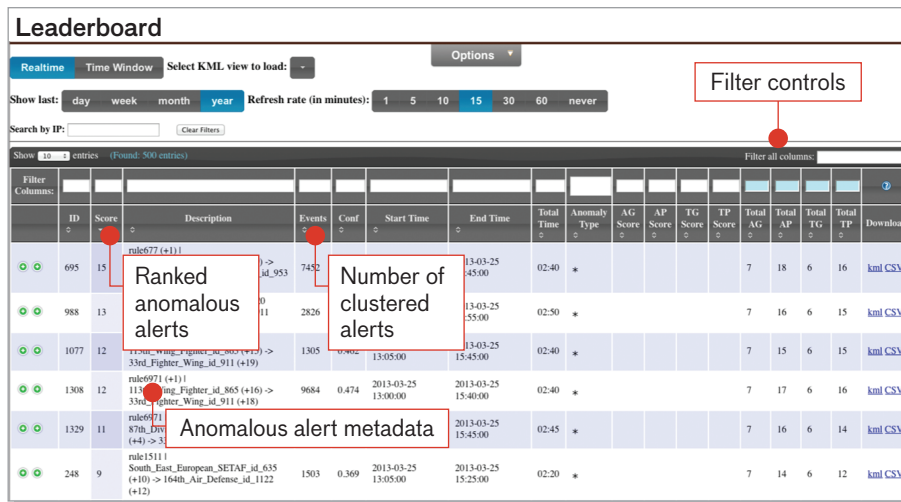
As demonstrated in the Cyber Analytical Station case, the Cloudbreak process can be used to mitigate risk within the resource-constrained JCCs. A lean, efficient interdisciplinary team was able to execute at a low cost, and the providers of existing capabilities benefit from interacting with a broader and more diverse user base. Involving cyber operators early and reacting to their requirements with an iterative strategy were critical to gaining strong user acceptance of the new capabilities. Once the initial new technology was delivered, assessments of it, both formal and informal, were invaluable in generating useful feedback. For example, the need for an automated copy-and-paste function may have not been identified if we had not informally observed how operators actually interact with the software. Finally, by remaining responsive to the requests of users and to the continuing evolution of the JCCs, the Cloudbreak team was able to deliver quick-turn features and to modify the tool set in a matter of days.

During its initial deployment, the Cyber Analytical Station quickly demonstrated utility during a brute-force attack (i.e., an exhaustive trial-and-error method to breach password or cryptographic protections) against a public-facing file-transfer server. The Cyber Analytical Station provided enhanced cyber situational awareness by enriching the data with context and enhancing data exploration. The Cyber Analytical Station provided interactive access to data not previously available and allowed operators to focus on high-priority tasks, thus improving operators’ efficiency and accuracy.

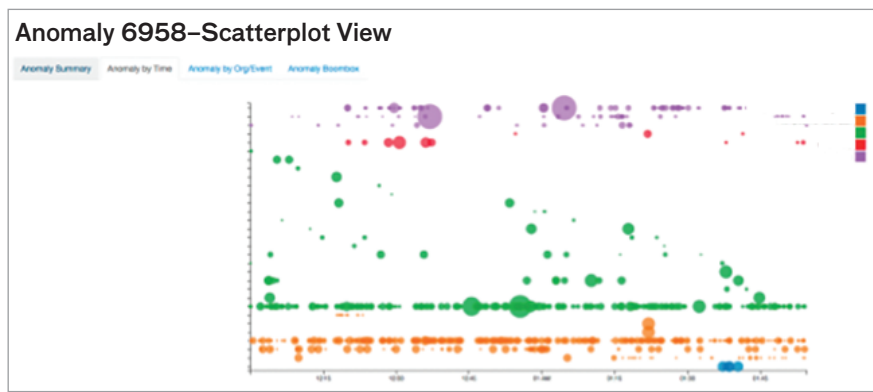
### Case Study: Cyber Dashboard

The Cyber Dashboard was conceived as a means to integrate and visualize disparate data sources (e.g., cyber, operational, and intelligence data) to support information exchange and commanders’ cyber situational awareness.

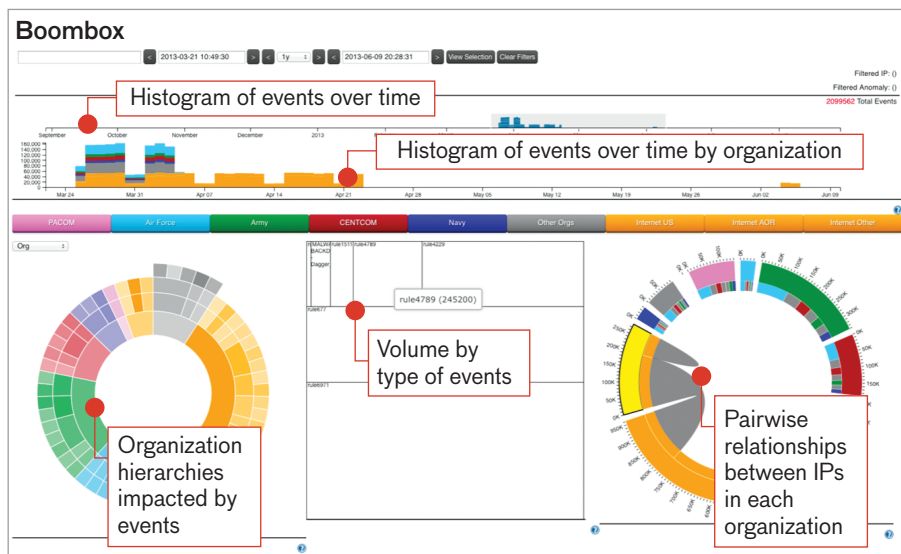
Achieving operational cyber situational awareness requires the integration of data from six classes of information: the current and near-term threat environment, anomalous network activity, vulnerabilities, key cyber terrain, current operational readiness, and ongoing operations [14]. In the COCOMs, the providers of this information were stove-piped within each organization, requiring the Cloudbreak team to work across joint services to locate and obtain access to the authoritative sources. In many



(a)



(b)



(c)

**FIGURE 4.** This interactive visualization tool consists of an anomaly leaderboard (a); a screen for anomaly inspection (b); and a visualization of all alerts on a display called the Boombox (c). The leaderboard display allows the watch-floor manager to focus on high-priority alerts and to group and sort those alerts on the basis of various criteria. The scatterplot view shows the appearance, prevalence, and disappearance of alerts on the network. The Boombox display is designed to allow analysts to explore the network data: it color-codes attacks in terms of an organization’s hierarchy; its top third shows histograms of overall attacks experienced by organizations and of attacks encountered by each organization over time; its middle region shows the volume of attacks presented as a treemap in which the rectangles for different attack scenarios are proportional to the total number of attacks; and the right circular graphic displays the pairwise communications occurring between systems in each organization. Collectively, the three displays (a–c) help analysts to prioritize incoming events, to understand why events are considered anomalous, and to discover relevant information about events and put it into an organizational context.

cases, because data providers were not under the jurisdiction of the COCOMs, the Cloudbreak team had to make connections to external data-providing entities, such as the Joint Improvised-Threat Defeat Agency.

Analysts rely on a variety of data sources to maintain accurate cyber situational awareness. The initial focus for the dashboard was on the display of anomalous activity; subsequent development spirals incorporated additional data sources into the display. Analysts at the COCOMs also require the ability to overlay additional arbitrary data sources on the map to visually fuse information from various sources. This capability allows analysts to connect information across multiple sensors and visually inspect and analyze the relations and interactions between the data. For example, a network outage located in the same geographical area that is experiencing an unusually high number of alerts could be a cause for concern; a display coordinating those two pieces of information allows operators to identify a situation that may need further investigation. Additional data sources could number in the 100s, depending on the situation.

The Cloudbreak team identified the following additional capability needs. Analysts desired functionality for preserving the current state of the dashboard and sharing it with others. Two reasons drove their request for a shareable dashboard: (1) analysts wanted to share a link for a particu-

lar finding in the data with other analysts or managers, and (2) analysts wanted to create custom dashboards to address emerging situations. The needs and the respective dashboard approaches to meeting those are illustrated in Table 1.

The Cyber Dashboard was built as a series of Microsoft’s SharePoint Web Parts to best leverage existing capability. Four types of SharePoint libraries compose the capability: the Map libraries render the main map canvas and geospatial data, the Tree/Graph libraries render hierarchical data, Timeline libraries render temporal data, and Data libraries transform, correlate, and archive the original data sources. Each of the Web Parts requires configuration of a data source from a common data format: Keyhole Markup Language (KML), Extensible Markup Language (XML), Comma Separated Values (CSV), JavaScript Object Notation (JSON), or SharePoint lists are all supported. Data are not stored or managed by the Cyber Dashboard, but remain in their original location and under their existing access control policies. Dashboards can be customized by modifying the configuration files and can be shared via unique Uniform Resource Locators (URL). The architecture is illustrated in Figure 5.

The design of the dashboard visualization utilizes a canvas-palette metaphor; a geographic map serves as the background of the browser window and as a canvas upon which geospatial data are depicted. Multiple

**Table 1. Dashboard Capability Needs of COCOMs and Cloudbreak Solutions**

NEEDED CAPABILITY	SOLUTION
Commanders need a flexible, customizable dashboard that presents a common operational picture of cyber situational awareness	Provide an agile display that has a “brief from tool” capability
Operators need a system that enables them to react promptly to evolving situations (e.g., Ebola outbreaks or disaster response efforts)	Supply a dashboard whose easy configuration and customization enable a new dashboard to be created and shared within minutes
System must be capable of integrating data sources from within and outside COCOMs	Eliminate the use of back-end databases; allow data to be accessed from original authoritative sources
Display must be easy to interpret for operators who may have limited experience with and knowledge of the onscreen visualization	Create a display that uses a geospatial background (a familiar reference point for users) and that supports multiple data formats (potentially 700 data sources)
Capabilities must mitigate problems of COCOMs’ existing limited infrastructure, hardened systems, long acquisition process	Utilize in-house SharePoint infrastructure and expertise as much as possible

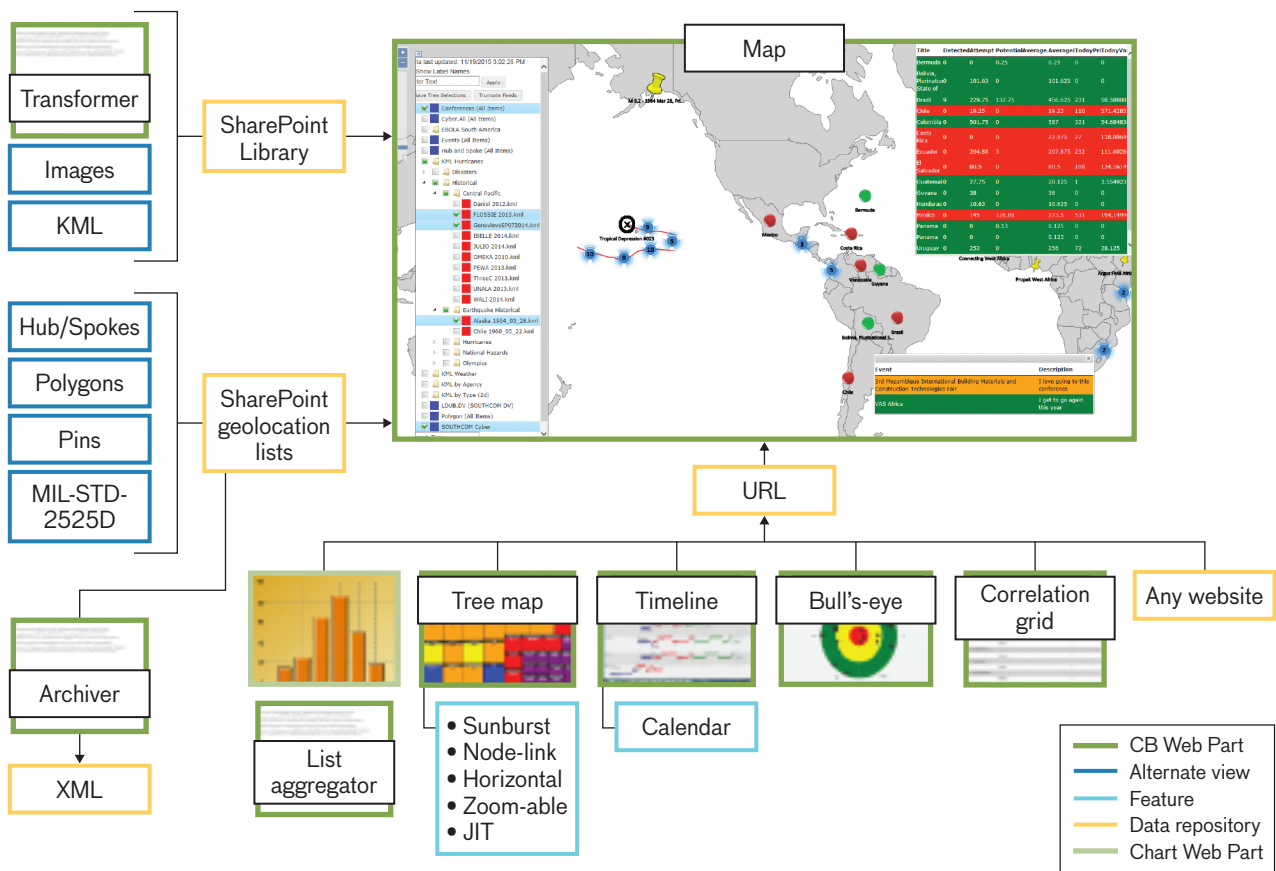
floating palettes are then layered on top of this canvas for non-geospatial data. Analysts acknowledged that a geospatial map may not be the most suitable display paradigm for all cyber data, but the map provides a good start as a common visual representation that is familiar and accessible to all audiences. The flat design style of the map also helps remove visual clutter, such as the representation of terrain features, and allows the data points to be viewed more clearly.

A permanent, large palette on the left contains the master list of data sources; other palettes can be drawn on demand and positioned as needed. Palettes can display data in a number of conventional visualizations (e.g., tree map, node-link diagram, sunburst chart, timeline). Each visualization palette has basic parameters that can be configured: data sources, transformations of the data sources (correlation, georeferencing), and graphical elements, such as sizes or colors. The configu-

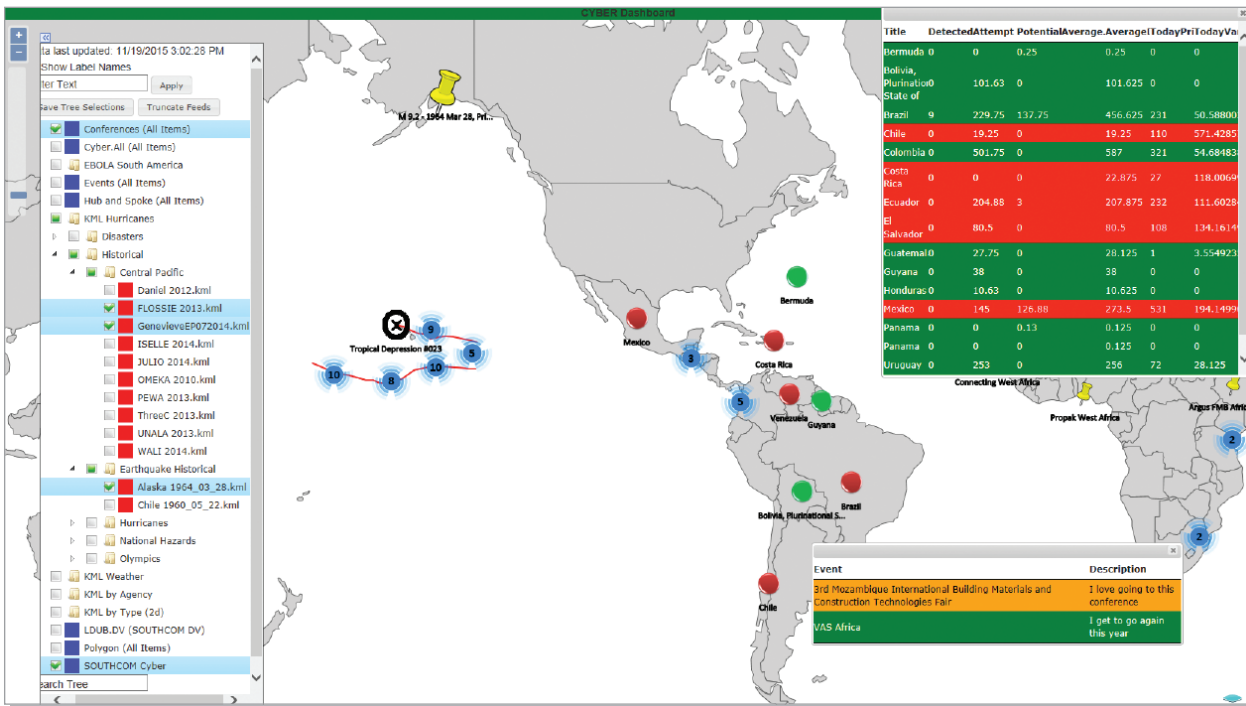
ration for the entire dashboard can be saved and shared. The final design is illustrated in Figure 6.

**RESULTS OF THE CYBER DASHBOARD DEVELOPMENT**

The dashboard, using operational datasets, was demonstrated for analysts in three cyber operations centers to solicit feedback on the display and to gauge its operational utility. Analysts and managers provided qualitative feedback via comments, both as a group during the demonstration and in private conversations after the presentation. Developers then worked with analysts individually to identify new requirements, deliver software updates, incorporate new data sources, and gather further feedback. As a result of the interactions with users, the team delivered 92 iterations of the software within the 2015 calendar year. These deliveries included two significant product features that were incorporated on the basis of user feedback: (1) visualization network and circuit dia-



**Figure 5.** The Cyber Dashboard architecture is based on SharePoint Web Parts (green), which render authoritative data types, i.e., data that have been verified as coming from an official trusted source, (yellow) on a geospatial canvas. Many of the Web Parts offer multiple options (e.g., node-link or horizontal charts, calendar) for rendering visualizations (blue).



**FIGURE 6.** The Cyber Dashboard design uses a background of a flat geographic map over which are laid palettes that present various types of data. In this image, the palette on the left side enumerates the data sources available. Users can interact with icons on the map to obtain details on demand; the drill-down information is displayed in the floating palettes depicted on the right side of the screen. The colors in the palettes can be defined for each data source; in this example, color relates to severity of the event, with green being low and red being high.

grams as a geospatial overlay and (2) a what-you-see-is-what-you-get (WYSIWYG) editor to assist novice users in managing and configuring their data sources.

**FUTURE DESIGN CONSIDERATIONS FOR THE CYBER DASHBOARD**

During the development of the final dashboard design, we identified several considerations to address in future versions. These items were considered out of scope for the original project but remain important operational considerations:

- Standardization. When users are incorporating 100s of data sources in a single geospatial display, it is difficult for a human to keep track of the provenance of the data. Color-coding, iconology, and taxonomy standards all play a role in helping a human in the loop to distinguish between data sources and perform visual search tasks. Conventions for how to standardize these elements must be included in future design guidelines.
- Data source “freshness.” When using multiple data

sources, analysts must understand how current the information is. Standard conventions for how to convey the timeliness of data need to be developed.

- Representation of complex relationships. For the cyber security domain in particular, complex dependencies beyond geospatial coordinates exist between data points. Understanding how these additional visual conventions can work in conjunction with map-based representations would improve situational awareness.
- Interpalette interactions. A natural next step for this project is to explore interaction paradigms by using the map-canvas metaphor to produce a generic framework that can be applied to interactions among arbitrary data sources.
- Intelligent fusion. Our current design allows analysts to do basic data management tasks but requires manual integration of cyber data across sensors and visualizations. Providing the ability to automatically tie together data sources on the basis of common fields or other dimensions would be beneficial.



## Lessons Learned

Over the past three and a half years, the Cloudbreak team has learned many valuable lessons that researchers can apply to future DoD applications of the Cloudbreak approach.

Informal discussions and interviews emerged as a rich source of data to identify key behaviors and needs of the operators, enabling the team to quickly turn around new functionalities to improve operators' subjective experiences and performance while using the software. These interactions with the operators also were useful in gaining an understanding of the training, expertise, and experiences of the current operators. For example, during our discussions, we found that current Internet applications (e.g., Google, YouTube, social media) define operators' expectations of how DoD information technology (IT) systems should work. Operators expect speed and behaviors consistent with these applications; however, developing tools that accommodate both DoD system restrictions and operator expectations is a challenging balancing act. As an example, for a large text-analytic system, we updated the search interface from a powerful, expressive Boolean language to one that behaved similarly to popular Internet search engines.

Because of budget and security constraints, web browsers in use by the DoD tend to lag behind those found on the Internet. The latest web technologies, such as HTML5, are often simply not supported by DoD systems. For instance, we could not use the HTML5 canvas and scalable vector graphics elements to drive rich visualizations in all environments. We had to adjust our mindset and strike a balance between advanced technology and compatibility when we designed the web-based interfaces.

We found that operators were quick to discard or ignore capabilities and features that required them to employ many steps to accomplish a task; therefore, we worked hard to design utilities that eliminated excess steps from operator workflow and to take advantage of familiar interaction paradigms. For example, many workflows at COCOMs revolve around sharing data stored in a COTS enterprise content management system. Operators were more likely to use a capability when it was integrated into their content management system. We noticed a similar reaction to integrating an existing system with a public key infrastructure (PKI): By eliminating the need for additional username and password combinations and

integrating the PKI with the operators' existing authentication system, we removed another barrier to their adoption of new tools.

From an organizational point of view, one cannot make assumptions that all operations centers share common processes. While most COCOMs have a common goal, there are variances in the battle rhythms based on the preferences of the leader and the current available skillsets within the command. Each leader has specific requirements for the way he or she prefers to consume information. The systems we provide must help operators prepare ahead of time for their commanders' needs. This task does not, however, equate to flooding leaders with information. The solution is to supply commanders with timely, mission-relevant data and to preserve other details for on-demand access.

The ultimate success of Cloudbreak is the users' adoption of the new technology. Because COCOMs are extremely busy, and the operators' time is split in many different directions, changes to systems must demonstrate immediate, recognizable benefits. Adoption results when the new capability demonstrates that it has practical value. Without this value, every capability is "just another tool" and will sit idle. Our example of eliminating the manual copy-and-paste operation is a great example of the addition of an immediate, clear benefit. Furthermore, we found that once an initial value of a tool or concept had been established, operator enthusiasm for new capabilities and the Cloudbreak process increased significantly.

Furthermore, aside from contractors, most operators at COCOMs rotate duty assignments every two to three years. Therefore, system developers cannot assume the operators' level of technical expertise or their familiarity with software and existing systems to remain constant. Continuing to communicate with operators and leadership to evolve systems as COCOM personnel, work practices, and goals change is critical to long-term adoption of systems.

Once adopted, capabilities cannot be set up and left to run indefinitely. Improvements, bug fixes, constant security monitoring, and hardware concerns all drive the need for a clear operation and maintenance plan. Cost, particularly that related to operations and maintenance, is a major consideration for DoD leadership. The COCOMs do not have the resources to either take on additional IT responsibilities or hire external IT support services. Consequently, in addition to providing capabilities, we were charged with determining how the

capabilities would be sustained. Working with multiple government organizations and leveraging common interests, we were able to construct ways to distribute the costs and responsibilities for operating and maintaining systems among developers and users.

### Future Work

The Cloudbreak initiative has been successful within a resource-constrained DoD. Through technology reuse and composability, we enabled cost savings. Our process for successful capability insertion has its foundation in a strong relationship with operators. The connections and trust we developed with them helped us discover the true areas where they most needed help. We allowed the operators' needs to drive the technology development, thus supporting operators by rapidly filling critical COCOM technology gaps. The Cloudbreak process also led to a new relationship with operators that could inform other collaborative projects and provide us access to operational datasets for future research, development, and experimentation.

Our experience with the Cloudbreak model leads us to endorse its continuation and replication across other areas of the DoD and within Lincoln Laboratory. The individual capabilities we have delivered, such as the Cyber Analytical Station, have reduced risk for systems currently being developed by identifying and validating requirements. The workflows and the features that we helped define have the support of JCC operators and COCOM leaders; therefore, the process of developing requirements for future systems does not need to start from scratch.

Moving forward, we will continue to apply an agile, user-centered research and development model. Currently, we are using the Cloudbreak approach in several ongoing efforts with USCYBERCOM, the U.S. Transportation Command, and the U.S. Navy. Building off lessons learned and the relationships with and access to operators developed under Cloudbreak will enable current and future Lincoln Laboratory programs to effectively align capability development with user requirements and to accomplish successful technology insertions. ■

### References

1. T.J. Doshier, "NORAD, USNORTHCOM Joint Cyber Center Stands Up," U.S. Northern Command website, 2012, available at <http://www.northcom.mil/Newsroom/tabid/3104/Article/563711/norad-usnorthcom-joint-cyber-center-stands-up.aspx>.
2. "2011 Tōhoku Earthquake and Tsunami," Wikipedia, Jan. 2016, available at [https://en.wikipedia.org/wiki/2011\\_T%C5%8Dhoku\\_earthquake\\_and\\_tsunami](https://en.wikipedia.org/wiki/2011_T%C5%8Dhoku_earthquake_and_tsunami)
3. "Operation Tomodachi," Wikipedia, Dec. 2015, available at [https://en.wikipedia.org/wiki/Operation\\_Tomodachi](https://en.wikipedia.org/wiki/Operation_Tomodachi)
4. J.H. Pendleton, M. Morgan, N. Bleicher, M. Jones, M. Silver, J. Spence, and K. Williams, "Defense Management: Perspectives on the Involvement of the Combatant Commands in the Development of Joint Requirements." Washington, D.C.: Government Accountability Office, 2011, available from the Defense Technical Information Center at <http://www.dtic.mil/dtic/tr/fulltext/u2/a546159.pdf>.
5. "Ergonomics of Human-System Interaction—Part 210: Human-Centred Design for Interactive Systems," ISO 9421-210. Geneva: International Organization for Standardization, 2010.
6. P. McInerney and F. Maurer, "UCD in Agile Projects: Dream Team or Odd Couple?" *Interactions*, vol. 12, no. 6, 2005, pp. 19–23.
7. D. Sy, "Adapting Usability Investigations for Agile User-Centered Design," *Journal of Usability Studies*, vol. 2, no. 3, 2007, pp. 112–132.
8. D. Fox, J. Sillito, and F. Maurer, "Agile Methods and User-Centered Design: How These Two Methodologies Are Being Successfully Integrated in Industry," *Proceedings of the Agile 2008 Conference*, 2008, pp. 63–72.
9. S. McKenna, D. Staheli, and M. Meyer, "Unlocking User-Centered Design Methods for Building Cyber Security Visualizations," *Proceedings of the 2015 IEEE Symposium on Visualization for Cyber Security*, 2015, pp. 1–8.
10. D. Staheli, T. Yu, R.J. Crouser, S. Damodaran, K. Nam, D. O'Gwynn, S. McKenna, and L. Harrison, "Visualization Evaluation for Cyber Security: Trends and Future Directions," *Proceedings of the 11th Workshop on Visualization for Cyber Security*, 2014, pp. 49–56.
11. T. Yu, R. Lippmann, J. Riordan, and S. Boyer, "Ember: A Global Perspective on Extreme Malicious Behavior," *Proceedings of the 7th International Symposium on Visualization for Cyber Security*, 2010, pp. 1–12.
12. R.K. Cunningham, R.P. Lippmann, and S.E. Webster, "Detecting and Displaying Novel Computer Attacks with Macroscopic," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 31, no. 4, 2001, pp. 275–281, doi:10.1109/3468.935044.
13. K.M. Carter and W.W. Streilein, "Probabilistic Reasoning for Streaming Anomaly Detection," *IEEE Statistical Signal Processing Workshop*, Ann Arbor, Mich., 2012, doi: 10.1109/SSP.2012.6319708.
14. J. Dressler, C.L. Bowen, W. Moody, and J. Koepke, "Operational Data Classes for Establishing Situational Awareness in Cyberspace," *Proceedings of the 6th International Conference on Cyber Conflict*, 2014, pp. 175–186.

**About the Authors**



**Diane Staheli** is a member of the technical staff in the Cyber Systems and Operations Group at Lincoln Laboratory. Her current projects focus on field research with cyber security analysts and operators and the translation of user needs into visualization capabilities. Her research interests include cyber decision making,

cyber human cognition, visual analytics, visualization evaluation, human-computer interaction, and emerging user interfaces. She joined the Laboratory in 2010, bringing 10 years of experience in industries ranging from a small home-networking startup to a global information security company. She serves on the board of the New England Chapter of the Human Factors and Ergonomics Society and is a current program committee member for the IEEE Visualization for Cyber Security Symposium, the VAST Challenge, and the Visualization for Data Science Workshop. She holds a bachelor's degree in communication, studio art, and film from the University of Massachusetts, Amherst, a master's degree in human factors in information design from Bentley University, and a master's degree in information technology and software engineering from Harvard University.



**Vincent F. Mancuso** is a member of the technical staff in the Cyber Systems and Operations Group at Lincoln Laboratory. His research interests include exploring issues of human factors in cyber operations and team cognition in distributed environments. Prior to joining the Laboratory, he was a postdoctoral researcher in

the Human Performance Wing's Applied Neuroscience branch at the U.S. Air Force Research Laboratory, conducting research on cyber operator performance monitoring and optimization. He holds a bachelor's degree in information systems and human-computer interaction from Carnegie Mellon University and a doctoral degree in information sciences and technology from the Pennsylvania State University.



**Matthew J. Leahy** is a member of the technical staff in the Cyber Systems and Operations Group. He currently works on projects dealing with operations and cyber situational awareness. Previously at Lincoln Laboratory, he has worked on distributed test beds for live testing of ballistic missile defense systems, radar systems

development, network communications, and distributed software systems. He holds a bachelor's degree in computer science from Worcester Polytechnic Institute and a master's degree in information technology with a specialization in software engineering from Carnegie Mellon University.



**Martine M. Kalke** is an assistant leader in the Cyber Systems and Operations Group. She currently directs projects to improve network operations and cyber situational awareness. At Lincoln Laboratory, she has previously worked on the development and evaluation of advanced algorithms for ballistic missile defense and intelligence,

surveillance, and reconnaissance applications. She also was involved in designing a system architecture for the Missile Defense Agency while she was stationed in Washington, D.C. She holds a bachelor's degree in physics from Carleton College, and a master's degree in physics and a doctorate in condensed matter physics from Indiana University. Her doctoral thesis focused on computational condensed matter physics.

# Recommender Systems for the Department of Defense and Intelligence Community

Vijay N. Gadepally, Braden J. Hancock, Kara B. Greenfield, Joseph P. Campbell,

William M. Campbell, and Albert I. Reuther

Recommender systems, which selectively filter information for users, can hasten analysts' responses to complex events such as cyber attacks. Lincoln Laboratory's research on recommender systems may bring the capabilities of these systems to analysts in both the Department of Defense and intelligence community.



**In the past five years, the machine learning and artificial intelligence communities have done significant work in using algorithms to identify patterns within data.**

These patterns have then been applied to various problems, such as predicting individuals' future responses to actions and performing pattern-of-life analysis on persons of interest. Some of these algorithms have widespread application to Department of Defense (DoD) and intelligence community (IC) missions. One machine learning and artificial intelligence technique that has shown great promise to DoD and IC missions is the recommender system, summarized by Resnick and Varian [1], and its extensions described by Adomavicius and Tuzhilin [2]. A recommender system is one that uses active information-filtering techniques to exploit past user behavior to suggest information tailored to an end user's goals. In a recent working paper [3], the Office of the Director of National Intelligence's Technical Experts Group's Return on Investments team has identified recommender systems as a key "developing application" in their process map of "The Intelligence Cycle and Human Language Technology." The most common domain in which recommender systems have been used historically is commerce: users are customers and the objects recommended are products. Other feasible uses for recommender systems include recommending actions, e.g., suggesting a direct traffic route, and following interactions between users, e.g., proposing possible colleagues as the popular service LinkedIn does.

In the cyber arena, recommender systems can be used for generating prioritized lists for defense actions [4], for detecting insider threats [5], for network security [6], and for expediting other analyses [7].

### Elements of Recommender Systems

Recommender systems rely on four important elements:

- *Information filtering.* Recommender systems do not singlehandedly convert data to knowledge; they are just one component of the information pipeline. Sensors collect data, data processing turns those bytes of data into useful pieces of information, and then recommender systems help to filter that information into the most relevant pieces from which a human can extract knowledge and take action. Note that filtering referred to here does not imply deletion of any information but rather prioritization.
- *User behavior.* The value of having computers learn from user behavior rather than apply prescribed rules or heuristics is that the users are never required to explicitly state what the rules are. The rules by which users make decisions are inferred from the way the users act. This utilization of user behavior rather than heuristics enables recommender systems to reflect nuances in individual human preferences that would otherwise be difficult to quantify. It also provides us with a simple test for classification of decision support systems: if a system makes recommendations that do not include considerations of past user behavior, then it is not a recommender system.
- *Suggest information.* Recommender systems operate under a “push” rather than a “pull” paradigm. An information-retrieval system, such as a search engine, is guided by a query submitted by the user—a pull for information. Recommender systems, on the other hand, utilize user behavior and context history to ascertain the needs of users and are therefore equipped to predict or prescribe, i.e., push, new information to the user.
- *End user goals.* The main distinction between recommender systems and the broader class of filtering and sorting techniques is the applicability of the output of a recommender system to the needs of a particular user or group of similar users.

Recommender systems consist of four primary components: users, objects, ratings, and a model. Users include anyone whose behavior is being recorded in some

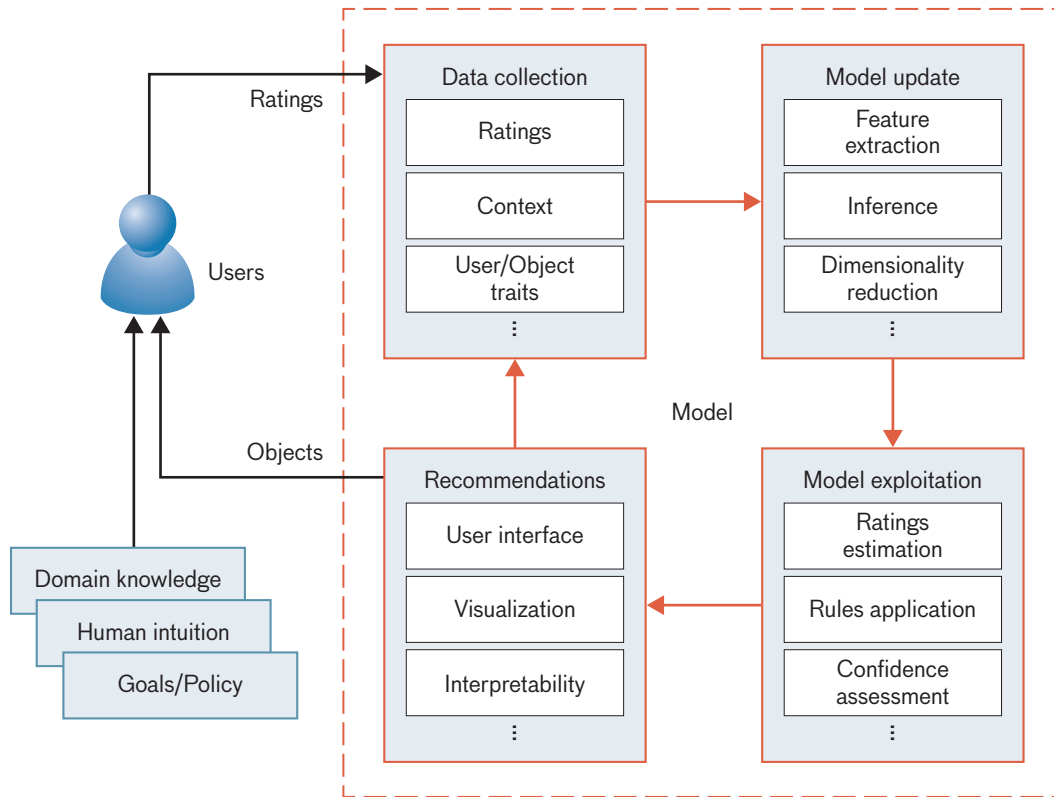
way to train the recommender system or anyone who is receiving recommendations. Objects refer to products, documents, courses of action, or other recommendations. Ratings are some quantifiable measure of the utility of a given user-object pair and may come from explicit feedback (e.g., thumbs-up votes, assessments on a five-star rating scale, or text reviews) or implicit feedback (e.g., number of clicked links, number of downloaded files, or time spent on a page). The model is used to process known ratings and make recommendations based on the predicted ratings for unrated user-object pairs. The functional architecture depicted in Figure 1 shows these four elements, with additional detail shown for the four stages in the workflow of a generic model.

Recommender systems are able to make relevant suggestions in a given situation by observing how users act (i.e., recording ratings assigned to particular objects in a specific context). How users act is, in turn, affected by the goals or policy that the user follows, the user’s intuition about what objects will satisfy those goals, and domain-specific knowledge that the user may have. This information is generally not formalized or conveyed to the recommender system. However, as the recommender learns from the user behavior that is affected by these influences, its recommendations will begin to reflect these influences.

User behavior is recorded in the data collection stage of a recommender system. In addition to ratings, information about traits of users, such as range of ratings or scores, objects, or context may be recorded. Context denotes situation parameters that can be known by the system and may have an impact on the selection and ranking of recommendation results.

Once these data have been collected, they are used to update the model of user preferences. First, significant features such as important aspects of a dataset (e.g., in a cyber network log, one feature may be time of logged event) must be extracted. Explicit user ratings may be entered directly, whereas implicit ratings may require some processing or inference to relate user behavior to a quantifiable rating to be stored. To reduce noise and lower computational complexity, some form of dimensionality reduction (i.e., a mechanism to reduce the number of variables being considered to the most critical variables) is often performed at this stage.

Once the model is updated, the next task is to estimate the ratings that the current user would give to the objects



**FIGURE 1.** A recommender system consists of users, objects, and ratings that interact with each other through a model developed by the recommender system. A recommender system takes information collected from domain knowledge, human intuition and goals, or policy and combines that information with user ratings. The recommender model is derived from data collections, model updates, model exploitation, and recommendations from other users or previous actions.

with which he or she has not yet interacted. Collaborative filtering and content-based, knowledge-based, or hybrid techniques (described in the next section) are used to generate these rating estimates. At this stage, additional recommendation rules, such as favoring recommendations that support specific objectives, may be applied. For example, a commercial recommender may be designed to favor products with large profit margins.

Then, using the estimation just performed, the recommender returns results to the user in a desired form (e.g., a top result, top *n* list of results, or all results above a given threshold). The subsequent actions of the user are recorded, and the cycle repeats.

**How a Recommender System Works**

To illustrate how a recommender system works, let us look at a very simple recommender system that recommends online articles or documents for an analyst to

examine. In this example (modified from Jannach et al. [8]), we are applying a collaborative-filtering recommender system in which analysts are searching through a corpus of online documents for information about potential exploits or cyber attacks. In this scenario, because the number of documents is greater than the number of analysts, the analysts rely on a recommender system to prioritize important documents. For ease of illustration, we will consider only five analysts and six online documents although a real-world system could easily consist of many millions of analysts and documents. Assume that whenever an analyst reads a document, that document is given a rating on a scale of 1 to 5 (1 = not useful at all, 5 = very useful). This rating may come from both explicit analyst input and implicit input. Shown in Table 1 are the analyst ratings for the documents.

In this illustration, we want to predict what the rating of Analyst 1 would be for Doc 5 and Doc 6. The document

**Table 1. Analysts' Document Ratings**

	DOC 1	DOC 2	DOC 3	DOC 4	DOC 5	DOC 6
Analyst 1	5	3	4	4	?	?
Analyst 2	3	1	2	3	3	1
Analyst 3	3	3	1	5	4	5
Analyst 4	4	3	4	3	4	2
Analyst 5	1	5	5	2	1	3

**Table 2. Similarity Scores Between Analysts via Adjusted Cosine Similarity**

	ANALYST 1	ANALYST 2	ANALYST 3	ANALYST 4	ANALYST 5
Analyst 1	–	0.85	0.00	0.71	–0.79
Analyst 2		–	0.43	0.30	–0.89
Analyst 3			–	–0.71	–0.59
Analyst 4				–	–0.14
Analyst 5					–

with the higher rating can then be recommended to her to read next. The predicted document ratings for Analyst 1 will be based on the ratings given to those documents by analysts who have expressed similar ratings to hers in the past on other documents that they all have rated. This prediction will require some metric for measuring the similarity between users. Common metrics, many of which are described in Herlocker et al. [9], include cosine similarity, Pearson's correlation coefficient, Spearman's rank correlation coefficient, or the mean squared error difference. We will use a variant of cosine similarity called adjusted cosine similarity.

If we represent the ratings of a particular analyst by a vector, the cosine similarity of two vectors (ratings of two different analysts) is equal to their dot product, divided by the product of their magnitudes:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

The cosine similarity of Analyst 1 and Analyst 2 for the first four documents can then be calculated as

$$sim(A1, A2) = \frac{(5*3) + (3*1) + (4*2) + (4*3)}{\sqrt{5^2 + 3^2 + 4^2 + 4^2} * \sqrt{3^2 + 1^2 + 2^2 + 3^2}} = 0.975$$

This calculation, however, does not take into account that analysts may generally give ratings in different ranges. For example, a particular analyst may tend to give ratings only in the range of 3 to 5 or may click more links than most analysts click on every page he visits (leading to higher implicit ratings). These types of factors may be accounted for by subtracting from each rating the average rating given by that user. Using this adjusted cosine similarity formula, we obtain the similarity scores shown in Table 2 in which a higher score indicates greater similarity.

Because we are basing Analyst 1's unknown ratings on the ratings of those who have similar rating histories, we may choose to use the ratings of only the  $k$  closest

neighbors or of those who have similarity scores above a certain threshold. In this case, we will use the ratings of only those who have Analyst 1 similarity scores greater than or equal to 0.5 (Analyst 2 and Analyst 4). The predicted rating of document  $d$  for user  $a$  thus becomes

$$r_{a,d} = \bar{r}_a + \frac{\sum_{b \in k} sim(a,b) * (r_{b,d} - \bar{r}_b)}{\sum_{b \in k} sim(a,b)}$$

From this equation, we obtain the following values for  $r_{AI,Doc5}$  and  $r_{AI,Doc6}$ :

$$r_{AI,Doc5} = 4 + \frac{0.85 * (3 - 2.17) + 0.71 * (4 - 3.33)}{0.85 + 0.71} = 4.75$$

$$r_{AI,Doc6} = 4 + \frac{0.85 * (1 - 2.17) + 0.71 * (2 - 3.33)}{0.85 + 0.71} = 2.97$$

Comparing the magnitudes of these predicted ratings reveals that Doc 5 should be recommended to Analyst 1 over Doc 6.

**Example Recommender System Using Topic Modeling**

A common form of a recommender system can use a mechanism of topic modeling to recommend objects such as new webpages, articles, or movies. The idea behind such a recommender system is that if a user is interested in a particular topic, she will be interested in other objects with the same topics. Similar to using techniques employed in a content-based recommender system, one may model a corpus of documents to find important topics. Once a topic is highlighted, a user is recommended other documents containing similar topics or terms. In this section, we will describe a simple but powerful way to perform topic modeling on very large datasets.

Non-negative matrix factorization (NMF), described by Lee and Seung [10], is a technique used to factorize a given matrix into two matrices, both of which only consist of non-negative elements. Multiplying these two matrices produces an approximation of the original

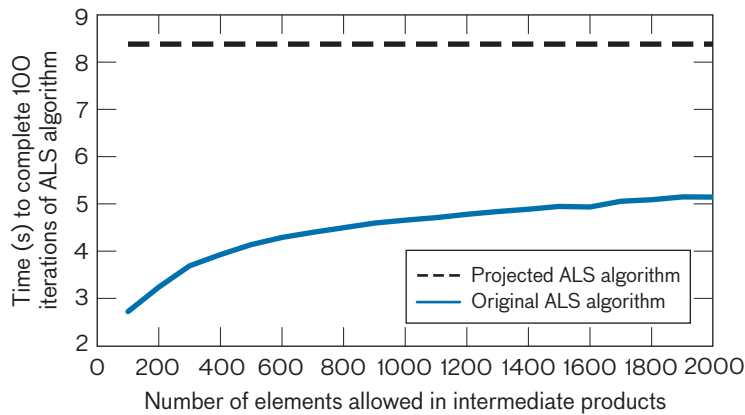
matrix. Consider a matrix  $A_{m \times n}$  to be factored into matrices  $W_{m \times k}$  and  $H_{k \times n}$ , where  $m$  corresponds to the number of rows of  $A$ ,  $n$  corresponds to the number of columns in  $A$ , and  $k$  corresponds to the number of topics. By definition,

$$A = W * H .$$

In the above factorization, the columns of  $W$  can be considered a basis for the matrix  $A$  with the rows of  $H$  being the associated weights needed to reconstruct  $A$ . A common method to solve this factorization problem is through the alternating least-squares (ALS) algorithm as described in Gadepally et al. [11]. However, one of the challenges in working with very large datasets is the inability to store intermediate products produced by the ALS algorithm. Very often, intermediate matrices created in each iteration of the ALS algorithm can be many times larger than the original dataset or available computational resources.

We have recently developed a new tool to perform NMF on large, sparse datasets. We refer to this tool as the projected ALS. In addition to removing non-negative elements in each iteration of the ALS algorithm, we can also enforce a particular sparsity level. This method has been shown to perform qualitatively as well as the original dense ALS algorithm. However, with this extra projection step that enforces sparsity, we are able to achieve much better computational performance as shown in Figure 2. By computing the matrix factorization through the projected ALS algorithm, we can determine a set of topics from a data corpus. We applied the projected ALS algorithm to a corpus of data collected from the popular social media site Twitter. We then found five topics from this dataset (Table 3). If a user highlights a certain tweet, a recommender system can then find other tweets that have keywords within the same topics of the selected tweet. For example, if the user highlights a tweet with the word “love,” the recommender system can suggest tweets with the hashtag “#PerksOfDatingMe” because these tweets are related via topic 2. This technique can be easily extended to the identification of malicious conversations about cyber attacks or other cyber events that often occur on social media sites such as Twitter; once the system discovers suspicious conversations, it can alert analysts to take a closer look at the suspect tweets.





**FIGURE 2.** The time in seconds taken for 100 iterations of the alternating least-squares (ALS) algorithm. The dashed black line corresponds to the time taken for the original ALS algorithm that yields dense intermediate products. The solid blue line corresponds to the time taken for 100 iterations of the projected ALS algorithm that enforces sparsity within each iteration. The large reduction in time is due to the computational efficiency of the projected ALS algorithm.

**Table 3. Topics in Twitter Posts Determined by Alternating Least-Squares Algorithm**

TOPIC 1 (TWEETS WITH TURKISH WORDS)	TOPIC 2 (TWEETS RELATED TO DATING)	TOPIC 3 (TWEETS RELATED TO ACOUSTIC GUITAR COMPETITION IN ATLANTA, GEORGIA)	TOPIC 4 (TWEETS WITH SPANISH WORDS)	TOPIC 5 (TWEETS WITH ENGLISH WORDS)
word :)	word #PerksOfDatingMe	word #5sosacousticATL	word con	word I'll
word @	word @	word #5sosfam	word creo	word I've
word Airport	word My	word #5sosgettoatlanta	word cuando	word If
word Hastanesi	word go	word @5SOS	word da	word Just
Word International	word love	word acoustic	word del	word Lol
word Kadiköy	word out	word atlanta?	word dormir	word My

**Recommender Systems and the Cyber Domain**

Recommender systems have the potential to greatly reduce the response time to cyber threats. In the cyber domain, it is very easy for analysts to be inundated with information. For example, the Target Corporation’s security breach was reported by the company’s security software but was ignored along with many false positive alerts [12]. In such enterprise environments, recommender systems can be valuable tools to filter and prioritize information that may be of interest to an analyst. Consider the common case of an information technology (IT) security team defending an organization against evolving cyber threats. As reported by

the 2015 Global Information Security Workforce Study [13], 62% of organizations claim that their information security teams are too small. These resource-constrained teams are also often responsible for paying attention to 100s of websites and blogs to look for information about publicly reported exploits. These teams may then have to turn to the National Vulnerability Database [14] to understand the impact of exploits to their organization. Finally, these teams may develop patches that are eventually deployed across the organization with varying levels of impact to the end users.

As of 2015, approximately 25 new vulnerabilities are reported per working day (as calculated by using

the number of Common Vulnerabilities and Exposures listed in the National Vulnerability Database [14] for the year 2015). In developing an appropriate response, the security team must weigh dozens of factors, such as the time since the vulnerability's discovery, severity of the exploit, existence of a patch, difficulty of deploying the patch, and impact of the patch on users. Recommender systems can provide a mechanism to greatly simplify this response process. A recommender system can automatically track 100s of sites, learn from past user behavior about important cyber security news items, and recommend them to the IT security team. Recommender systems can use prior information about the vulnerability's severity and impact to the user community to suggest a course of action for patching the vulnerability. For example, a recommender system may propose postponing the deployment of a minor vulnerability's patch that would cause a major impact for the user community, or the system may recommend immediate deployment of a major vulnerability's patch that would have minor user impact. Furthermore, recommender systems can be used to track anomalies across the network (such as unpatched systems or systems exhibiting behavior very different from that of others on a network) to allow the limited resources of the IT security team to quickly address potentially important problems rather than being inundated with regular traffic.

### **Specific Concerns of the Department of Defense and Intelligence Community**

While recommender systems have reached maturity in the commercial world, there are many challenges in directly applying these systems to DoD and IC problems. Commercial and government entities both have the need to collect, store, and process a large amount of high-dimensional data. However, government applications have certain traits that make utilizing traditional methods to produce actionable intelligence more difficult. Some of these differences are shown in Table 4.

The first difference concerns the lack of ground truth and the difficulty in quantifying success for DoD applications. In industry, success tends to be measured by a concrete action, such as a sale of a product or a click on a webpage. In DoD and IC applications, however, the desired measure of effectiveness is whether or not an action will lead to a greater probability of mission

success. Because this metric is speculative, it is much more difficult to measure than the commercial standard of profitability.

Compared to industry applications, government applications typically carry much more extreme consequences for false automatic calculations that lead to suboptimal decisions. Once again, the magnitudes of these consequences are harder to quantify. Dollars provide an obvious surrogate for risk in a commercial setting, but there is no clear, established metric for measuring operational readiness. The lack of such a metric makes it difficult to determine whether government organizations should use a particular piece of technology in support of their mission.

Similar to commercial cyber security applications, government applications exist in a space where the adversary is continually evolving. Yet, the current architectural and political landscape found in most government organizations necessitates that analytics are developed, deployed, and re-engineered over a much longer time scale than industrial applications typically employ. Thus, government organizations experience fewer opportunities to make incremental improvements to the underlying analytics.

Differences in the skill levels of users affect the design and value of recommender systems. Users of recommender systems in the DoD and IC will likely be experts in their fields who engage with these systems daily. Such familiarity with the system may allow for more capable and complex functionality to be utilized. Perhaps, more importantly, the inclusion of experts in this human-in-the-loop process may lead to a different balance of autonomy. Recommender systems may need to be capable of making the reasons behind their recommendations transparent in order to gain the confidence of experts who are making high-stakes decisions. For example, a system may provide the end user with a confidence measure (such as probability) associated with each recommendation.

Another significant difference between big data applications used by commercial and government groups is that a commercial entity generally controls its data sources and approaches the data with specific goals and questions while government groups usually do not. For instance, Google may create a new feature on Google+ to obtain a different type of information from its user base to better its advertising services. However, government agencies, which usually do not have collection authority over the data they use, have limited control over designing data collection

**Table 4. Comparison of Commercial Applications to DoD Applications**

COMMERCIAL APPLICATIONS	DOD APPLICATIONS
High dimensionality of data	High dimensionality of data
Large volume of data	Large volume of data
Known truth; easier to quantify success	Unknown truth; difficult to quantify success
Mild consequences of decisions	Large consequences of decisions
Past is representative of future	Past does not represent future
Continual development and improvement	Deployment; long durations between improvements
Average or untrained users	Expert or trained users

paradigms; even those agencies with collection authority are often bound by regulations that restrict their ability to deploy sensors that can collect the specific data they desire.

Finally, commercial applications are often designed to learn from millions to billions of users whereas DoD and IC applications may only have 100s to 1000s of users whose behaviors can be used to model recommender systems. Also, very often, commercial entities employ user agreements to determine data collection and usage whereas government organizations may not be able to readily access data without the help of law enforcement or legal statutes.

### Recommender Systems Applied to Lincoln Laboratory Programs

Lincoln Laboratory has a rich history in developing decision support systems. Over the past five years, these systems have been incorporating recommender system concepts and technologies. In this section, we summarize past, current, and future Laboratory programs that incorporate recommender systems. The work conducted in these programs can inform research into systems that improve cyber security.

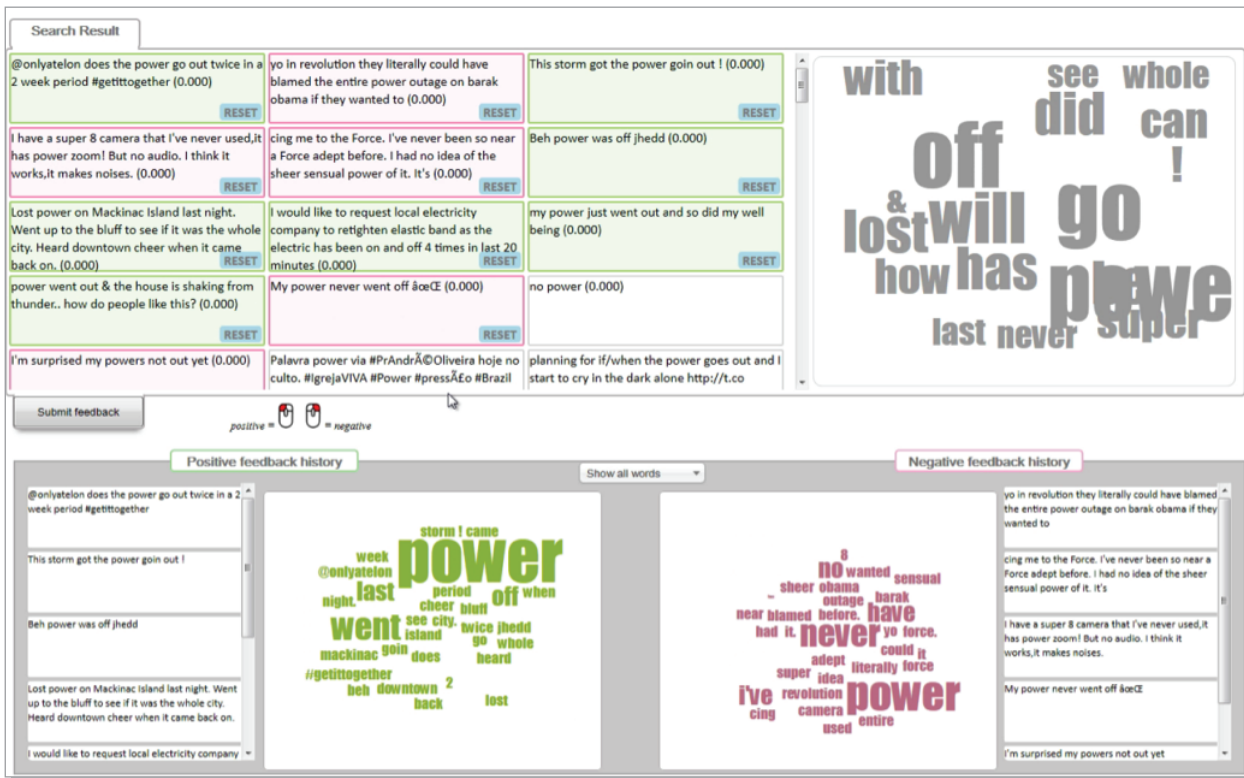
### Dynamic Customization of Content Filtering

The objective of Dynamic Customization of Content Filtering (DCCF) is to allow an analyst to perform on-the-fly customization of content filtering (for example, open-source social media data mining) on the basis of simple relevance feedback acquired during the inspection of filtered content (Figure 3).

First, the analyst sets the parameters of an initial data-stream filter (e.g., keywords, geographical area, time interval) to mine for content of interest. Typically, as when keyword filters are used on social media data, this approach will lead to a mixture of relevant content embedded within various types of irrelevant content. While reviewing the content, the analyst provides simple binary feedback (indicating relevance or irrelevance) as desired and submits this feedback to the system. The DCCF model uses this feedback to create a secondary filter to remove irrelevant data that passes through the first filter. The creation of this secondary filter is based on a broad set of text- and image-derived feature spaces (i.e., characteristics of a general dataset; a dataset of a network's cyber attacks may include feature spaces such as date, time, type) coupled with aggressive feature space downselection and classifier training so that the model is suited to potentially diverse content-filtering needs. The DCCF model is generated on the fly (during analyst use) every time new feedback is submitted, thus improving content filtering as the user increasingly interacts with DCCF. The DCCF tool may be considered a recommender system because it pushes out filtered content that is based on earlier user-specific feedback.

### Delve

The goal of Delve is to develop an approach for recommending documents to analysts who are answering broad, complex questions. This task is particularly suited for recommender systems because analysts are often uncertain as to what relevant information may be available to them



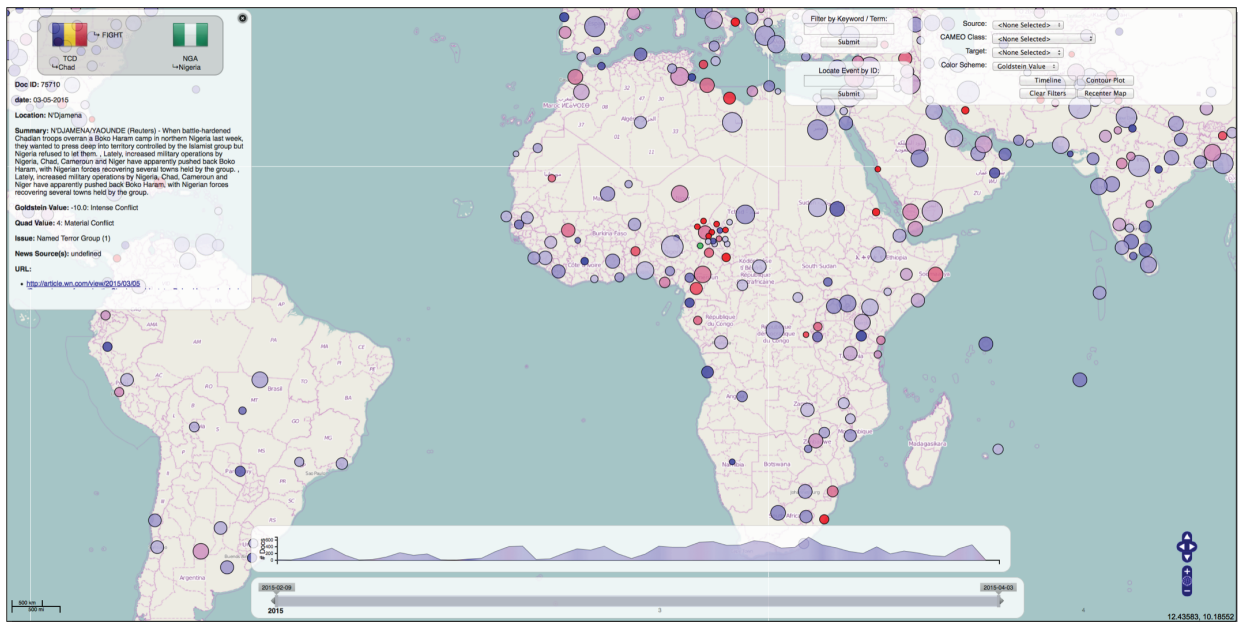
**FIGURE 3.** The Dynamic Customization of Content Filtering (DCCF) uses analyst feedback to perform on-the-fly customization of content filtering. A user first sets parameters such as keywords, geographical area, or time interval. The DCCF model will automatically filter content on the basis of these keywords to expedite retrieval of useful content. The word cloud on the top right corresponds to all retrieved results from the key word filter. The bottom word clouds correspond to results from the secondary filters.

and therefore are ill-equipped to find all the information that they need via precise queries only. The Delve system employs a hybrid recommender that calculates both individual document characteristics (e.g., word count, number of entities) and collective browsing behavior (e.g., identification of articles that tend to co-occur or follow others in a browsing path). Using these calculations along with dimensionality-reduction techniques, Delve significantly outperforms baseline approaches, such as using only webpage attributes or term frequency-inverse document frequency (TF-IDF), for recommending additional documents of interest, given an initial document selected by the analyst.

### Global Pattern Search at Scale

The Global Pattern Search at Scale (GPSS) is a scalable visual analytics platform to support the analysis of unstructured geospatial intelligence. With GPSS,

analysts can interactively explore the document corpus at multiple geospatial resolutions, identifying patterns that cut across various data dimensions, and can uncover key events in both space and time. The tool includes an interactive visualization featuring a map overlaid with document clusters and events, search and filtering options, a timeline, or a word cloud (Figure 4). As an information filtering tool, GPSS provides detail on demand. However, in its current form, GPSS does not “push” new intelligence information to the analyst. In the coming year, the GPSS team plans to augment their tool with a recommender system that will profile user activity and suggest new documents of interest after periods of inactivity, similar to the way that advanced news websites such as Google News will suggest articles related to topics, locations, or stories in which a specific reader has expressed past interest.



**FIGURE 4.** The Global Pattern Search at Scale (GPSS) platform enables interactive exploration of intelligence information by topic, time, and location. The GPSS system provides users with quick geospatial visualization about documents. In the display above, a user can search for terms, and the circles of different sizes and shades indicate, respectively, the prevalence of articles in a particular geospatial location (color spectrum runs from red for a region of conflict to lavender for a region of cooperation) at a particular time.

### Covert or Anomalous Network Discovery and Detection

Networks are often used to describe relations or interactions between individuals, systems, or other entities via graphical models. The Covert or Anomalous Network Discovery and Detection (CANDiD) program aims to develop the mathematical understanding for constructing operationally relevant networks, detecting important sub-graphs of these networks, and inferring and influencing the properties of select vertices in the network. Networks of interest often arise from large collections of complementary, redundant, and potentially noisy relational data sources, introducing challenges both in terms of algorithmic scalability and algorithmic accuracy. Through the CANDiD program, we are currently looking into applying a recommender system perspective to the problem of filtering and personalizing the multisource, noisy data used to construct and estimate the network of interest.

### Adaptive, Reinforced, Interactive Visual Analytics

The goal of the Adaptive, Reinforced, Interactive Visual Analytics (ARIVA) program is to identify important

information that aligns with analyst-provided feedback to better facilitate algorithmic-aided exploration of complex data for evolving open-ended missions, such as deterring cyber threats. User-provided feedback, in the form of similarity and dissimilarity assessments between pairs of data points, is utilized to perform data preprocessing via feature selection and transformation. When feedback-aligned data embeddings are accurately identified, common exploration analytics, such as data clustering, nearest-neighbor classification, and information retrieval techniques, show algorithmic performance improvement and produce results that are grounded in an analyst's preferences, understanding of mission goals, and expertise in a given domain. The improvement of retrieval algorithms in feedback-aligned data spaces suggests that recommender systems can augment tools like ARIVA by utilizing the similarity or proximity of data points in the learned data embedding. The use of explicit pairwise similarity and dissimilarity constraints allows this application to avoid problems commonly found in recommendation engines whose limited feedback often leads to a reduction in recommender system performance.

### Structured Knowledge Space

Structured Knowledge Space (SKS) is an end-to-end software system that combines information extraction, information retrieval, and natural language processing to intelligently explore a corpus of unstructured documents, such as intelligence reports of cyber threats. The SKS suite of tools extracts entities and creates structured metadata for each document to improve its searchability (Figure 5). With this metadata, analysts can find all documents that refer to a single organization or person (even when that entity has several aliases or variations), that contain a geospatial reference within a certain distance of a location, or that reference a time within a specified date range. Currently, SKS operates under a “pull” rather than a “push” paradigm (i.e., the user searches and browses rather than the system making recommendations). However, there are multiple ways in which recommender system concepts can be utilized to further enhance SKS. One enhancement would be to recommend new articles on the basis of past searches performed (e.g., “This article was recommended to you because of your interest in phishing attacks on enterprise networks.”).

Another enhancement would be to guide novice analysts’ searches by using the search paths that more experienced analysts have taken.

### Cyber Human Language Technology Analysis, Reasoning, and Inference for Online Threats

Through the Cyber Human Language Technology (HLT) Analysis, Reasoning, and Inference for Online Threats (CHARIOT) program, we are developing an interactive filtration system to automatically identify documents that are relevant to analysts’ current investigations (Figure 6). With CHARIOT, analysts are presented with online discussions concerning cyber attack methods, defense strategies, and tools’ effectiveness through the automated examination and classification of forum threads. CHARIOT leverages techniques such as topic classification, entity recognition, and sentiment analysis (i.e., opinion mining) to separate malicious cyber discussions from irrelevant discussions. The “Finding Malicious Cyber Discussions in Social Media” article in this issue discusses the CHARIOT program in further detail.

The screenshot shows the SKS search interface with several key components highlighted by red boxes and callouts:

- Search box:** A standard Google-like search operator is located at the top left.
- Advanced search:** Search filters include 'Search by Reported Geos' (with a map icon), 'Search by Reported Dates' (date range), 'Type' (DOC, PPT, XLS, PDF, TXT), and 'Search by Ingest Date'.
- Facet categories:** A sidebar on the left lists entities by document count under categories like PEOPLE, LOCATIONS, ORGANIZATIONS, COVER TERMS, CODE WORDS, and SELECTOR A.
- Search results:** The main area displays results 1-10 of 340. Each result includes a document title, date, and a preview snippet with highlighted terms. A 'Geo' link is present for each result.
- Similar documents:** A 'Similar' link is provided for each result to find documents with similar text.
- Preview and download:** A 'Download' link is available for each document preview.
- Plot geocoordinates:** A 'Geo' link is used to plot geocoordinates for one or all search results on a visualization tool like Google Earth.

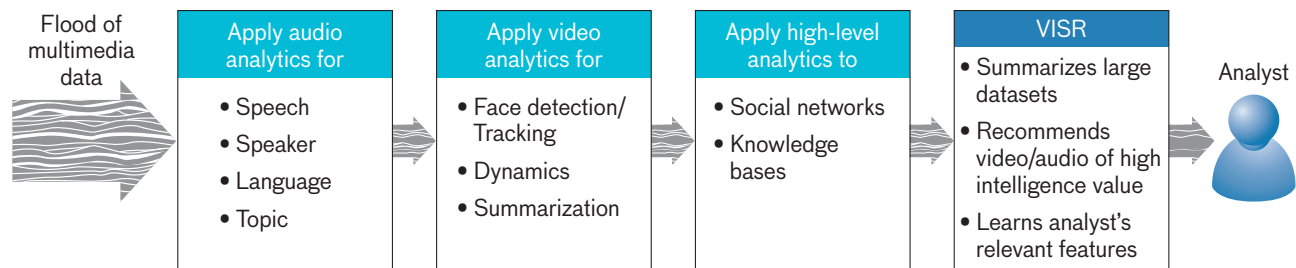
**FIGURE 5.** The Structured Knowledge Space search page provides diverse, useful information for the exploration of a corpus of unstructured documents.

### Visualization, Summarization, and Recommendation for Multimedia

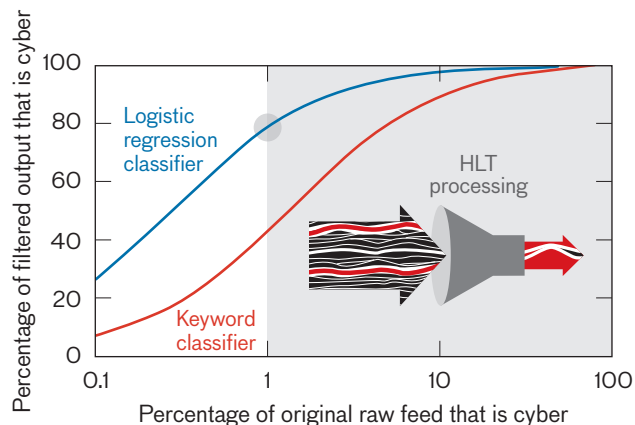
The goal of the Visualization, Summarization, and Recommendation (VISR) for Multimedia program is to develop tools to allow analysts to effectively explore large multimedia data sources. The program builds on previous Lincoln Laboratory work in text analytics by expanding to multimedia data, especially audio and video, with the addition of a recommendation component (Figure 7). This recommender system utilizes a user’s ongoing work to identify other information of interest. For example, it may suggest videos of likely interest to an analyst on the basis of his or her current and past searches.

### XDATA

The Defense Advanced Research Projects Agency’s (DARPA) XDATA program aims to meet the challenges of big data analytics and visualization by developing computational techniques and software tools for processing and analyzing large, noisy, and incomplete data. For scalable analytics, this work includes research into distributed databases, statistical sampling methods, and new algorithmic advances to lower the computational complexity of pattern matching. For information visualization, this effort is focusing on the development of web-based human-computer interaction tools that factor computation between the client and the server and that are built from an open code base to enable rapid customization of tools to different missions. The XDATA program is investigating software that can efficiently fuse, analyze, and disseminate the massive volume of data these tools produce.



**FIGURE 7.** The Visualization, Summarization, and Recommendation (VISR) for Multimedia system takes a collection of multimedia audio and video data as an input. Notionally, three analytic processes at the individual item level process the data. First, standard speech processing is applied—speech, speaker, language, and topic recognition. Second, video analytics are applied to find faces and summarize object content. Third, high-level analytics are applied to find links between individual items. These links are user selectable. Examples of links include common faces, similar object content, or similar audio content. The VISR interface integrates all this functionality to display data to an analyst in a structured form for navigation and triage.



**FIGURE 6.** The goal of CHARIOT is to filter social media discussions to find cyber content. This figure contrasts a simple keyword classifier that detects cyber discussions via keywords or phrases with the CHARIOT-implemented logistic regression classifier. For an incoming data stream containing 1% cyber content, the CHARIOT logistic regression classifier can output a data stream containing 78% cyber content (compared to 43% for the keyword classifier).

Lincoln Laboratory’s approach for the DARPA XDATA program is to provide key enabling technologies—including those for natural language processing, topic clustering, and text language identification—to extract information from structured, semistructured, and unstructured text, speech, image, and video data. This information is then used by the Laboratory and our partners for upstream (later in the development pipeline) analytics and visualization. The Laboratory has developed several analytics and user-interface technologies for graph query by example, entity

disambiguation, community detection, and correlation of similar network graph portions. Many of these enabling component technologies for recommender systems have been made publicly available via DARPA's Open Catalog of DARPA-sponsored software and peer-reviewed publications (<http://opencatalog.darpa.mil/>).

### Future Work

Before the DoD and IC can employ recommender systems in operational settings, many technical and sociotechnical challenges must be overcome. We have identified six specific challenges that future work at Lincoln Laboratory and within the DoD and IC could productively address:

- *Establishing user trust.* The potentially serious consequences of decisions made via DoD applications necessitate a level of user trust in the recommender system. One way to increase trust is to enhance the interpretability or transparency of results, using algorithms that enable explanations to be given for why a particular object has been recommended, as discussed in O'Donovan and Smyth. [15]. Another approach to fostering trust in a recommender system could be to verify the reliability of the source of data used in the development of the system's model by tracking the data's provenance, i.e., its origins and route of transfer.
- *Preserving privacy and security.* While there is certainly a need to make the recommendations of a recommender system transparent, there is simultaneously a potentially conflicting need to ensure the privacy of users, as described in Avesani et al. [16] and Brekovsky et al. [17]. A system that relies on tracking user history—sometimes in great detail (e.g., purchase history, browsing history, eye tracking)—has the potential to be misused by users to learn nonpublic details about other users if security precautions are not taken. Similarly, the security of the system may be at risk if individual users are able to reverse engineer the system to learn, for example, that submitting a certain number of specific inputs can ensure that another user will see a given output. Cryptographic techniques, such as those described by Gadepally et al. [18] and Shen et al. [19], may prove a useful means for building into recommender systems guarantees that prevent information from being discoverable by unauthorized users.
- *Adapting to user environment.* The types of users and usage contexts of recommender systems can reasonably be expected to vary significantly between commercial and defense applications. Whereas commercial systems tend to be used in environments that demand little user concentration, have few time constraints, and assume minimal user experience with the system, defense systems have the potential to be used in environments that require high concentration from users, adhere to strict deadlines, and employ operators who have been trained to engage with the system on an intricate level. Research into how existing mechanisms may be modified to address DoD and IC constraints or to exploit the capabilities of their personnel and systems seems prudent.
- *Developing multilevel metrics.* Of the many ways to assess the value of a recommender system, the majority of these assessments pertain to the perceived quality of recommendations. Some of these metrics are described by Gunawardana and Shani [20]. In addition to these recommendation-level metrics, however, there is also a need for system-level and user-level metrics. System-level metrics may reflect the measured time savings of a decision process or a change in the percentage of documents read that are considered relevant. User-level metrics consider the users' experiences with the system—how are concentration, decision fatigue, or confidence in decisions affected when users interact with the system? This area may overlap to some extent with the requirement of establishing user trust.
- *Promoting system extensibility.* While the core algorithms of recommender systems are often made public via publications and presentations, deployment and maintenance details are rarely discussed. From an institutional standpoint, it is important for the DoD and IC and their partners to understand how transferable the developed technology in this area will be from one domain or mission to another. It would be valuable to understand which technologies require domain-specific tuning and which ones can be rapidly deployed in new scenarios with little modification. Determining which pieces may be modularized for reapplication or redeployment could lead to improved cost estimates over the life cycle of the developed technology. The development of a standardized recommender system application program interface could



lead to users' ability to immediately and easily interact with data in multiple forms on a variety of databases.

- *Developing partners in academia and industry.* The future research areas we have described focus on the specific needs of the DoD and IC. However, work in recommender systems encompasses a number of fields, including machine learning, big data analytics, and user experience, and many individuals in academia and industry are also conducting research in these fields. These researchers could partner with us to provide innovative ways to advance the role of recommender systems in various domains.

Recommender systems could have a significant impact in defense and intelligence applications. With the ability to learn from user behavior and push suggestions to users, they have the potential in mission scenarios to shift computational support from being reactive to being predictive. Recommender system technology has been advanced substantially in recent years by commercial entities, but some future work will be required to adapt these technologies for use in the defense domain, where requirements and objectives differ from those of commercial applications.

### Acknowledgments

We would like to thank the Lincoln Laboratory staff members who contributed to this work: Paul Breimyer, Paul Burke, Rajmonda Caceres, R. Jordan Crouser, Matthew Daggett, Jack Fleischman, David Weller-Fahy, Vitaliy Gleyzer, Robert Hall, Andrew Heier, Stephen Kelley, David Martinez, Benjamin Miller, Paul Monticciolo, Kenneth Senne, Danelle Shah, Mischa Shattuck, Olga Simek, Steven Smith, William Streilein, Jason Thornton, Michael Winterrose, and Michael Yee. We would also like to thank research librarian Robert Hall for his help with the literature review and Marc Bernstein for supporting this work. A special thanks to Ariana Tantillo and Dorothy Ryan for their help in editing the article. ■

### References

1. P. Resnick and H.R. Varian, "Recommender Systems," *Communications of the ACM*, vol. 40, no. 3, 1997, pp. 56–58.
2. G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, 2005, pp. 734–749.
3. S. Bailey, "The Intelligence Cycle and Human Language Technology," HLT Return on Investment Working Group, internal document, July 2015.
4. K.B. Lyons, "A Recommender System in the Cyber Defense Domain," master's thesis no. AFIT-ENG-14-M-49, Air Force Institute of Technology Graduate School of Engineering and Management, Wright-Patterson Air Force Base, 2014.
5. P. Thompson, "Weak Models for Insider Threat Detection," *Proceedings of SPIE*, vol. 5403: "Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense," 2004, pp. 40–48.
6. T.A. Lewis, "An Artificial Neural Network-Based Decision Support System for Integrated Network Security," master's thesis no. AFIT-ENG-T-14-S-09, Air Force Institute of Technology Graduate School of Engineering and Management, Wright-Patterson Air Force Base, 2014.
7. C.J. Wood, "What Friends Are For: Collaborative Intelligence Analysis and Search," master's thesis, Naval Postgraduate School, 2014.
8. D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. New York: Cambridge University Press, 2010.
9. J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, 2004, pp. 5–53.
10. D.D. Lee and H.S. Seung, "Algorithms for Non-negative Matrix Factorization," in *Advances in Neural Information Processing Systems: The Proceedings of the 2000 Neural Information Processing Systems Conference*, T.K. Leen, T.G. Dietterich, and V. Tresp, eds., 2001, available at <http://papers.nips.cc/book/advances-in-neural-information-processing-systems-13-2000>.
11. V. Gadepally, J. Bolewski, D. Hook, D. Hutchinson, B. Miller, and J. Kepner, "Graphulo: Linear Algebra Graph Kernels for NoSQL Databases," 29th IEEE International Parallel and Distributed Processing Symposium Workshops, Hyderabad, India, 25–29 May 2015.
12. R. Lemos, "Survey Says Security Products Waste Our Time," *arstechnica*, 16 Jan. 2015, <http://arstechnica.com/security/2015/01/survey-says-security-products-waste-our-time/>.
13. M. Suby and F. Dickson, "The 2015 (ISC)<sup>2</sup> [Information Systems Security Certification Consortium] Global Information Security Workforce Study," Frost and Sullivan White Papers, 16 April 2015, available at <https://www.isc-2cares.org/IndustryResearch/GISWS/>.
14. National Institute of Standards and Technology, National Vulnerability Database, available at <https://nvd.nist.gov/>.
15. J. O'Donovan and B. Smyth, "Trust in Recommender Systems," *Proceedings of the 10th International Conference on Intelligent User Interfaces*, 2005, pp. 167–174.
16. P. Avesani, P. Massa, and R. Tiella, "A Trust-Enhanced Recommender System Application: Moleskiing," *Proceedings of the 2005 ACM Symposium on Applied Computing*, 2005, pp. 1589–1593.

17. S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci, "Enhancing Privacy and Preserving Accuracy of a Distributed Collaborative Filtering," *Proceedings of the 2007 ACM Conference on Recommender Systems*, 2007, pp. 9–16.
18. V. Gadepally, B. Hancock, B. Kaiser, J. Kepner, P. Michaleas, M. Varia, and A. Yerukhimovich, "Improving the Veracity of Homeland Security Big Data through Computing on Masked Data," 2015 IEEE International Symposium on Technologies for Homeland Security, Waltham, Mass., 14–16 Apr. 2015.
19. E. Shen, M. Varia, R.K. Cunningham, and W.K. Vesey, "Cryptographically Secure Computation," *Computer*, vol. 48, no. 4, 2015, pp. 78–81.
20. A. Gunawardana and G. Shani, "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks," *Journal of Machine Learning Research*, vol. 10, 2009, pp. 2935–2962.

**About the Authors**



**Vijay N. Gadepally** is a technical staff member in Lincoln Laboratory's Secure Resilient Systems and Technology Group and a researcher at the MIT Computer Science and Artificial Intelligence Laboratory. His research is focused on big data and Internet of Things systems, machine learning, high-performance

computing, advanced database technologies, and pattern recognition. Prior to joining Lincoln Laboratory in 2013, he worked as a postgraduate intern with the Raytheon Company, a visiting scholar with the Rensselaer Polytechnic Institute, and a student intern with the Indian Institute of Technology. He holds a bachelor's degree in electrical engineering from the Indian Institute of Technology in Kanpur, and master's and doctoral degrees in electrical and computer engineering from The Ohio State University. His doctoral dissertation on signal processing focused on developing mathematical models to accurately estimate and predict driver behavior for autonomous vehicle applications.



**Braden J. Hancock** is a computer science doctoral student at Stanford University and a former 2014 and 2015 summer intern in Lincoln Laboratory's Cyber Security and Information Sciences Division. His current research focus is on the development of machine learning techniques for automatically

converting unstructured data into structured information for integration and inference tasks. Between his time at Stanford and as an undergraduate in mechanical engineering at Brigham Young University, he has received numerous awards, including the National Science Foundation (NSF) Graduate Research Fellowship, National Defense Science and Engineering Graduate Fellowship (declined to accept NSF fellowship), Phi Kappa Phi Marcus L. Urann Fellowship, Finch Family Fellowship (Stanford), Barry M. Goldwater Scholarship, American Society

of Mechanical Engineers Kenneth Andrew Roe Scholarship, and American Institute of Aeronautics and Astronautics Vicki and George Muellner Scholarship. Prior to his internships at Lincoln Laboratory, he interned at the Johns Hopkins University Human Language Technology Center of Excellence and the Air Force Research Laboratory.



**Kara B. Greenfield** is a technical staff member in the Human Language Technology Group. Her work focuses on research in named-entity recognition, social network analysis, and visual analytics. Prior to joining Lincoln Laboratory, she interned at CA Technologies and Pegasystems, collaborated with HP Labs

and the Hungarian Academy of Sciences, and served as a teaching assistant at Worcester Polytechnic Institute (WPI), where she won the Teaching Assistant of the Year Award from the Department of Mathematical Sciences. She holds bachelor's degrees in mathematics and computer science and a master's degree in industrial mathematics from WPI. Her master's research focused on utilizing crowdsourcing to develop a gold-standard-quality corpus for named-entity recognition. She is a member of two honor societies: Pi Mu Epsilon and Upsilon Pi Epsilon.



**Joseph P. Campbell** is the associate leader of Lincoln Laboratory's Human Language Technology Group, where he directs the group's research in speech, speaker, language, and dialect recognition; word and topic spotting; speech and audio enhancement; speech coding; text processing; natural language processing;

machine translation of speech and text; information retrieval; extraction of entities, links, and events; cross-language information retrieval; multimedia recognition techniques, including both voice and face recognition for biometrics applications; advanced analytics for analyzing social networks on the basis of speech, text, video, and network communications and activities; and recommender systems. He specializes in the following for government applications: research, development, evaluation, and transfer of speaker recognition technologies; design of speaker recognition and biometrics evaluations; design of corpora to support those evaluations; and development and evaluation of biometrics technologies and systems. He joined Lincoln Laboratory in 2001 as a senior staff member after serving 22 years at the National Security Agency. He was an IEEE Distinguished Lecturer and is an IEEE Fellow. He earned bachelor's, master's, and doctoral degrees in electrical engineering from Rensselaer Polytechnic Institute, The Johns Hopkins University, and Oklahoma State University, respectively.



**William M. Campbell** is a senior technical staff member of the Human Language Technology Group. He provides leadership and technical contributions in the areas of speech processing, machine learning, and social networks. His speech processing work has resulted in advances in speaker and language recognition, including the

development of algorithms that have been widely cited in published papers and implemented. He has made numerous contributions in social network graph analysis as it relates to simulation of social networks; machine learning involving social networks; and construction of networks from multimedia content. Prior to joining Lincoln Laboratory in 2002, he worked on speech processing and communication systems at Motorola. An active contributor to the speech and machine-learning research community, he has served as reviewer and scientific committee member for several conferences: IEEE Odyssey: The Speaker and Language Recognition Workshop; Annual Conference on Neural Information Processing Systems; International Conference on Acoustics, Speech, and Signal Processing; INTERSPEECH; and IEEE Spoken Language Technology Workshop. He is the author of more than 100 peer-reviewed papers, including multiple book chapters; a recipient of the Motorola Distinguished Innovator Award; the holder of 14 patents; and a senior member of the IEEE. He received three bachelor's degrees—in computer science, electrical engineering, and mathematics—from South Dakota School of Mines and Technology and master's and doctoral degrees in applied mathematics from Cornell University.



**Albert I. Reuther** is the assistant group leader of the Secure Resilient Systems and Technology Group, where he leads research teams in graph processing and analytics, high-performance parallel and distributed computing, machine learning, and novel computer architectures. He joined Lincoln Laboratory in 2001. His

publications include a number of papers and book chapters on interactive, on-demand supercomputing, dynamic computational and database prototyping environments, parallel and distributed signal processing, and supercomputing scheduling algorithms. He received the 2005 Eaton Award in Design Excellence from Purdue University's School of Electrical and Computer Engineering, was an Intel Foundation Graduate Fellowship recipient, and is a member of the IEEE and Association for Computing Machinery (ACM) professional societies. He earned a dual bachelor's degree in computer and electrical engineering, a master's degree in electrical engineering, and a doctoral degree in electrical and computer engineering, all from Purdue University, and a master's of business administration degree from the Collège des Ingénieurs in Paris, France, and Stuttgart, Germany.

# Repeatable Reverse Engineering with the Platform for Architecture-Neutral Dynamic Analysis

**Ryan J. Whelan, Timothy R. Leek, Joshua E. Hodosh,  
Patrick A. Hulin, and Brendan Dolan-Gavitt**

Many problems brought on by faulty or malicious software code can be diagnosed through a reverse engineering technique known as dynamic analysis, in which analysts study software as it executes. Researchers at Lincoln Laboratory developed the Platform for Architecture-Neutral Dynamic Analysis to facilitate analyses that lead to profound insight into how software behaves. This tool was recognized with a 2015 R&D 100 Award for being one of the year's 100 most innovative technologies.



**Billions of lines of computer code direct** the flow of information that drives the world's activities. This vast amount of code powers software programs that instruct systems to perform tasks as commonplace as word processing and as specialized as analyzing DNA sequence data. However, lurking within this benign software are critical vulnerabilities that cyber criminals exploit to steal or corrupt information. In addition, as new software versions, capabilities, and operating systems are introduced to the marketplace, older software code often becomes incompatible with new technology, rendering the software either ineffective or completely unusable. Although the U.S. government and businesses annually spend millions of dollars to recover from attacks that inject malicious software, or malware, into their computer systems and to keep their software operational, more effective analysis capabilities are still needed to enable rapid, successful diagnosis and resolution of software problems. Lincoln Laboratory researchers have created an open-source tool, the Platform for Architecture-Neutral Dynamic Analysis (PANDA), for analysts to use to quickly develop instrumentation that helps answer complex questions about software and that informs appropriate responses to malware intrusions.

## Reverse Engineering

PANDA facilitates an analysis technique known as reverse engineering (RE), i.e., the process of analyzing a program's code to discover its undocumented internal principles. By closely inspecting the binary code that runs a piece of software, an analyst can study how the program has been constructed to perform its operations. Reverse engineering is frequently employed to enable legacy code to continue functioning, to identify vulnerabilities in software, and to understand the true purpose and actions of a software program.

It is common for legacy code to stop working as the software ecosystem surrounding it evolves. When that failure happens, and when corporate support for the old program has also long terminated, RE is the most cost-effective avenue to revive the functionality of the software. Using RE, analysts can discover the inputs and outputs to, and the dependencies and requirements of, a software program so that they can then develop appropriate fixes that allow the old code to run in a more modern environment.

Accurately identifying vulnerabilities is usually impossible without detailed RE knowledge. Analysts might be able to observe that a software bug exists, but being able to determine if it is exploitable, and therefore a critical vulnerability, is a much more difficult problem. Part of the solution to this problem is the determination of which specific parts of the program are questionable; often, source code is not available to help make this determination. Thus, without either performing RE or making use of the RE efforts of others, it is difficult to discriminate between unimportant bugs and serious vulnerabilities.

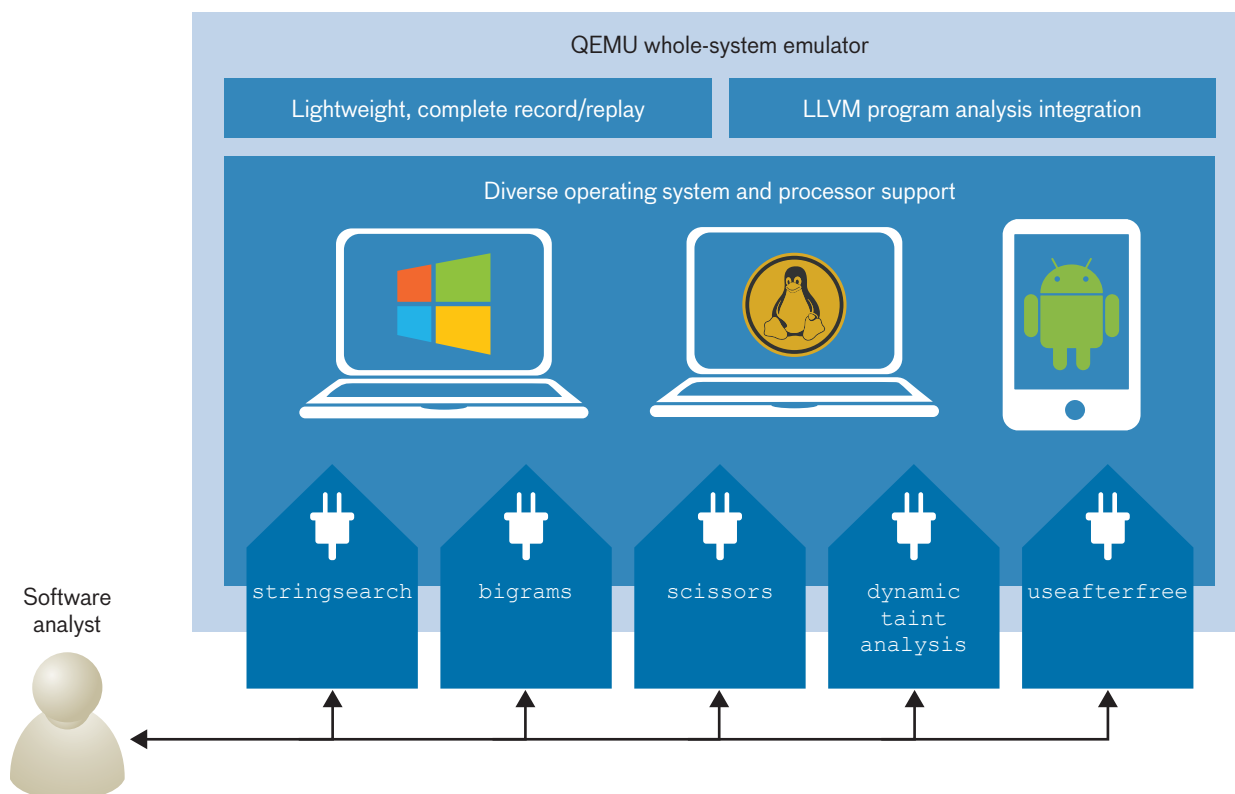
Vetting software to determine if it does what it is purported to do and nothing else is an important, complicated task. When the code is believed to be malware, this determination is usually obvious. However, we believe there is an increasingly fine distinction between malware and misbehaving code. Consider a program written by a legitimate, large U.S. company, and imagine that its code performs a host of unintended malicious actions, such as accessing personal information or modifying system settings. None of this behavior is indicated in the documentation or advertising literature, nor is it clearly essential for the primary purpose of the software. How is this code functionally distinct from malware? This scenario is not simply a thought experiment: in 2005, Mark Russinovich, the cofounder of Winternals Software,

discovered that audio CDs produced by Sony BMG Music Entertainment were installing a rootkit onto millions of computers [1]. The Sony rootkit recorded information about users' computers to send back to Sony and hid every file on users' systems with a certain prefix; worse, Sony's uninstaller allowed any webpage to download and execute arbitrary code [2].

## Reverse Engineering Through Dynamic Analysis

One approach to RE is *static analysis*. In this approach, analysts use tools such as disassemblers and decompilers to translate binary code into a form more easily read. Humans painstakingly navigate these representations, adding extensive annotations to ultimately reassemble a picture of how code and data operate at various levels of abstraction. *Dynamic analysis* is another approach to RE. In a dynamic analysis, while software executes on the system, analysts observe its behavior.

PANDA is fundamentally a dynamic analysis tool that can help analysts gain deep insight into software code by observing the code's behaviors across all levels of the operating system. Figure 1 provides a high-level overview of PANDA, and its use is depicted in Figure 2. First, an analyst captures a recording of some whole-system execution that he or she wishes to understand thoroughly. Then, the analyst writes analysis code in the form of plugins, which are modules that add specific capabilities to the software. Plugins collect data and consult or control other plugins. They are typically written quickly and iteratively, running a replay of the previously gathered recording over and over to construct a deeper understanding of the important aspects of system execution. For example, an initial plugin might just get a rough outline of what processes execute on the system and when key operating system events happen during the replay. A second analysis pass over the replay might focus in on the activity of a particular program or a portion of the replay. Further iterations over the replay might be more complex and allow analysts to selectively label interesting data and track those data as they flow around the system; this process is metaphorically similar to a positron emission tomography (PET) scan [3], which provides diagnostic scans of organs and tissues by tracing a radioactive substance as it travels through the body. We have found that this workflow powerfully enhances RE, as it enables analysts to iteratively build knowledge about dynamic software executions.



**FIGURE 1.** This high-level overview of PANDA shows its key features. PANDA has the ability to efficiently record and replay whole-system executions; the ability to support diverse operating systems, such as Windows, Linux, and Android; and a modular software design in which each analysis can be implemented as a plugin, and the plugins can be used in conjunction with one another. Plugins can execute a number of diverse tasks according to how they are programmed by the analyst. For example, they can track information about which processes are executing, enable dynamic searching of data in the system, perform automated web-traffic decryption for certain algorithms, and perform detailed exploit analysis.

### PANDA System

PANDA is largely based upon the open-source whole-system emulator known as QEMU (Quick Emulator). QEMU is a robust platform that uses binary translation to support multiple processor architectures. Utilizing QEMU allows us to emulate an entire Windows or Linux desktop, an Android phone, and other embedded systems.

PANDA has four key features: the ability to record and replay entire software executions, an extensible plugin architecture, the ability to extend software analyses across multiple processor architectures, and the ability to emulate Android systems.

### Record and Replay

PANDA’s record and replay feature is conceptually simple. At the beginning of recording, we take a snapshot of the machine state, which includes the contents of registers

and memory. Then, we record to a log all sources of non-deterministic data entering the system, which primarily includes the sources of input and output, such as network traffic and hard-drive data, but also includes other low-level sources that we have identified in the system. When any of these inputs comes into the system, we also record the information needed for us to determine when to replay the input.

PANDA’s replay function, which has been tested extensively on two processor architectures (32- and 64-bit x86 and ARM), is quite stable and effective. It can record boot for a variety of operating systems; this action is challenging because of the complexity of the boot operation. PANDA recordings are also fairly compact in size even though our record log must capture the contents of all inputs into the system. Table 1 gives the record log sizes for a number of workflows. The modest size of these files makes them ideal

for sharing and thus for enabling repeatable experiments. One of this article’s authors has set up a website from which any of these and a number of other replay files can be downloaded and analyzed independently.<sup>1</sup>

Because PANDA allows full repeatability of replays, it is incredibly useful for dynamic analysis. Traditionally, manual dynamic analysis involves running a program inside a debugger and using the debugger to periodically inspect the program state. However, debuggers largely cannot execute backwards, so in order to inspect an earlier program state, an analyst must restart the program from scratch. This restart not only adds time to the analysis process but also changes many dynamic aspects of the program. With PANDA replays, dynamic information is the same each time, so information about the state of memory can be built up piece by piece, greatly accelerating RE.

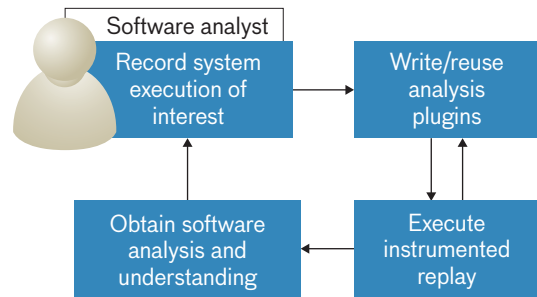
**Plugin Architecture**

PANDA plugins take the form of shared libraries that can be loaded at any time during an analysis. The plugins are event-driven; that is, they perform tasks in response to events in the system that are specified by analysts’ instructions. The analysts perform system instrumentation by using interfaces that have been made available in PANDA.

Many plugins depend on some common functionality. To avoid duplicating functionality throughout plugins while keeping the core of PANDA simple, we have implemented a mechanism for plugin-plugin interaction to allow individual plugins to expose a public interface that other plugins can utilize. The plugin-plugin interaction allows code reuse and reduces the duplication of specialized code that is used for complex analyses.

**Architecture-Neutral Analysis**

A number of dynamic analyses that happen at the system instruction level are invaluable for RE. For instance, in taint analysis, data in the system are labeled (tainted) and then tracked to enable a detailed understanding of the true information-flow patterns around, in, and out of a system. This analysis can be thought of as a PET scan for a computer [3]. In order to properly track labels, one must perform an additional complex analysis alongside every system instruction. Some of the complexity



**FIGURE 2.** In the replay-based reverse engineering (RE) workflow, PANDA can record and replay whole-system executions. This capability is the foundation of PANDA’s use in RE. To use PANDA, the analyst captures a recording and then iteratively uses or builds data analyses to incrementally build RE knowledge.

**Table 1. Record Log for Various Replays**

REPLAY	INSTRUCTIONS (BILLIONS)	LOG SIZE (MB)
Operating system boot	9.3	533.0
Spotify playing a song snippet	12.0	229.0
Malware recording	9.1	43.0
User browsing to a website	8.6	9.4

of these additional analyses is due to the differences in processor architectures of systems. For example, desktop architectures (such as x86) are more complex than power-constrained architectures (such as ARM).

PANDA avoids the difficulties associated with supporting multiple processor architectures by performing analyses in a generic intermediate representation that is not specific to a particular processor architecture. We perform dynamic binary translation, which is the process of translating the code under analysis to the intermediate representation, to enable the generic analyses. Dynamic binary translation is the underlying technology that makes some of our novel analyses possible.

<sup>1</sup><http://www.rrshare.org>

### Android Support

An emulator similar to PANDA is included in Google's Android software development kit and contains the necessary emulated hardware to produce a realistic Android environment. In order to provide Android support to PANDA, we ported the features necessary to emulate devices that are unique to Android phones. Significant additional work was required to fully support modern Android emulation: integrating telephony, camera, and Android debug bridge support; integrating secure digital (SD) memory card support; translating inputs for graphical interfaces; supporting common formats for the storage devices; arranging to support PANDA's record and replay mechanism; and employing various other bug fixes, including one for a graphics bug.

### Plugin Details

To date, more than 40 robust analysis plugins that can be applied to RE have been developed for PANDA by Lincoln Laboratory researchers, collaborators at a number of universities, and the open-source community at large. These plugins are available in our github repository at <https://github.com/moyix/panda>. The following novel plugins have proven particularly useful in RE.

### Tappan Zee (North) Bridge

Reverse engineering tasks often hinge on finding out what piece of code either implements some high-level functionality or handles some particular data. In large programs, these discoveries can be quite difficult. When the data are a fixed string embedded in the program, analysts are usually able to easily determine the function of a piece of data, but when the data are dynamic, analysts must laboriously trace the flow of data from some known input source through a chain of intermediate functions to the location where it is finally used. Moreover, when the data sought are some intermediate values not directly derived from the input, even this approach may fail.

In previous work [4], we developed a system, Tappan Zee (North) Bridge (TZB), for locating points at which we can interpose on memory accesses in a system to monitor events during system execution. We have since discovered that TZB is also immensely useful for RE. The central concept behind TZB is that memory accesses can illuminate the internal details of a system. As a program runs, functions called from different

contexts read and write input, output, and intermediate results to memory. By appropriately separating out these memory accesses according to program and calling context (Figure 3), we obtain coherent streams of data that can then be searched and analyzed for information of interest. These streams of data accessed at a particular point in a program are called tap points because they are places in the code where one might “tap” to get useful information from a system.

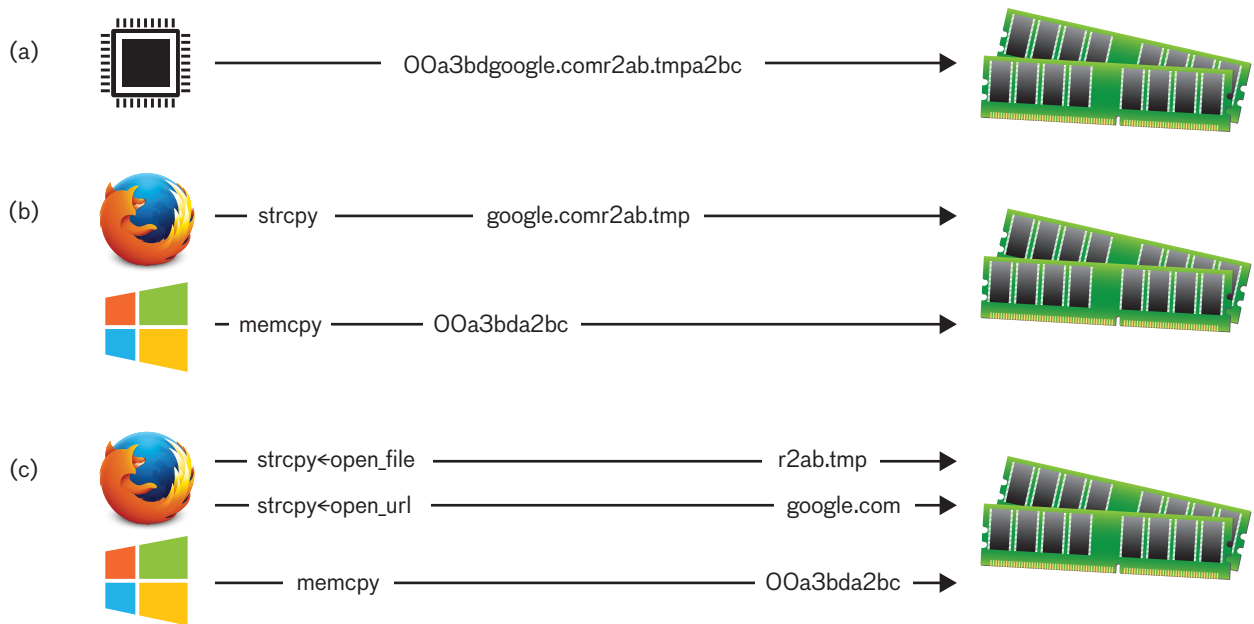
In the simplest case, we may wish to find out what part of a program handles a certain bit of data, such as a string we type into a program, or what function causes a particular string to be printed to the console or shown in the user interface. For such tasks, searching all tap points for some fixed strings will give us a set of functions that were seen to read or write data matching our search string. To accomplish this type of search, we created the `stringsearch` plugin, which tracks all memory accesses made in the system; splits them up according to the calling context, program counter, and address space; and then searches the resulting streams for a list of keywords.

Sometimes, we may not know the exact format of the data, but we may know some statistical features of it. For example, if we are searching for a digital rights management (DRM) decryption function, we know from previous work by Wang et al. [5] that the inputs to such functions have high byte entropy and are statistically random (according to a test such as Pearson's chi-squared test), but their outputs are not random. We recently used Pearson's test to locate the DRM decryption function within Spotify and showed that this function could be used to extract unencrypted audio files [6].

To support this latter type of search, PANDA can use appropriately named `unigrams` and `bigrams` plugins to collect unigram and bigram (one- or two-unit sequences in a string) statistics about each tap point. These functions collect unigram and bigram histograms for data read or written at each tap point during an execution. Once these histograms are gathered, an analyst can write scripts to compute statistical features, such as byte entropy, chi-squared values, or a distance measure (such as Kullback–Leibler divergence) to a previously observed distribution.

The ability to search through tap points for data of interest can allow a reverse engineer to quickly zero in on the parts of a program of greatest interest or to extract normally unobservable data from a program as it runs.





**FIGURE 3.** Memory accesses are made by a program with varying amounts of context: (a) presented as a single stream of information from the CPU to memory, (b) split up according to program, and (c) split up according to program, location within the program, and calling context.

The TZB system depends crucially on PANDA’s record and replay functionality: on a live execution, the amount of compute resources needed to identify and halt on every memory access and inspect its contents would cause the operating system to become prohibitively slow and impossible to analyze. In a replayed execution under PANDA, plugins can have arbitrary performance overhead without perturbing the events in the replay.

### Scissors

One of PANDA’s biggest contributions to RE is its ability to do offline analysis, that is, to collect a recording of a system’s execution at normal speed and then replay that execution with heavyweight analyses running, potentially over a long period of time. Still, many analyses are too computationally intensive to be tractable over replays that have potentially billions of instructions. To address this issue, we created the `scissors` plugin, which enables the user to excise smaller portions of replays and then analyze just the shortened portion. Combined with the ability of components such as TZB to rapidly locate sections of interest in the replay, the `scissors` plugin allows analysts to focus their attention on key events during execution and ignore everything else.

### Taint Analysis

As previously mentioned, taint analysis is the process of tagging data in the system with labels and then tracking those labels to enable a detailed understanding of the true information-flow patterns around, into, and out of a system. We have implemented dynamic taint analysis as a PANDA plugin that permits precise labeling of data in a number of ways, such as labeling file contents, network data, raw memory, and CPU registers. These labels are then tracked automatically and stored in a shadow memory that associates tainted memory, registers, and input/output buffers with label sets.

Unlike other existing taint analysis systems, our system is independent of the underlying processor architecture and has been used to analyze replays based on the x86, x86-64, and ARM processor architectures. We can also easily extend our taint analysis system to all of the architectures that our system emulator supports.

Our system includes query mechanisms that allow an analyst to ask if data are tainted at some replay point and to examine the set of associated taint labels. These mechanisms permit the analyst to ask very detailed questions about the software they are analyzing: If I mark incoming network traffic as tainted, where does it flow through the system? Is

any of this traffic interacting with vulnerable code that could potentially be exploited by an adversary? If I mark sensitive files as tainted, are they ever unknowingly exfiltrated? How much control does a potential adversary have over untrusted data at various points in the execution of a program?

Many of the specifics of how this subsystem was designed and implemented have been described in our previous work [7]. Here we describe some salient features of PANDA that enable analysts to observe software in detail:

- Whole-system support—PANDA’s taint analysis tracks labels even if they flow between different processes and privilege levels.
- Input/output support—Because our shadow memory includes the hard drive, network card, and associated input/output buffers, an analyst can precisely introspect into how data propagate through the system at a low level and can properly track how data moves through these devices.
- Replay-based taint analysis—Our taint analysis plugin uses a record and replay system to turn an intractable online analysis into a tractable offline analysis. For many platforms, such as Android, even pure emulator-based execution is barely fast enough to prevent operating system and networking timeouts during taint analyses, which are typically computationally intensive.
- Detail and fidelity—Taint analysis in PANDA focuses primarily on the detail that can be obtained from the analysis. For instance, a file can be labeled such that every byte in the file gets a different label. Further, computation is modeled with high fidelity by tracking detailed metadata with each byte of memory, allowing an analyst to measure how much the tainted data have changed since they were originally labeled.
- Interface—Taint labeling and querying can be either driven by events (through callbacks registered with the taint plugin) or invoked by a call to the interface exposed by the plugin. This choice of approaches provides flexibility to the analyst.

### Case Studies

The following three RE use cases for PANDA illustrate the system’s capabilities. In the first example, we revived an old version of the game StarCraft for which the CD key had been lost; with PANDA’s plugins, we were able to rapidly locate the key verification code and harness it to

produce keys on demand. In the second, PANDA’s whole-system replay function enabled us to perform an in-depth diagnosis of a Windows Internet Explorer vulnerability to characterize this vulnerability as a use-after-free bug (i.e., an attempt to access previously deallocated memory). In the third, an Android chat client suspected of censoring messages was quickly determined to be doing so via a censorship blacklist that was readily extracted. Note that, while we used some of the plugins that were mentioned earlier in this article, we did not apply all of them to our use cases; rather, we allowed the task at hand to drive the choice of plugins to employ.

### Reviving Legacy Code

StarCraft is a science fiction video game released in 1998 by Blizzard Entertainment. Each of the game’s discs comes with a unique CD key that identifies the copy and permits both installation and online play. Originally, CD keys were 13 numbers, but Blizzard revised later copies of the game to use keys consisting of 26 alphanumeric characters. The original 13-number format was very simple to reverse engineer; however, if you legally purchased a newer copy of the game and lost your 26-character CD key, you would be unable to install and play the game.

We used PANDA to find and rapidly reverse engineer the 26-character CD key validation algorithm for StarCraft. First, we collected a recording of the StarCraft installer rejecting a random sequence of letters and numbers. We then provided both this incorrect key sequence and the text of the rejection dialog as searches to PANDA’s TZB, which promptly found both in the replay. This discovery focused our attention on about 200,000 instructions out of the 60 million in the complete replay (a 300-fold reduction). We then used the `scissors` plugin to extract just this operative segment containing the validation algorithm.

Through manual static analysis of the code in the remaining replay segment, we ascertained that the installer decrypts the CD key and checks the high-order bits of the resulting 120-bit integer against a fixed value. This “magic number” is not immediately apparent in the code’s disassembly, but a simple PANDA plugin was rapidly written that printed the magic number out when it was read from memory through the use of a concept similar to the `stringsearch` plugin. Our analysis

showed that the fixed value was 23. Manual RE from there easily revealed the complete key-computation algorithm. Some additional mathematical analysis indicated a very low key density: only 1 in 27,000 of the possible CD keys are actually valid.

We then used the Hex-Rays decompiler to recreate source code of interest identified by the `stringsearch` plugin. The extracted code was harnessed as a decoder in a small program that was fed random keys to determine which ones were valid. Overall, this RE effort was very successful. PANDA allowed us to reduce the replay to a size at which a complicated analysis was immediately tractable with the `scissors` plugin, rapidly locate the code of interest for key validation with the `stringsearch` plugin, and ultimately to play our StarCraft game again by using a validated CD key generated by our extracted test harness.

### Deep Vulnerability Diagnosis

Software vulnerabilities often have deep causes, with the underlying bug occurring well before a potential crash or exploit. One classic example is the use-after-free bug that exploits a program's retention of information referencing invalid, deallocated memory. When a program accesses this invalid memory, the program may crash because its data structures have been corrupted; however, the crash itself will give no hint about its underlying cause—e.g., where the bug was created, when the memory was freed, or even if the bug involved a use-after-free exploit.

As an experiment to test the effectiveness of PANDA in finding deep vulnerabilities, we had a team member prepare a replay containing a known triggered vulnerability. This replay was then given out with no information other than the fact of an application's crash and the standard Windows error message, "Application has stopped working." First, we used the `replaymovie` plugin to make a series of captures from the screen and to then stitch them together into a video of replay execution. This video indicated that the failing process was Internet Explorer and that the vulnerability was triggered by loading a malicious website. We then used TZB to search for "<HTML" and "has stopped working"; this search gave us temporal bounds in the replay for the bug's location. The `scissors` plugin enabled us to reduce the size of the replay and conduct more heavyweight analyses. Using TZB again, we extracted all further output at the

<HTML tap point, which was exactly the full webpage that triggered the bug. The webpage indicated that the vulnerability was probably a use-after-free bug.

We then wrote a custom PANDA plugin called `useafterfree` to detect use-after-free memory corruption situations. This plugin was written for a Windows operating system, but it could easily be adapted for other systems. It tracks calls to Windows' low-level memory allocation functions, and it maintains shadow lists of valid and invalid memory. When a pointer to invalid memory is used, a use-after-free has occurred and the plugin detects it.

In this case study that highlights the iterative approach analysts often take while performing RE tasks with PANDA, the repeatability of PANDA replays was a key advantage. Writing custom plugins to target a specific replay is easier than writing plugins that generalize over a broad set of situations.

### Uncovering Censorship Blacklists

We cannot always trust that the software we use is acting in our interests. For example, it is not uncommon for instant messaging clients to actively censor the conversations of their users [8, 9]. Such censorship can either be performed on server-side operations or be accomplished by a client-side blacklist that is periodically updated. In the former case, PANDA can be of no help because there is no code available to run and examine in vivo; however, in the latter, PANDA can extract a list of censored words from the client. To test PANDA's ability to uncover such a list, we examined the free LINE messenger client for Android. Analysts from the Citizen Lab at the University of Toronto's Munk School of Global Affairs had previously investigated LINE to determine that it censors certain users [10].

For our analysis, we created a recording in which we launched the LINE messenger and sent an instant message to another user. The sent message did not include any content we thought might be censored. Simply by sending the instant message, we supposed that LINE would still have to load its list of censored words and check our message against it, thus leaving the list open to extraction by PANDA.

To find the encrypted wordlist, we employed the TZB plugin, supplying some guesses as to words that might be subject to censorship and then searching all memory reads and writes made by LINE for these words. This process gave us a set of tap points that contained the

sensitive words. As we suspected, the words we sought were indeed included in LINE's list of censored words. By the end of our analysis, we discovered 536 specific words that LINE was censoring in a completely automated fashion. PANDA expedited our LINE analysis far beyond the level of time and effort we would have expended had we used a more manual approach.

### Future Directions

We have been actively using PANDA for the past three years to quickly reverse engineer large, real-world software systems without the availability of source code. In most case studies during this time, we have found PANDA to be invaluable for speeding up RE, either by entirely obviating the need for manual analysis or by precisely directing human attention to the critical portions of a large code base.

In the near term, we are planning to enhance several key aspects of PANDA: performance, architecture support, and analysis capabilities. Performance can potentially be improved in our recording infrastructure and in many of our plugin implementations. PANDA is processor architecture-neutral in principle, but a number of features have not yet been ported to all supported processor architectures. For deep operating system analysis capabilities, we plan to create new plugins that encapsulate the domain-specific knowledge necessary to retrieve useful information about various operating systems.

In the long term, we wish to make this a tool that can be used by anyone to reverse engineer complex systems. We have released PANDA as an open-source tool to the cyber security community, and we have transitioned PANDA technology to several other programs within Lincoln Laboratory's Cyber System Assessments Group and to their respective sponsors within the Department of Defense. We hope that continued development by the open-source community and technical staff at Lincoln Laboratory will make this a common tool for dynamic software analysis and RE. ■

### References

1. M. Russinovich, "Sony, Rootkits and Digital Rights Management Gone Too Far," Microsoft Server & Tools Blogs, 31 Oct. 2005, available at <http://blogs.technet.com/b/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx>.
2. J.A. Halderman and E. Felten, "Sony's Web-Based Uninstaller Opens a Big Security Hole; Sony to Recall Discs," Freedom to Tinker blog, 15 Nov. 2005, available at <https://freedom-to-tinker.com/blog/felten/sonys-web-based-uninstaller-opens-big-security-hole-sony-recall-discs/>.
3. S. Mysore, B. Mazloom, B. Agrawal, and T. Sherwood, "Understanding and Visualizing Full Systems with Data Flow Tomography," *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2008, pp. 211–221.
4. B. Dolan-Gavitt, T. Leek, J. Hodosh, and W. Lee, "Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection," *Proceedings of the 2013 Association for Computing Machinery Special Interest Group on Security, Audit and Control, Conference on Computer and Communications Security*, 2013, pp. 839–850.
5. R. Wang, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Steal This Movie: Automatically Bypassing DRM Protection in Streaming Media Services," *Proceedings of the 22nd USENIX Conference on Security*, 2013, pp. 687–702.
6. B. Dolan-Gavitt, "Breaking Spotify DRM with PANDA," Push the Red Button blog, 3 July 2014, available at <http://moiyix.blogspot.com/2014/07/breaking-spotify-drm-with-panda.html>.
7. R. Whelan, T. Leek, and D. Kaeli, "Architecture-Independent Dynamic Information Flow Tracking," *Proceedings of the 22nd International Conference on Compiler Construction*, 2013, pp. 144–163.
8. J. Knockel, J.R. Crandall, and J. Saia, "Three Researchers, Five Conjectures: An Empirical Analysis of TOM-Skype Censorship and Surveillance," *Proceedings of IEEE Symposium on Foundations of Computational Intelligence*, 2011.
9. A. Senft, A. Sinpeng, A. Hiltz, et al., "Asia Chats: Analyzing Information Controls and Privacy in Asian Messaging Applications," The Citizen Lab's Reports and Briefings, 14 Nov. 2013, available at <https://citizenlab.org/2013/11/asia-chats-analyzing-information-controls-privacy-asian-messaging-applications/>.
10. S. Hardy, "Asia Chats: Investigating Regionally-Based Keyword Censorship in LINE," The Citizen Lab's Reports and Briefings, 19 Nov. 2013, available at <https://citizenlab.org/2013/11/asia-chats-investigating-regionally-based-keyword-censorship-line/>.

**About the Authors**



**Ryan J. Whelan** is a member of the technical staff in the Cyber System Assessments Group at Lincoln Laboratory. He currently conducts research in the areas of static and dynamic program analysis and transformation, compiler design and implementation, malware analysis, software reverse engineering, vulnerability discovery, and instrumentation techniques for cyber security. Prior to joining the Laboratory, he earned bachelor's and master's degrees in computer engineering from Northeastern University in 2011 and 2012, respectively. During that time, he completed several cyber security internships at Lincoln Laboratory.



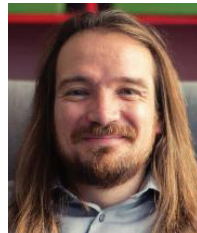
**Timothy R. Leek** is a member of the technical staff in the Cyber System Assessments Group. He is currently researching automated techniques for finding errors in programs by using a combination of static and dynamic analysis and some machine learning. He joined Lincoln Laboratory in 2001 as a member of the Ballistic Missile Defense System Integration Group, where he investigated machine learning techniques for building decision architectures. Prior to joining the Laboratory, he worked as a technical writer for U.S. Surgical Corporation, as a scientific programmer at Yale University, and as a systems administrator at the Salk Institute for Biological Studies in La Jolla, California. After earning a master's degree in computer science at the University of California, San Diego, in 1997, he joined Bolt Beranek and Newman, where he applied statistical methods to information retrieval, topic detection and tracking, machine translation, and summarization. He also holds bachelor's degrees in physics and English from the University of Connecticut.



**Joshua E. Hodosh** is a member of the technical staff in the Cyber System Assessments Group and has been at Lincoln Laboratory since 2010. He holds a bachelor's degree in computer science from Rensselaer Polytechnic Institute and a master's degree in computer science from Northeastern University.



**Patrick A. Hulin** joined the Cyber System Assessments Group in 2014. His current research lies primarily in the areas of virtual machine introspection, automated software reverse engineering, dynamic program analysis, and vulnerability understanding. He earned a bachelor's degree in mathematics from MIT in 2014.



**Brendan Dolan-Gavitt** is an assistant professor of computer science at the New York University Polytechnic School of Engineering. Previously, he worked as a postdoctoral researcher in the Intrusion Detection Systems Lab at Columbia University. His research interests lie in the area of systems security, particularly in the development of automated techniques for understanding computing systems and the application of that understanding to the creation of novel defenses. He is also active in the rapidly growing field of memory forensics and has published papers on extracting forensically relevant information from images of RAM. He received his doctoral degree from the Georgia Institute of Technology and his bachelor's degree in mathematics and computer science from Wesleyan University. He also spent two years working as an information security analyst and researcher for the MITRE Corporation.

# Moving Target Techniques: Leveraging Uncertainty for Cyber Defense

Hamed Okhravi, William W. Streilein, and Kevin S. Bauer

Cyber moving target techniques involve randomizing cyber system components to reduce the likelihood of successful attacks, adding dynamics to a system to shorten attack lifetime, and diversifying otherwise homogeneous collections of systems to limit attack damage. A review of five dominant categories of cyber moving target techniques assesses their benefits and weaknesses.



**Securing critical computer systems** against cyber attacks is a continual struggle for system managers. Attackers often need only find one vulnerability (a flaw or bug that

an attacker can exploit to penetrate or disrupt a system) to successfully compromise systems. Defenders, however, have the technically difficult task of discovering and fixing every vulnerability in a complex system, which usually comprises an operating system, device drivers, numerous software applications, and hardware components. Within cyberspace, this imbalance between a simple, one-vulnerability attack tactic and a complicated, multipart defense strategy favors attackers. While defensive applications have grown significantly in complexity and size over many years, malicious software, i.e., malware, has remained relatively simple, computationally small, and still effective in bypassing defensive applications [1].

A major contributing factor to the imbalanced security of cyberspace is the static nature of systems and defenses. The same copy of a popular software application with the same internals developed by a major software vendor may run on millions of machines. As a result, an attack designed to infect that software application is likely to compromise millions of machines. Similarly, many defensive applications are static; they discover suspicious inputs by applying a set of rules and checks commonly used by software built to detect attacks. Therefore, clever cyber invaders can craft attacks to bypass existing defenses by analyzing local copies of readily available defensive applications and then exploiting the weaknesses within those applications.

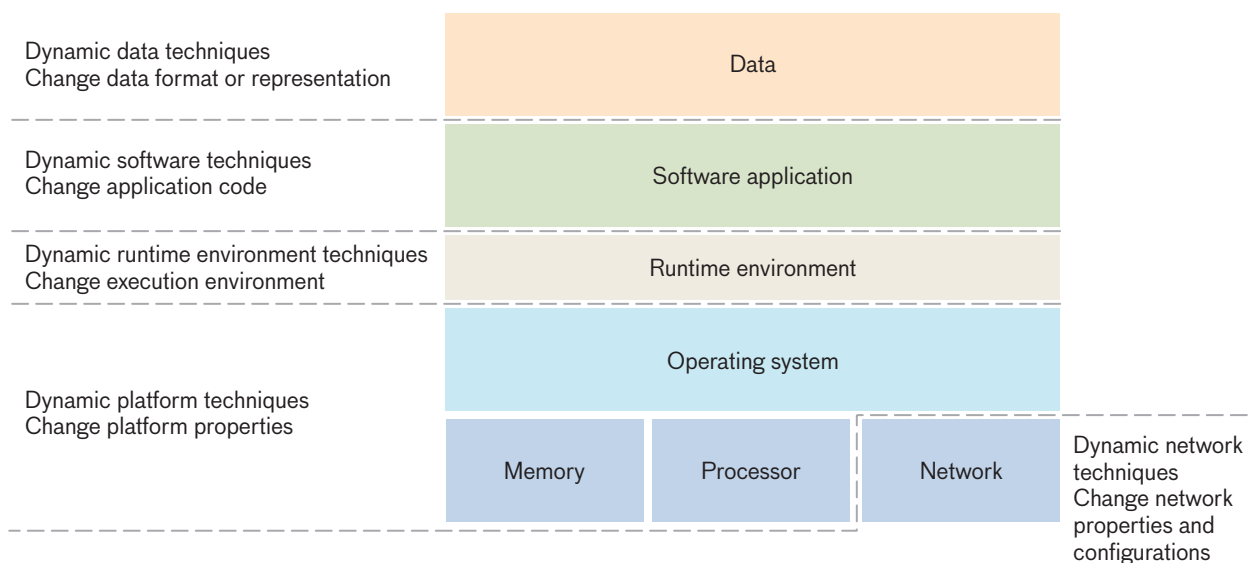
A promising approach to defense that attempts to rebalance the cyber landscape is known as cyber moving target (MT) defense (or just moving target). Moving target techniques change the static nature of computer systems to increase both the difficulty and the cost (in effort, time, and resources) of mounting attacks. Simply put, these techniques turn systems into moving targets that will be hard for cyber villains to hit. Defenders using MT techniques pursue any or all of the following goals: make computer systems more dynamic by changing their properties over time, make internals of computer systems more random and nondeterministic, and make computer systems more diverse.

Although numerous techniques categorized as MT have been offered in the academic literature, we are limiting our overview of dynamic MT techniques to those in five computer domains—platforms, runtime environment, software, data, and network. Readers can find a more detailed discussion of these five categories of MT techniques in Okhravi et al. [2].

### Moving Target Overview

An overview of different components of a computer system is a good place to start to understand the domains of MT techniques. For ease of design and implementation, a computer system (e.g., a desktop or laptop machine, a mobile device, or a process control

machine in an industrial control system) often consists of multiple layers of software and hardware. These layers are commonly referred to as the software stack although the stack includes the hardware elements as well. Each layer relies on other layers for its proper operation and function. Figure 1 presents one representation of such a layered design. At the very bottom of the software stack are the hardware components of the machine: the processor, the motherboard, the memory cards, and other peripheral devices and cards, such as the sound card and video card. Above this layer resides the operating system, which is responsible for controlling and managing the hardware components and providing an abstraction of them to the application. This abstraction is key to the interoperability and compatibility of the applications because the vast majority of the applications do not interact directly with the hardware components; rather, they use the operating system’s abstraction. The abstraction layer, which is the interface that the operating system provides to the application, is sometimes referred to as the runtime environment. The hardware and operating system of a machine are collectively called the platform. Above the operating system reside the applications that are used to process and present data. The data themselves and their representation can be considered a layer atop the application. Finally, many systems do not operate as isolated devices but, in fact, are connected to other



**FIGURE 1.** On the right side of the figure is a depiction of the software stack. The layers of the stack address the five different domains of cyber moving target techniques (explained in text at the figure edges) that are assessed in this article.

machines through a network. In general, five domains of MT techniques address dynamically changing the above-mentioned software stack layers.

### Dynamic Platform

The dynamic platform domain consists of cyber defensive techniques that dynamically change the properties of the computing platform. Consider a system that runs a given application on top of multiple operating systems and hardware architectures. The application can run on top of a platform consisting of the Fedora operating system and x86 processor architecture or a platform consisting of the FreeBSD operating system and ARM processor architecture. Such a system can be implemented by compiling the application for different processor architectures and employing a platform-independent checkpointing mechanism to preserve the current state of the application during platform changes [3]. This type of system illustrates a dynamic platform MT technique. Other examples of dynamic platform techniques include a voting system that runs an application on top of different platforms, each platform voting on the output of the system [4], or a system that randomizes the operating system's internals that are unimportant for the correct functionality of the application.

The major benefit of a dynamic platform technique is that it can prevent platform-dependent attacks. Crafting a successful exploit against a system usually requires that an attacker consider the exact platform of that system. By varying the computing platform, an MT technique can mitigate attacks that are platform-dependent. An attacker can develop a strong attack by incorporating different exploits against different platforms, but this approach increases the cost (in time and/or computation complexity) of developing the attack. Note that dynamic platform techniques cannot mitigate attacks that target a higher-level application logic flaw and that do not depend on the platform. For example, SQL<sup>1</sup> injection attacks, which inject malicious commands into a database application by leveraging a flaw in the application's high-level logic, are typically not mitigated by dynamic platform techniques.

While dynamic platform MT techniques offer the potential to defeat platform-dependent attacks, these techniques can increase the complexity of the overall system,

are generally difficult to effectively manage, and can actually be detrimental to security if used inappropriately [5]. Perhaps the greatest challenge from a system complexity and management perspective is the synchronization of application state across the set of diverse platforms. Examples of application states could include information about open data files, user input from a keyboard or mouse, or network traffic that needs to be correctly delivered to a specific running process (while correctly maintaining connection-specific state in the kernel). Synchronizing these resources among the dynamic platforms in real time requires a complex management infrastructure that can migrate state with speed and agility. Such a management infrastructure increases system complexity considerably.

Another potential limitation of dynamic platform techniques is that the use of multiple distinct platforms can actually increase the system's attack surface, that is, the components of the system that are exposed to and could be targeted by a potential attacker. Suppose that a dynamic platform MT technique migrates an application between three platforms: Linux, Windows, and Mac. If the attacker has an exploit that works on the Windows host, the attacker simply needs to wait until the application migrates to the Windows host to launch the exploit and compromise the application. Making the program migration less predictable can help, provided that the attacker cannot reliably guess which platform is running the application.

Dynamic platform techniques are only effective defenses when the attacker must compromise all platforms (i.e., an in-series configuration) not just one platform (i.e., an in-parallel configuration). If the attack requires a long time to succeed (a long-duration disruption of service), a dynamic platform approach can be helpful in thwarting the attack; for short-duration attacks, that approach can be detrimental to security because the attacker's goal may be accomplished on one platform.

### Dynamic Runtime Environment

Techniques in the dynamic runtime environment domain dynamically change or randomize the abstraction provided by the operating system to the applications, without hindering any important functions of the system. One of the most important abstractions in a computer system is how memory is presented to the applications. For various reasons, including isolation of different applications, compatibility, and interoperability, a memory location that is

<sup>1</sup>SQL stands for Structured Query Language, a standardized programming language for requesting information from a database.



presented to an application in most modern computer systems is not a direct representation of the actual physical memory. Rather, a redirection is applied by the operating system, i.e., an abstraction known as the virtual memory. A well-known dynamic runtime environment MT technique randomizes what addresses in the virtual memory are used by the application. The technique is typically referred to as address space layout randomization (ASLR) [6] and is implemented in most modern operating systems, including Linux, Windows, Mac OS X, Android, and iOS. By randomizing the addresses, ASLR makes exploit development significantly more difficult for attackers because they do not know where to place their malicious code on the system. Other dynamic runtime environment techniques include those that change the processor instruction encoding (also called instruction set randomization) or finer-grained variants of ASLR in which smaller regions of memory are randomized.

Dynamic runtime environments are among the most practical and widely deployed MT techniques. Despite the success of this MT domain, two important weaknesses can allow an attacker to circumvent the defense. First, ASLR requires memory secrecy. If the contents of memory are disclosed or leaked to an attacker, the attacker may be able to use this information to defeat ASLR. Such memory disclosures are possible via separate vulnerabilities, known as buffer over-read vulnerabilities, in which the contents of memory are read beyond the allowed boundary, disclosing how memory has been randomized. Without strict memory secrecy, an attacker can circumvent the ASLR protections to launch code injection or code reuse attacks. Second, the low granularity of randomization in many ASLR implementations reduces the overall protection provided by the technique. For example, in Linux, only the start location of certain memory regions (e.g., dynamically linked libraries) is randomized by default, and the executable program code itself is often not compiled with ASLR support. As such, this section of the program's memory is not protected and can be a vector for exploitation.

### Dynamic Software

In the dynamic software domain, MT techniques randomize or diversify the internals of the software application. One technique, the multcompiler [7], creates different versions of software executables (binaries) from the same source code (e.g., written in C) that perform the

same function. Variations in the versions can arise from the use of different but equivalent processor instructions utilized during the compilation process or from the use of the same instructions utilized in different locations inside the executable. Note that a given copy of the executable with a given set of internals may never change, but various machines in an enterprise may run different executables. In other words, this technique can create spatial diversity (i.e., diversity among many machines) as opposed to temporal diversity (i.e., diversity in one machine over time). The major benefit of dynamic software techniques is that they mitigate the impact of large-scale attacks. If an exploit is designed against a given variant of the executable, that exploit will have a small chance of working against other variants of the executable. Hence, an attacker cannot compromise many machines at once. This situation is contrary to the current one in which an attacker develops malware that can successfully compromise many machines running the same target application. In recent sophisticated breaches, attackers reuse parts of the benign code of the target application itself to achieve malicious behavior. Known as code reuse attacks, or return-oriented programming attacks [8], these attacks can successfully circumvent existing defenses that detect and stop foreign pieces of code. By varying the benign application code, dynamic software techniques can effectively stop code reuse attacks.

Dynamic software techniques often employ specialized compiler techniques to produce executable software variants with different and unpredictable memory layouts. These variants may use padding (adding meaningless bytes of data) to make the size of memory regions unpredictable. They also may contain within the executable code a no-operation (NOP) instruction that does not perform any operation but can make code reuse attacks hard to launch because the instruction changes the location of other instructions.

Dynamic software techniques suffer from a variety of weaknesses. Recompile to produce a software variant requires access to a program's source code and is not possible with proprietary, third-party software for which source code is not made available. Furthermore, ensuring correct operation of the compiled variant can be challenging because one cannot simply verify a known integrity measurement of the executable file to guarantee that the code has not been (maliciously) modified.

Another drawback of dynamic software methods is that software is often compiled with special optimization flags that reduce the space and/or computational complexity of the compiled binary code. An MT technique that explicitly compiles the software to introduce randomness in the memory layout (by randomizing the size and/or location of objects) may not be compatible with the space saving or compute-time saving optimization passes performed by the compiler. Consequently, the dynamic software is unlikely to maintain the same performance properties as the ideally optimized compiled code.

In addition, dynamic software techniques that use execution monitors to instrument and compare multiple versions of an executable introduce significant performance costs. For example, if an MT technique compares execution for an application that has two variants, there is at least a twofold performance cost relative to native execution of the application (in terms of processor, memory, and input/output utilization). This cost may be reasonable for protecting one or two applications for which the highest degree of security is required, but the cost is likely to be unacceptable for the protection of all applications running on a host. Techniques in the dynamic software domain may also be subverted by information leakage attacks. If attackers can expose how an executable has been diversified, they can attack it as if it were not diversified at all.

### Dynamic Data

Moving target techniques under the dynamic data domain change the format, syntax, representation, or encoding of the application data to make attacks more difficult. In this domain, the diversity can be temporal or spatial. For example, to protect a Linux operating system, a defender could dynamically change the representation of the user identifier (UID) that determines what access rights a user has. This defense is effective against attempts that seek to increase a user's access rights; for example, an attacker may attempt to change the UID to that of a privileged administrator so that the attacker can exploit the expanded rights to access sensitive resources. This type of attack is one example of a larger class known as privilege escalation actions that can be mitigated by UID randomization.

Dynamic data techniques offer the promise of protecting data from theft or unauthorized modification, but

these techniques suffer from two important weaknesses. First, the number of acceptable data encodings is limited. For example, for encoding binary data either the base64 or the hexadecimal encoding scheme would most likely be used because there are few other accepted standards for data encoding. Nonstandardized schemes are certainly possible, but these may increase the complexity of the interoperation among system components. Second, the use of additional data encodings may also increase the attack surface of the software. For each encoding type, the software must have the proper parsing code to encode and decode the data. This additional parsing code itself could have security-relevant software bugs.

### Dynamic Network

Techniques in the dynamic network domain change the properties of the network to complicate network-based attacks. One such technique frequently changes the Internet protocol (IP) addresses of the machines in an enterprise network [9]. This IP rotation technique can thwart rapidly propagating worms that use a fixed hit list of IP addresses to infect a network. Another technique, known as an overlay network, creates dynamically changing encrypted tunnels (i.e., encrypted communication connections over public networks).

Dynamic networks is an appealing class of techniques to reduce an adversary's ability to conduct reconnaissance on a network, map a defended network, or select specific hosts for a targeted attack. However, these techniques face two important obstacles to deployment. First, because many dynamic network techniques lack a well-articulated threat model, it may be unclear to network defenders what threat needs to be mitigated and thus how best to deploy the defensive technique. Consider a technique that isolates a small group of machines from the larger network (or Internet). If hosts within the isolated network can still communicate to hosts beyond the isolated network, protected hosts may be vulnerable to any number of client-side attacks that exploit vulnerabilities within the unprotected hosts' web browsers or document viewers. For example, targeted spear phishing (fraudulent email messages that try to elicit information such as passwords to Internet accounts) could penetrate a protected network through the network's connections to unprotected hosts. Dynamic network-based MT techniques do not address these types of attacks.

Second, many dynamic network techniques introduce randomization into the fundamental protocols that are used on the Internet. However, the effectiveness of this randomization at stopping attacks is unclear. Suppose an MT technique randomizes network identifiers (such as an IP address). If service discovery protocols such as the domain name service (DNS) are used to convert human-readable domain names to machine-readable IP addresses, these services may undo any potential security benefit obtained through the MT technique itself, provided that the attacker can issue DNS queries.

**Summary**

One way to understand the benefits of MT techniques is to look at the steps of a cyber attack that these techniques are trying to mitigate. To successfully compromise a system, an attacker must progress through the several phases depicted in Table 1. The first phase is conducting reconnaissance; an attacker collects information about the target. The second phase is accessing the victim; the attacker collects enough information about the configurations, applications, and software versions that are running on the target machine to develop an attack against it. During the third phase, the attacker develops an exploit against a vulnerability in the target machine. Next is the launch of the attack, which may include, for example, sending a malicious network packet to the target machine, luring the user to click on a maliciously crafted link, or using a malicious thumb drive. After the attack is launched and verified, the attacker may take additional steps to maintain a foothold on the target

machine (i.e., persistence). These phases, together referred to as the cyber kill chain, are correlated in Table 1 with the MT technique domain that is aimed at mitigating the effectiveness of each step.

**Evaluating MT Techniques**

Because attackers need only exploit the weakest link in any MT technique to bypass it or render it ineffective, it is difficult for researchers to evaluate the fundamental effectiveness of the technique. Lincoln Laboratory researchers have been developing MT evaluation and assessment capabilities to further their understanding of the techniques’ efficacy.

**Quantifying Information Leakage Attacks**

Although a variety of classes of attacks have been used against MT techniques, for the sake of brevity in our discussion, we describe a class of cyber attacks known as information leakage to illustrate our evaluation capabilities. Information leakage attacks are a crucial class to consider when one is measuring the effectiveness of MT techniques because these attacks are widely employed.

Information leakage attacks, through which an attacker can discover how a system has been randomized or diversified, can be achieved in two ways: (1) by exploiting a vulnerability that forces a system to include its randomized internals directly in its output, thereby allowing the attacker to observe those internals (this type of attack is also referred to as a memory disclosure attack) or (2) by using remote side-channel attacks, i.e.,

**Table 1. Primary Attack Phases Disrupted by Techniques in the Five MT Domains**

MT DOMAINS	ATTACK PHASES				
	RECONNAISSANCE	ACCESS	DEVELOPMENT	LAUNCH	PERSISTENCE
Dynamic networks	✓			✓	
Dynamic platforms		✓	✓		✓
Dynamic runtime environments			✓	✓	
Dynamic software			✓	✓	
Dynamic data			✓	✓	

ones in which the randomized internals of a system are not leaked directly through the output but through an indirect property of the output, such as its timing.

Lincoln Laboratory researchers have discovered various new classes of side-channel attacks, including remote timing and fault-analysis attacks (which will be discussed in the following paragraphs), and have performed in-depth evaluations of their impact and how much information they can reveal to an attacker [10]. These attacks are particularly important in the context of dynamic software and runtime environment MT techniques.

Consider the code snippet below:

```

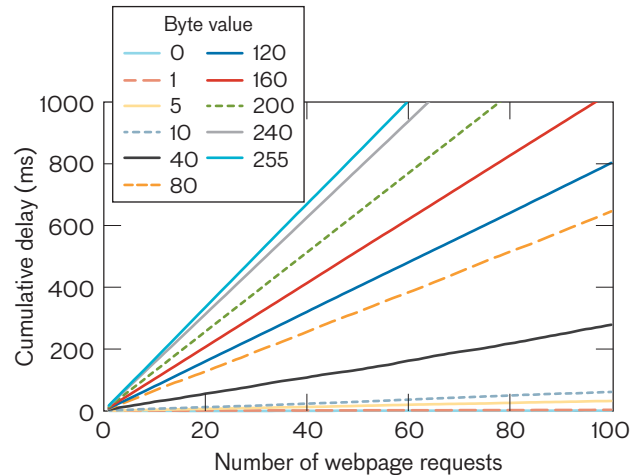
1 i = 0;
2 while (i < ptr->value)
3     i++;

```

Attackers can redirect the pointer `ptr` to a chosen location in the memory. In such cyber attacks, the timing of the loop (a sequence of instructions repeated until the desired result is achieved) will depend on the byte value at that memory location. If the byte value is high, the loop takes a long time to terminate; if it is low, the loop terminates rapidly. By remotely observing these timing differences, an attacker can infer byte values in memory, thus undoing the impact of software randomization or diversification.

We have evaluated the effectiveness of a remote timing side-channel attack against Apache, the most popular web server on the Internet. As the results of the assessment of this attack illustrate (Figure 2), the cumulative delay in a webpage request that can be observed by an attacker correlates well with the sensitive byte values from the diversified software of Apache stored in memory.

Figure 3 illustrates a more general result for the amount of information leaked to an attacker via timing for Linux’s main system library, `libc`. Here the  $x$ -axis indicates a metric called uncertainty set size (USS), which measures the uncertainty in attackers’ knowledge of the target system if they are able to observe the timing information. The  $y$ -axis denotes the fraction of the diversified software’s functions from which attackers can infer information through timing attacks. The different lines indicate the number of timing values observable by attackers.



**FIGURE 2.** The data show the correlation between cumulative delays in Apache webpage requests and the byte values stored in memory. By measuring the delay, an attacker can perform a remote timing attack and nullify the effectiveness of software diversification.

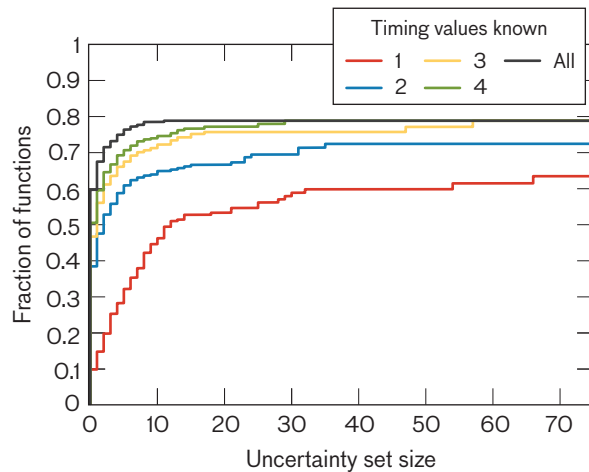
The figure shows that if attackers can observe just one timing value for the target server, they can narrow down 45% of all functions to a set of 10 or smaller (USS = 10). Note that a USS value of zero indicates the attackers made a correct identification of an exact function without any uncertainty. The figure also illustrates that by measuring a handful of timing samples, attackers can infer a lot of information from a diversified software application, thus negating the effect of randomization for the majority of functions.

Side-channel attacks can also be performed by using fault analysis, i.e., influencing a system to cause an error that the attacker can examine to gain insight into the system’s internal operation. For example, the code snippet below indicates a side-channel attack in which the attacker can infer on the basis of the output of the application whether a byte value is zero or nonzero. If the output is “SUCCESS,” `ptr` points to a byte value of nonzero; if the output is “ERROR,” the byte value is zero.

```

1 recv(socket, buf, input);
2 if (ptr->value)
3     rv = SUCCESS;
4 else
5     rv = ERROR;
6 send(socket, &rv, length);

```



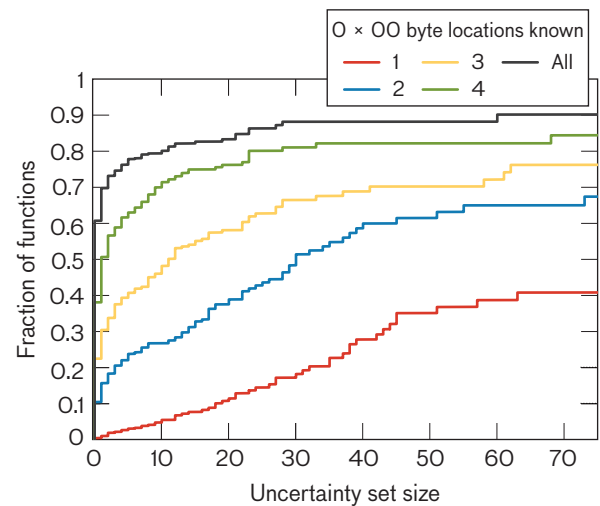
**FIGURE 3.** The plot shows the fraction of information on system functions ( $y$ -axis) leaked to an attacker from libc via timing side-channel attacks in which the indicated number of timing values is known. On the  $x$ -axis, the numbers represent measurements of the uncertainty in the attacker's knowledge of the target system.

Figure 4 illustrates the results for the fault-analysis attack for libc. Similar to timing side-channel attacks, fault side-channel attacks can leak valuable information to an attacker. For example, merely knowing the location of four zero bytes in the entire libc library allows an attacker to uniquely fingerprint 38% of functions ( $USS = 0$ ) and narrow down 70% of the functions to a set of 10 or smaller ( $USS = 10$ ), as shown in the figure by the green line on these points:  $(x = 0, y = 0.38)$  and  $(x = 10, y = 0.7)$ .

To summarize, our findings indicate that MT techniques can be maliciously bypassed using side-channel attacks. Our evaluations indicate that even a small number of timing or fault-analysis samples can leak a significant amount of information to an attacker. These results suggest that MT techniques must re-randomize system internals periodically to be resilient against information leakage attacks [11].

### Practical Considerations

When deciding to deploy an MT technique, system defenders have many practical issues to consider. They should understand the potential impact of the MT technique on the system's performance. Many MT techniques offer security against strong adversaries, but incur performance penalties that for some applications could be



**FIGURE 4.** The plots show the amount of information (about functions) that is leaked to an attacker from libc via a fault-analysis side-channel attack when various byte locations are known. Again, the  $x$ -axis indicates the uncertainty of the attacker's knowledge of the system.

prohibitively high. Recognizing the performance requirements of the system and the expected performance costs of the MT technique can help defenders make the right decision about deploying MT defenses.

Defenders should also understand the effectiveness of an MT technique against a relevant threat model before it is deployed. Techniques that have shown high effectiveness against realistic attack models should be selected before those that have uncertain benefits or those that protect against an unrealistic threat. Hence, it is important to have access to a well-defined attack model that describes the exact types of attacks that are of concern and that are relevant to the system being protected.

Finally, MT techniques do not necessarily solve all security problems; rather, they are best suited to defending against specific threats. Defenders, therefore, should understand the composability (i.e., combinatorial possibilities) of MT and non-MT techniques so that they can enhance protections against cyber security threats. For example, defenders may want to guard against code injection attacks by using ASLR. But to improve security even more, they might add signature-based network monitoring to examine network traffic in real time and drop all packets that appear to contain code injection payloads.

## Future Directions

Future research in MT techniques will take multiple directions. In designing new techniques or evaluating existing ones, researchers should analyze whether or not the additional complexity created by the randomization or diversification of the system's components is actually exposed to a potential attacker. Many MT techniques create complexity in a system component, but attackers can avoid or bypass the complexity through attacks that exploit information leakage or attacks that work regardless of the specific internals of a component (e.g., higher-level logic flaws in the application). The challenge for defenders, then, is to ensure that the complexity is not exposed to the system's operators and maintainers. Ease of deployment, operation, and maintenance is important for widespread deployment of cyber defensive techniques.

Additional work is needed in the area of evaluation and assessment of MT techniques. For cyber security to transition from a craft to a science, it is important for researchers to have concrete, meaningful, and repeatable evaluation methods. An imperative part of evaluation is the development of metrics that define measurement units of security and that can be used to evaluate the absolute security offered by an MT technique and to comparatively assess it against other techniques. Meaningful and objective evaluation of MT techniques can benefit from a variety of approaches, including abstract analysis, modeling and simulation, test bed experimentation, and real-world measurements in operational systems.

Finally, an important future direction for MT research is the examination, study, and evaluation of the composability of MT techniques with other MT and non-MT defenses. Cyber defenses in general, and MT techniques specifically, do not provide a "silver bullet," protecting against every known cyber attack. Therefore, in practice, multiple defenses should be combined to provide adequate protection of systems. Understanding the impact of these defenses on each other, as well as the composability challenges arising from these defenses, is an open research area. Other important areas for further study include determining if a defense will improve, co-exist, or conflict with another defense and investigating how a defense is influenced by second-order effects, such as an attacker's reactions to the presence of a new MT technique.

## Acknowledgments

The authors gratefully thank the following colleagues at Lincoln Laboratory for their contributions and support to the research on moving target techniques: David Bigelow, Kevin Carter, Robert Cunningham, Veer Dedhia, Thomas Hobson, Mark Rabe, James Riordan, Robert Rudd, and Richard Skowyr. ■

## References

1. D. Kaufman, *An Analytical Framework for Cyber Security*, Defense Advanced Research Projects Agency, 2011, available at [www.dtic.mil/dtic/tr/fulltext/u2/a552026.pdf](http://www.dtic.mil/dtic/tr/fulltext/u2/a552026.pdf).
2. H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein, "Finding Focus in the Blur of Moving Target Techniques," *IEEE Security & Privacy*, vol. 12, no. 2, 2014, pp.16–26.
3. H. Okhravi, A. Comella, E. Robinson, and J. Haines, "Creating a Cyber Moving Target for Critical Infrastructure Applications Using Platform Diversity," *International Journal of Critical Infrastructure Protection*, vol. 5, no. 1, 2012, pp. 30–39.
4. B. Salamat, A. Gal, T. Jackson, K. Manivannan, G. Wagner, and M. Franz, "Multi-variant Program Execution: Using Multi-core Systems to Defuse Buffer-Overflow Vulnerabilities," *Proceedings of the IEEE International Conference on Complex, Intelligent and Software Intensive Systems*, 2008, pp. 843–848.
5. H. Okhravi, J. Riordan, and K. Carter, "Quantitative Evaluation of Dynamic Platform Techniques as a Defensive Mechanism," pp. 405–425 in *Research in Attacks, Intrusions and Defenses, Lecture Notes in Computer Science*, A. Stavrou, H. Bos, and G. Portokalidis eds. Cham, Switzerland: Springer International Publishing, 2014.
6. PaX Team, "PaX Address Space Layout Randomization (ASLR)," 2003, available at <https://pax.grsecurity.net/docs/aslr.txt>.
7. M. Franz, "E Unibus Pluram: Massive-Scale Software Diversity as a Defense Mechanism," *Proceedings of the 2010 Workshop on New Security Paradigms*, 2010, pp. 7–16.
8. H. Shacham, "The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)," *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 552–561.
9. S. Antonatos, P. Akritidis, E.P. Markatos, and K.G. Anagnostakis, "Defending Against Hitlist Worms Using Network Address Space Randomization," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 51, no. 12, 2007, pp. 3471–3490.
10. J. Seibert, H. Okhravi, and E. Söderström, "Information Leaks Without Memory Disclosures: Remote Side Channel Attacks on Diversified Code," *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 54–65.

11. D. Bigelow, T. Hobson, R. Rudd, W. Streilein, and H. Okhravi, "Timely Rerandomization for Mitigating Memory Disclosures," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 268–279.

### About the Authors



**Hamed Okhravi** is a senior staff member in the Cyber Analytics and Decision Systems Group at Lincoln Laboratory. He leads programs and conducts research in the area of systems security. His research interests include cyber security, science of security, security metrics, and operating systems. Currently, he is developing cyber

attack resilient systems and networks, focusing on analyzing and creating cyber moving target techniques and resilient systems. He has served as a program committee member for a number of academic conferences and workshops, including the Association for Computing Machinery's (ACM) Conference on Computer and Communications Security, International Symposium on Research in Attacks, Intrusions, and Defenses, ACM Workshop on Moving Target Defense, and ACM SafeConfig Workshop. He is the recipient of a 2014 MIT Lincoln Laboratory Early Career Technical Achievement Award and 2015 MIT Lincoln Laboratory Team Award for his work on cyber moving target research. He holds master's and doctoral degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign.



**William W. Streilein** is the leader of the Cyber Analytics and Decision Systems Group. He initiates and manages research and development programs in cyber security. His current research interests include the application of machine learning and modeling techniques to problems in cyber decision making, investigation and

development of quantitative metrics for cyber security, exploration of moving target techniques to improve the resiliency of cyber and cyber-enabled systems, and exploration of automated techniques for discovering ways government missions map to a cyber infrastructure and for assessing risk to mission systems. He holds a bachelor's degree in mathematics from Austin College, a master's degree in electronic and computer music from the University of Miami, and a doctorate in cognitive and neural systems from Boston University. He is a senior member of the IEEE.



**Kevin S. Bauer** was a member of the technical staff in the Laboratory's Cyber Systems and Technology Group from 2012 to 2015. His research interests included privacy-enhancing technologies, low-latency anonymous communications, cyber security experimentation, and network security. He holds a bachelor's

degree in computer science from the University of Denver and a doctoral degree in computer science from the University of Colorado–Boulder.

# Secure Embedded Systems

Michael Vai, David J. Whelihan, Benjamin R. Nahill, Daniil M. Utin,

Sean R. O'Melia, and Roger I. Khazan

Developers seek to seamlessly integrate cyber security within U.S. military system software. However, added security components can impede a system's functionality. System developers need a well-defined approach for simultaneously designing functionality and cyber security. Lincoln Laboratory's secure embedded system co-design methodology uses a security coprocessor to cryptographically ensure system confidentiality and integrity while maintaining functionality.



**Department of Defense (DoD) systems**, e.g., computer networks, are increasingly the targets of deliberate, sophisticated cyber attacks. To assure successful missions, military systems must be secured to perform their intended functions, prevent attacks, and operate while under attack. The DoD has further directed that cyber security technology must be integrated into systems because it is too expensive and impractical to secure a system after it has been designed [1]. To address this directive, Lincoln Laboratory is using a co-design approach to systems that meet both security and functionality requirements. The Laboratory is at the research and development forefront of system solutions for challenging critical missions, such as those to collect, process, and exchange sensitive information. Many of Lincoln Laboratory's prototype systems must be designed with security in mind so that they can be quickly brought into compliance with the DoD's cyber security requirements and support field tests and technology transfer.

Many DoD systems require the use of embedded computing. An embedded computer system is designed for a dedicated function, in contrast to a general-purpose computer system, e.g., a desktop computer, which is designed for multiple functions [2]. An ideal design for an embedded system optimizes performance, e.g., small form factor, low power consumption, and high throughput, while providing the specific functionality demanded by the system's purpose, i.e., its mission. Developers must also determine the embedded system's security requirements according to mission objectives and a concept of operations (CONOPS). In general, security should be robust



enough to prevent attacks, ensuring that a system can successfully support a mission. Developers may need to enable a system to continue functioning, albeit with possibly degraded capabilities, when security fails. The design of security for an embedded system is challenging because security requirements are rarely accurately identified at the start of the design process. As a result, embedded systems' engineers tend to focus on well-understood functional capabilities rather than on stringent security requirements. In addition, engineers must provide security that causes minimal impacts on a system's size, weight, and power (SWaP), usability, cost, and development schedule.

To meet these challenges, we established a secure embedded system development methodology. When securing a system, we strive to achieve three goals: confidentiality, integrity, and availability, which are often referred to as the CIA triad for information security. The CIA triad is defined for embedded systems as follows:

- Confidentiality ensures that an embedded system's critical information, such as application code and surveillance data, cannot be disclosed to unauthorized entities.
- Integrity ensures that adversaries cannot alter system operation.
- Availability assures that mission objectives cannot be disrupted.

In this article, we use the example of a hypothetical secure unmanned aircraft system (UAS) to illustrate how we use cryptography to ensure confidentiality and integrity. Using this example, we demonstrate the identification of potential attack targets by considering the CONOPS, the development of countermeasures to these attacks, and the design and implementation of a cryptography-based security architecture. Because cryptography does not directly enable availability, we also provide insight into ongoing research that extends our methodology to achieve the resilience required to improve the availability of embedded systems.

### Challenges in Securing Embedded Systems

An embedded system will provide very little, if any, SWaP allowance for security; thus, security must not impose excessive overheads on the protected system. While the DoD has some of the most demanding applications in terms of throughput and SWaP, it no longer drives the development of processor technology. Therefore, security

technologies must be compatible with embedded systems that use commercial off-the-shelf (COTS) processor hardware platforms that the DoD can easily adopt.

As military electronic systems continue to increase in sophistication and capability, their cost and development time also grow. Each year, the DoD acquires and operates numerous embedded systems, ranging from intelligence, surveillance, and reconnaissance sensors to electronic warfare and electronic signals intelligence systems. Depending on their CONOPS, embedded systems have different security requirements. Methodologies for securing embedded systems must be customizable to meet CONOPS needs.

To meet application-specific requirements while also reducing technology costs and development time, developers have started to use open-systems architectures (OSA). Because OSAs use nonproprietary system architectural standards in which various payloads can be shared among various platforms, technology upgrades are easy to access and implement. The DoD has thus directed all DoD agencies to adopt OSA in electronic systems [3]. However, adding security to OSA could interfere with its openness. As most current security approaches are ad hoc, proprietary, and expensive, they are incompatible with OSA principles, especially when each payload developer individually implements and manages the payload security. Therefore, developing a system-level secure embedded system architecture that will seamlessly work with various OSA components is a challenge.

### Design Process

Embedded system CONOPS are developed from mission objectives and are used to derive both functional and security requirements. Researchers create, evaluate, and implement an initial system design, codeveloping functionality and security while minimizing security interference during functionality testing by decoupling security and functionality requirements. Several design iterations may be required before the mission objectives are met. Figure 1 captures the ideal process of designing a secure embedded system; the steps dedicated to security are highlighted in green.

To illustrate the secure embedded system design process, we use the design of a hypothetical UAS for a video surveillance application. The CONOPS of this example UAS application is as follows: At startup, the

UAS loads its long-term credentials for identification and authentication purposes. Mission-specific information—e.g., software, firmware, and data—is loaded into the respective memories. The system is then booted up and prepared for mission execution.

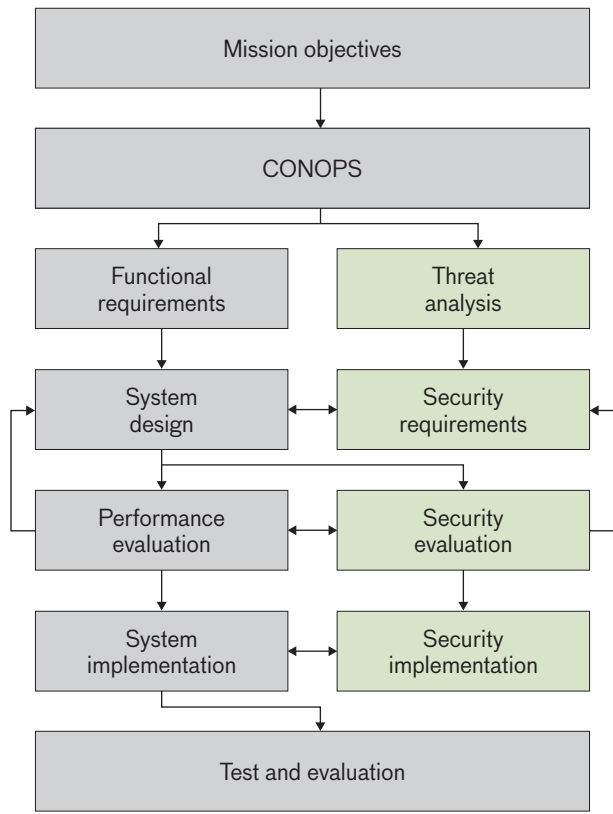
Figure 2 illustrates the UAS embedded system in its execution phase. Under the command of a ground control station, the UAS takes off, flies to its destination, and then collects video data. Video data containing target information are encrypted and broadcast to authorized ground stations (GT1 and GT2) via a radio. Raw video data are also saved for further processing after the UAS lands. When the UAS is shut down, both raw and processed video data are considered sensitive and must be saved securely. Any persistent state data, such as long-term credentials, must also be protected.

Figure 3 shows a high-level functional architecture initially designed for the example UAS embedded system. The architecture consists of a central processing unit (CPU) and a field-programmable gate array (FPGA) interconnected with a backplane network. The FPGA typically performs advanced video signal processing (e.g., for target detection and identification). The CPU handles command-and-control communications received from the ground control station and manages information (e.g., for target tracking).

Processing elements, such as the CPU and FPGA, must be chosen to securely deliver the UAS functionality requirements. This UAS application involves sophisticated signal processing and requires high throughput (measured by the number of floating-point operations per second) with a stringent SWaP allowance.

To support a complicated signal processing algorithm, the CPU needs a large memory and storage capacity. A popular mainstream processor likely has a variety of COTS software libraries that can be used in application development, but it may not have the security features desired for the CONOPS. On the other hand, a secure processor with built-in security features may simplify system development but may not possess the appropriate processing power or support the large memory space required for the application. We must consider system openness and upgradability before choosing a secure processor over a mainstream CPU.

Many popular FPGAs are built with embedded security features [4]. Developers should select these devices on the basis of their ability to encrypt and authenticate configuration bitstreams, incorporate security



**FIGURE 1.** In an ideal secure embedded system design process, functionality (gray) and security (green) are co-designed, yet they are appropriately decoupled during testing so that security does not interfere with functionality. This co-design is often difficult to achieve because functionality and security are two very different disciplines.

monitors to detect attacks, and erase decryption keys (a process known as zeroization) to protect critical information when attacks are detected.

**Threat Analysis**

The first step in designing a secure system is to analyze the potential attacks that the system may be subjected to when deployed. Adversaries seek to sabotage and develop countermeasures against U.S. missions, so the CONOPS determines not only functional requirements but also potential adversary attacks. The attacks depend on the adversary’s capability (e.g., a nation state’s sophisticated knowledge) and objectives (e.g., to exfiltrate information).

In the UAS example, we assume that there is a high probability of equipment loss resulting from the small size of the UAS and its operation in hostile areas. The examples of UAS attack targets in Figure 4 portray three logical

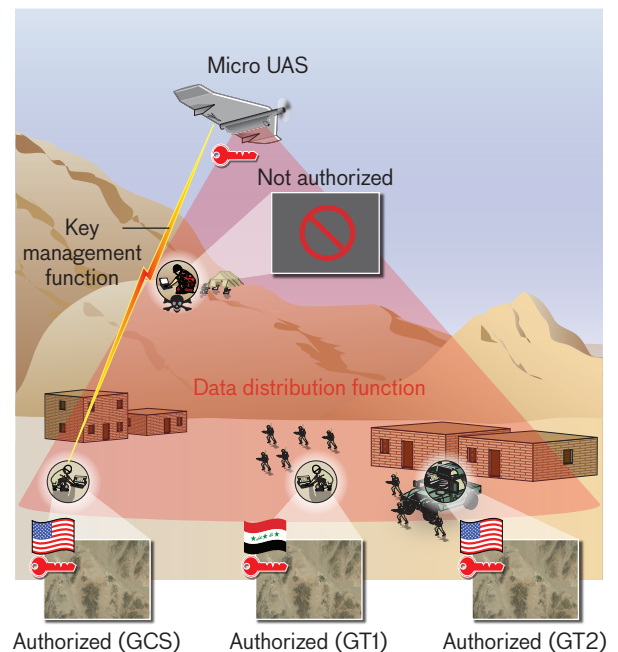
attack surfaces—boot process, system data, and software—and one physical attack surface, its physical system, that adversaries may attack to exfiltrate information.

During the CPU boot process, a secure system must establish a root of trust, which consists of hardware and software components that are inherently trusted, to protect and authenticate software components. Current practice uses the trusted platform module (TPM), an international standard secure processor that facilitates secure cryptographic key generation, remote attestation, encryption, decryption, and sealed storage [5]. Each TPM chip includes a unique secret key, allowing the chip to perform platform and hardware device authentication.

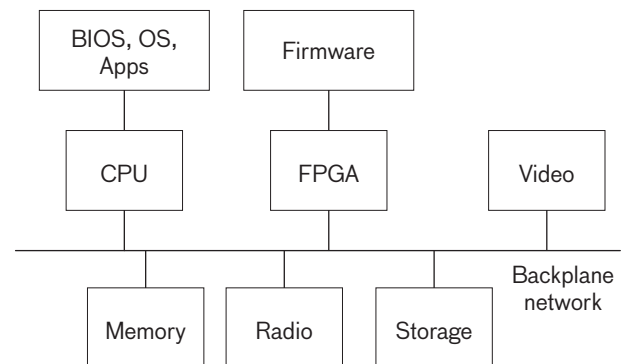
When creating the TPM, developers make a number of compromises that address cost and privacy concerns to ensure commercial adoptability of the module by vendors. The TPM must be inexpensive and cause as little disruption to the processing architecture as possible. Consumer privacy concerns dealing with user identification force module usage to be an optional and passive part of a processing system's operations. These compromises lead to a low-performance module that lacks adequate physical protection. In the "Architecture and Enabling Technologies" section, we will explain Lincoln Laboratory's security coprocessor that is equipped with a physical unclonable function, which was developed to address the TPM security inadequacy in tactical operations.

Despite a system's incorporation of an effective TPM, adversaries may exploit latent vulnerabilities within an authorized software component to access critical data or gain control of the platform itself. Even authorized users could deliberately or negligently introduce threats onto a system via untrusted software (e.g., malware) or unwanted functionality via third-party intellectual property.

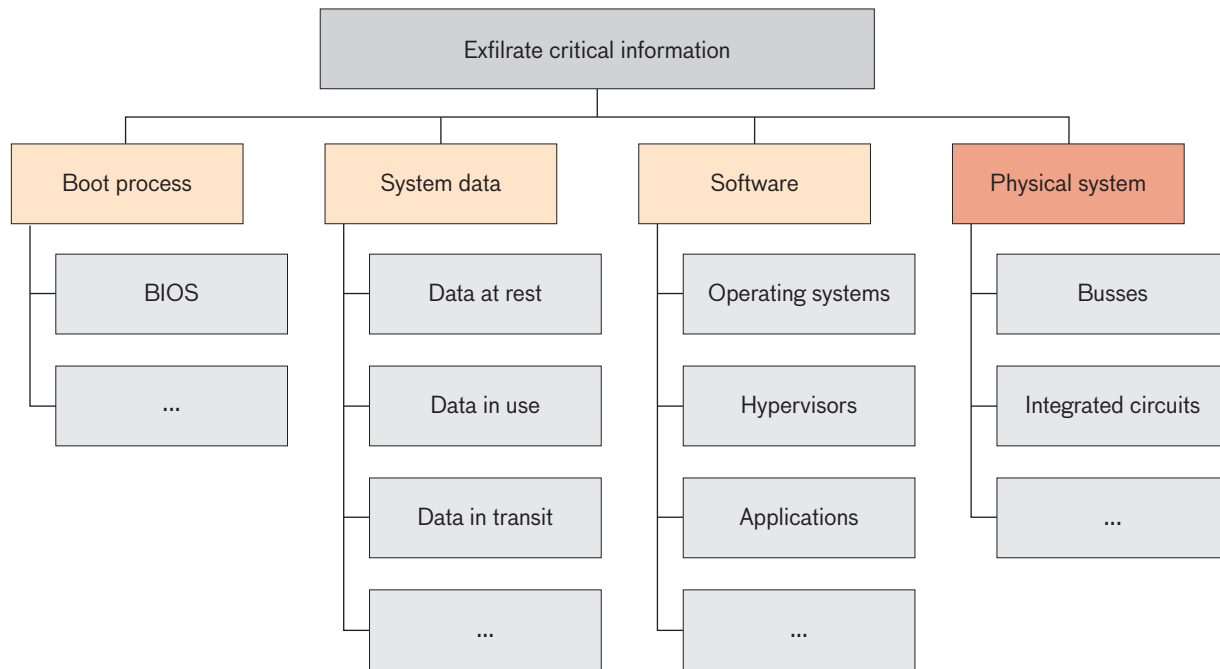
A secure system must be designed to prevent compromised software from giving an attacker unrestricted system access. Some developers are starting to address access issues on commercial systems. For example, software developers use separation kernels to establish and isolate individual computing processes, control information flow between the processes, and prevent unauthorized information access. On the hardware side, researchers are developing architectures that enforce isolations between processing threads executing on the same processor [6].



**FIGURE 2.** In this example of an unmanned aircraft system (UAS) application in its execution phase, the intelligence collected by the UAS needs to be shared by coalition partners yet protected from adversaries. Cryptography is the key technology enabling this operation.



**FIGURE 3.** This example of an unmanned aircraft system's embedded system functional architecture includes the central processing unit (CPU) that is supplied with a basic input/output system (BIOS), operating system (OS), and mission-specific application code (Apps). The field-programmable gate array (FPGA) has its configuration stored in a firmware memory. In addition to a video camera payload, the system has a random-access memory, a hard drive for storage, and a radio, all of which are accessible by the CPU and/or FPGA through a backplane network.



**FIGURE 4.** Example unmanned aircraft system (UAS) attack targets illustrate the vulnerabilities and sources of a threat scenario with three attack surfaces (boot process, system data, and software) and one physical attack surface (physical system).

Because the UAS is built with minimal system software for dedicated purposes, the exploitation of software vulnerabilities may be less likely than that for a general-purpose computer. The UAS has a strictly controlled provisioning environment accessible by a very limited number of authorized users, reducing the risk of introducing unverified and untrusted software into the UAS. However, one should always assume that an adversary will attempt to eavesdrop on wireless communication; thus, data protection is a high security priority.

Developers must also consider physical attacks because there is a high probability that adversaries will gain physical access to a UAS device, allowing enemies to reverse engineer the device or modify sensitive components in order to leapfrog their own technology or to gain unauthorized access to intellectual property. The most popular protection technique to date is the use of a strong protective enclosure equipped with electronic sensors to detect unauthorized accesses. However, because some systems are deployed and unattended for extended periods of time, it is challenging to maintain the standby power necessary for intrusion detection and response.

Developers must consider all threats and protect the confidentiality and integrity of the UAS data existing in three forms: data in use, data at rest, and data in transit. Various hardware and software solutions, most based on cryptography, are available à la carte. However, cryptographic technology must be fully integrated with the processor for efficient data protection via secure key management.

**Security Metrics**

Specifying and measuring security requirements for embedded system development are difficult. The requirements of the CIA triad for embedded systems are excellent objectives but are too abstract to be used as measurable security metrics to evaluate an embedded system during the design process. We have thus created three practical security metrics to facilitate the design of a secure embedded system: trustworthiness, protection, and usability. These metrics do not support absolute measurements but provide parameters to guide the design of embedded system security as the system’s mission functionality architecture evolves. In addition, multiple system architectures can be qualitatively evaluated and

compared to determine relatively how well they provide security. Because these metrics are qualitative and subjective, each security decision must include sufficient justification and documentation. Developers evaluate each metric by analyzing system functionality and security against a specific CONOPS. For example, developers will measure trustworthiness and protection on the basis of the system's current defense mechanisms compared with the level of defense that the CONOPS requires. If the system is lacking in defense, it will be less trustworthy and unable to adequately protect information.

Trustworthiness is a qualitative analysis of the system's effectiveness in defending against potential threats relevant to its CONOPS. On the basis of current system design and system information fidelity, developers have a certain level of trust in system behavior during an attack. For example, if a system is equipped with a defense mechanism against a certain threat, the system's security and trustworthiness likely improve. While unpatched system vulnerabilities reduce security, understanding those vulnerabilities enables developers to add protection technology to the design.

The protection metric is a qualitative analysis of the system's capability to support added-in protection technologies and address vulnerabilities identified in a CONOPS. Together, the trustworthiness and protection metrics can be used to measure how well a system's security addresses confidentiality and integrity requirements.

Usability is a qualitative analysis of the system's suitability to a task. A system that is highly secure but incapable of delivering the required functionality is not designed well. Usability metrics evaluate a system's design by considering the system's throughput, resilience, portability, upgradability, SWaP, and other similar parameters.

A system's processing requirements, threats, and protection needs vary during the course of a system's operation. To evaluate a system during operation, we examine four phases:

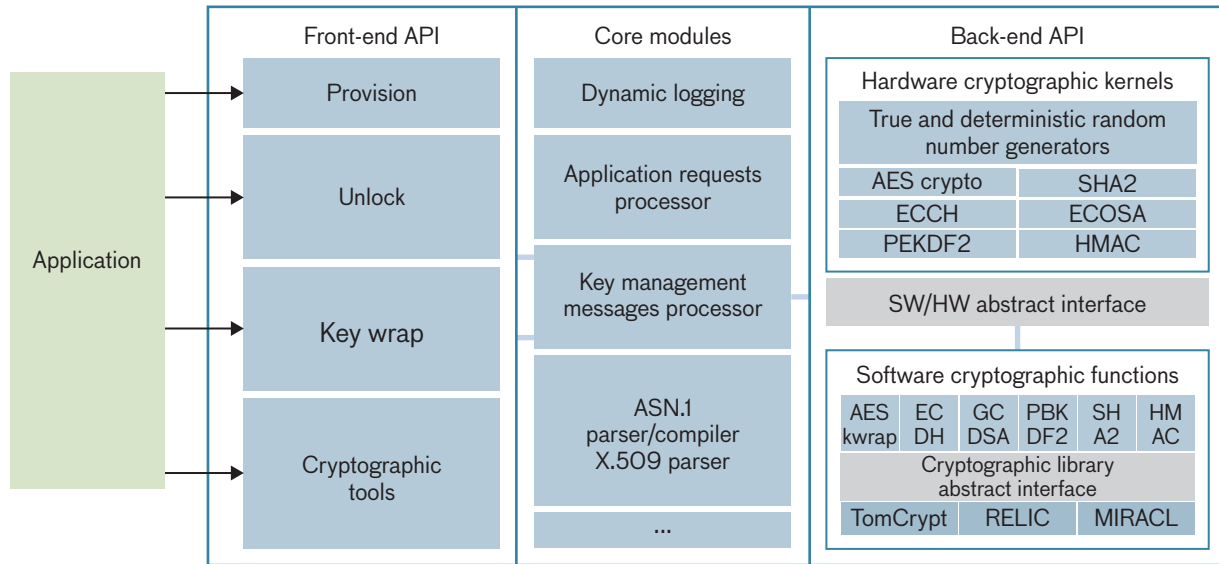
1. **Startup:** The system is being booted into a state suitable for operations; a trusted computing base (TCB), the set of components that provide the system with a secure environment, is established.
2. **Execution:** The system is in the operational state and performs functions required by the mission.
3. **Shutdown:** The system is in the process of turning off.
4. **Off:** The system is powered down.

## Architecture and Enabling Technologies

Because the critical information of a COTS-based embedded system is mostly in the system's software and firmware, cryptography is the foundation of the system's overall security. Many efficient, secure building blocks, such as the National Security Agency-approved Suite B cryptography [7], can be implemented with software, firmware, or hardware and are often obtainable as open-source intellectual property. However, simply using standard cryptographic primitives cannot guarantee the adequate implementation of security functions. Encryption effectiveness is based on the manner in which the cryptographic primitives (low-level cryptographic algorithms) are assembled and coordinated into the desired application-specific security functions. Encryption effectiveness also depends on key management, which includes the generation, distribution, and protection of keys.

Lincoln Laboratory has developed a solution to address encryption key management: Lincoln Open Cryptographic Key Management Architecture (LOCKMA), a highly portable, modular, open software library of key management and cryptographic algorithms that are suitable for embedded system uses. Designed to secure systems used in a wide range of missions, LOCKMA provides user, identity, and key management functions, as well as support for hardware and software cryptographic primitives, including the Suite B cryptographic primitives. LOCKMA has an intuitive front-end application programming interface (API) so developers can easily access LOCKMA's core functionality. To use LOCKMA, developers are not required to have advanced knowledge of the cryptography or key management algorithms implemented by LOCKMA's core modules; instead, they simply use the API to create security functions. LOCKMA handles the processing of key management messages and makes extensive use of cryptographic primitives available in several commercial and open-source libraries. Figure 5 shows LOCKMA's interfaces as high-level security functions and low-level cryptographic primitives.

Because software-implemented security functions may not meet extreme SWaP requirements, Lincoln Laboratory has implemented LOCKMA in a security coprocessor (S-COP), which applies cryptographic primitives in hardware. The benefits of hardware implementation over software implementation include much faster computation

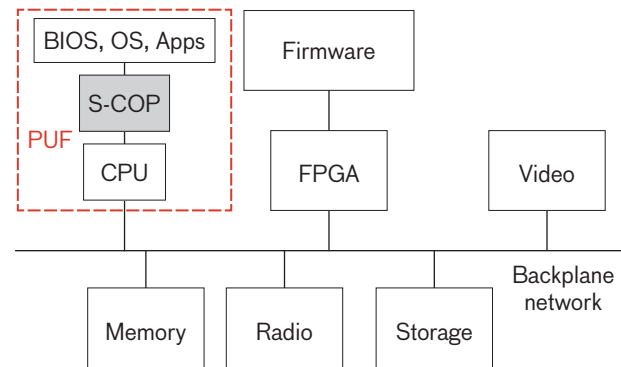


**FIGURE 5.** The LOCKMA software provides a front-end application programming interface (API) for high-level security functions that application developers can use directly. Complicated cryptographic algorithms are captured as core modules, which are hidden from application developers. The back-end API supports the use of low-level cryptographic kernels implemented in either hardware or software.

times, lower power consumption, hardware separation, and thus protection of sensitive keys from nonsensitive data and code.

Figure 6 shows the UAS embedded system architecture, previously shown in Figure 3, in which the CPU is secured with an S-COP and a physical unclonable function (PUF), which is a unique function that can be easily evaluated but hard to duplicate. The S-COP employs dynamic key management and accelerated Suite B cryptography for the authentication steps necessary to securely boot the CPU. The PUF provides an inviolable root of trust from which a unique cryptographic key is derived.

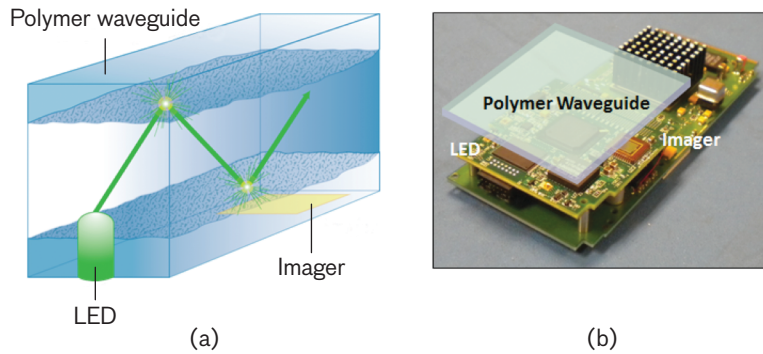
Lincoln Laboratory researchers have developed an optical PUF that can be implemented on a fully fabricated printed circuit board (PCB). As illustrated in Figure 7, the PUF is constructed by adding one or more light-emitting diodes (LED) and an imager to the PCB, which is then coated with a thin polymer planar waveguide. Upon powering up, the S-COP derives a unique numerical code from the imager, which receives light that is emitted by the LEDs and travels through the waveguide. This code is then used for device identification and key derivation. Manufacturing variations ensure a unique identification code for each PCB. Invasive attempts to learn about the PUF code (e.g., for



**FIGURE 6.** A security coprocessor (S-COP) is used along with a physical unclonable function (PUF) to secure a commercial off-the-shelf central processing unit (CPU).

cloning or other unauthorized actions), even when the PCB is unpowered, will disturb and damage the coating and irreversibly destroy the PUF code.

Because many environmental conditions, such as temperature and aging, can cause the PUF reading to vary, a technique called fuzzy extraction is employed to ensure that the same key will be derived from the PUF under various conditions [8]. This technique allows the S-COP to secure the boot process, load only trusted



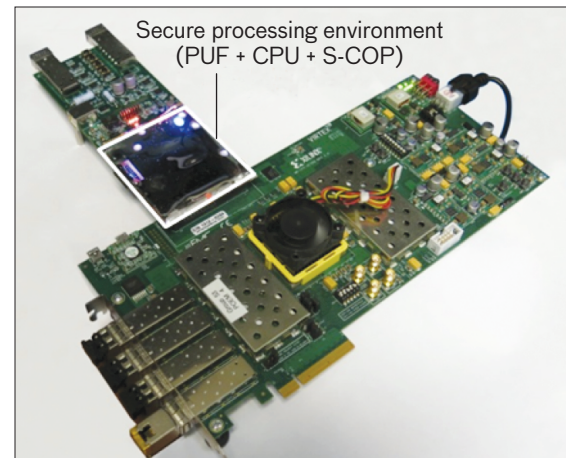
**FIGURE 7.** An optical physical unclonable function (PUF) is implemented with a waveguide. An operating concept illustration is shown in (a); implementation of the concept on a fully fabricated printed circuit board is shown in (b).

software, and confirm that the unique identity is intact before, during, and after the boot process. In addition to protecting data at rest with cryptography, the S-COP uses key management to support secure communications between subsystems to protect data in transit.

This S-COP-based secure embedded architecture allows software applications to be developed and tested initially without invoking security features. When a system is provisioned for deployment, developers apply the PUF to its PCB and load the finalized software code encrypted with the PUF-derived key. An incorrect PUF code will cause a failed software decryption, and the system will not start. The decoupling of the S-COP and the CPU allows DoD embedded systems to leverage mainstream CPUs, enhancing system usability and upgradability.

Figure 8 shows a test bed that we have developed to evaluate the S-COP-based secure architecture. In an unsecured architecture, the CPU reads in the basic input/output system (BIOS) and bootstraps the operating system (OS). Without authentication, the CPU is vulnerable to a maliciously modified BIOS and OS.

The S-COP-based secure architecture addresses this vulnerability by authenticating the BIOS, OS, and applications, as illustrated in Figure 9. When the embedded system powers up, the S-COP halts the CPU while the S-COP performs authentication. S-COP first reads the PUF and derives a key, which is used to decrypt the BIOS. If the decryption is successful, the CPU is released to execute the BIOS. The S-COP then authenticates and decrypts the OS and boots the system. Encrypted applications are loaded and handled in the same manner. In addition to associating an application with a designated system, the system can use LOCKMA key management to dynamically and seamlessly adjust the authorization of application execution (e.g., in time-specific and/or location-specific modes).

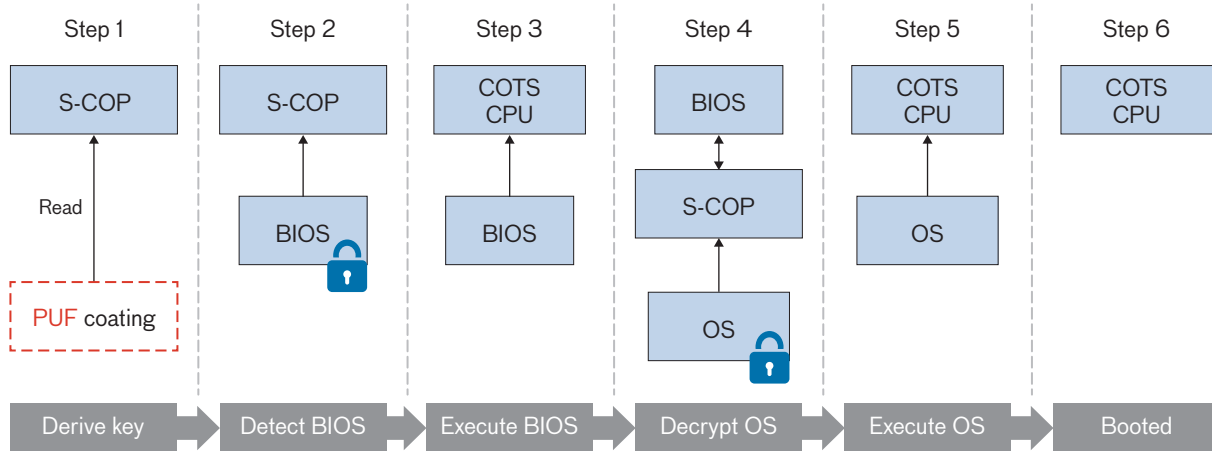


**FIGURE 8.** A secure processing environment integrates a central processing unit (CPU), a security coprocessor (S-COP), and a physical unclonable function (PUF).

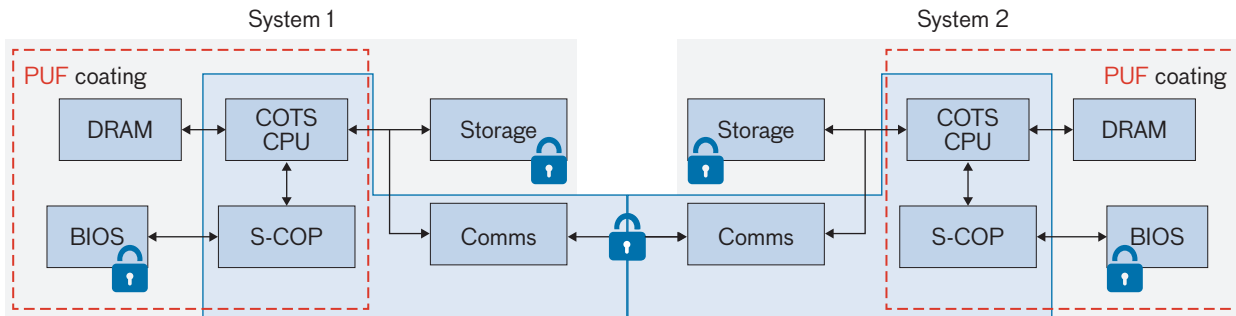
Figure 10 shows data-at-rest and data-in-transit protection enabled by the S-COP. In System 1 and System 2, the S-COP encrypts the CPU-generated data before they are stored, thus protecting them from unauthorized access. Likewise, the S-COP decrypts stored data before sending them to the CPU. Figure 10 also shows the concept of using S-COPs to protect data in transit between two systems by establishing an encrypted communication channel over which encrypted data can flow.

## Evaluation

In terms of the CIA triad, the S-COP addresses confidentiality and integrity by protecting the boot process, data, and communication channel from unauthorized access and alteration. The S-COP itself does not fully ensure a system's availability, but the decoupling of functionality and security, which allows for the use of a mainstream



**FIGURE 9.** During the secure boot process, the central processing unit (CPU) is halted until the security coprocessor (S-COP) successfully verifies system integrity. Data that are protected by encryption are indicated by lock symbols.



**FIGURE 10.** The security coprocessor (S-COP) enables data-at-rest and data-in-transit protection. Data that are protected by encryption are indicated by lock symbols.

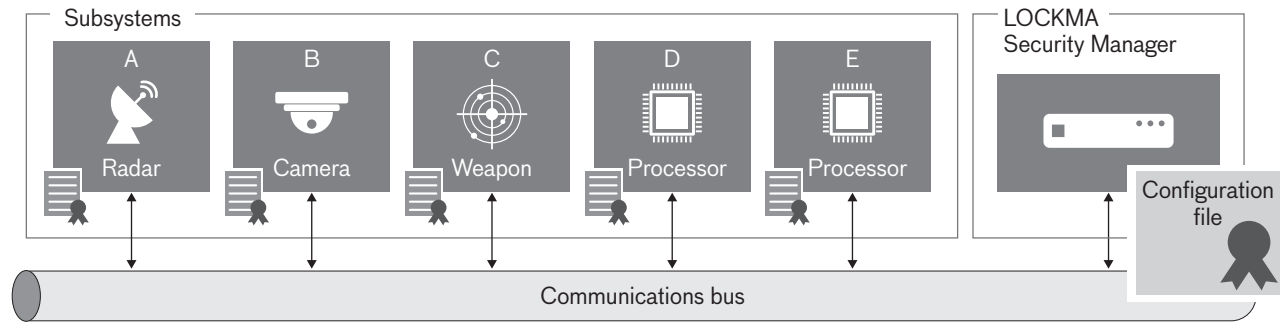
CPU, results in improved system usability. The system can be adapted to support other agility and resilience measures, such as moving target technologies [9]. As an example, we evaluate the hypothetical UAS embedded system with an S-COP-based secure architecture by using the same three security metrics: trustworthiness, protection, and usability.

A mainstream unsecured CPU receives low trustworthiness ratings during all system operation phases, as we assume that it needs an inherently large trusted computing base (TCB) and lacks hardware-enforced boot attestation. The security of such a CPU enhanced with an S-COP dramatically increases across all system operational phases, earning the CPU increased trustworthiness ratings. However, during the execution phase, the user still needs to trust the OS, which may have inherent vulnerabilities. The trusted boot does not completely eliminate

the risk of running untrusted or unverified codes that could potentially be exploited by attackers to escalate user privileges on the system or exfiltrate information.

If a CPU has no explicit support for physical protection, it will receive low protection ratings during the boot phase. Although the integration of a CPU with a TPM provides key storage and security measurements, the OS still needs to obtain, use, and revoke cryptographic keys, thus increasing the number of security components in the TCB. A lack of overall support for physical protection or for hardware-enforced encryption of code and data allows attackers to snoop or modify memory in the execution phase. During the off phase, the TPM could be physically replaced, and thus a new set of measurements could be inserted into the system. The S-COP-based secure architecture mitigates these deficiencies by creating a root of trust with a PUF and can be used to support physical protection.





**FIGURE 11.** In a LOCKMA-based open-systems architecture security framework, the LOCKMA security manager (LSM) checks subsystem credentials against a config file to ensure that the configuration is authorized.

Because the S-COP can be adapted to secure a mainstream CPU, the usability of the secure UAS embedded architecture rates high; the architecture can leverage all the benefits of a COTS CPU, such as high performance (e.g., for signal processing), large cache and memory support, and widely supported software libraries.

### Open-Systems Architecture Security

The use of OSAs can improve the development and life-cycle efficiency of system assets. Typically, OSAs incorporate several buses with well-defined interfaces for communication between components. A system can then be adapted to different needs by providing proper components and defining system interconnections.

Besides securing the CPU, LOCKMA is being developed into a cryptography-based secure framework that has been successfully demonstrated in OSA embedded system protection. The framework employs LOCKMA to provide encryption of data in use, data in transit, and data at rest to prevent eavesdropping, probing, and unauthorized access. In addition, developers can enforce a trusted configuration by accepting only predetermined payloads and preventing unauthorized hardware and/or software substitutes.


Figure 11 illustrates an example configuration that consists of several payloads and processors and a LOCKMA security manager (LSM). A digitally signed configuration (config) file that specifies authorized payloads, acceptable combinations of payloads, and secure communication channels establishes the authorized mission configuration. Figure 12 shows an example config file that has three sections: principals, constraints, and channels. The authorized subsystems are listed under the principals section; authorized configurations are noted under

```
# Principals
A, B, C, D, ...

# Constraints
A or B
A and D
...

# Channels
Channel 1:
Pub: A
Sub: D, E
...
```

Digital signature

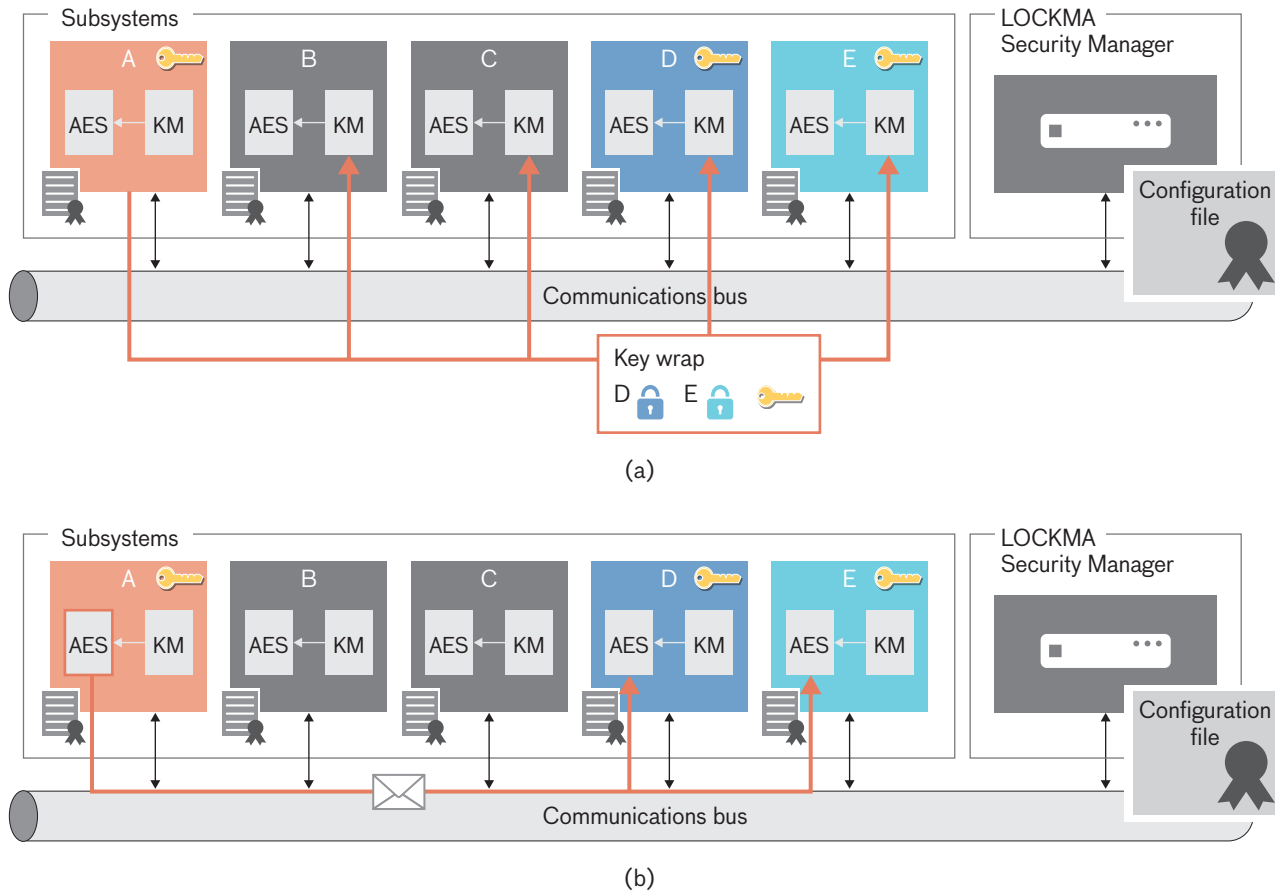


**FIGURE 12.** A security config file, an example of which is shown above, is used to enforce payload authorization and secure communication channels.

the constraints section; and authorized communication channels are specified in the channels section. In this example, the system can contain subsystems A, B, C, and D, among others. An authorized configuration is one that includes subsystem A or subsystem B with both subsystems A and D present. Subsystem A is given the role of publisher (pub) and subsystems D and E are assigned the role of subscriber (sub). A digital signature is created for the config file so that its integrity can be verified.

At startup, the LSM verifies the digital signature of the config file and ensures that it is unaltered. Using the config file, the LSM collects subsystem credentials and confirms the absence of unexpected system payloads, leading to authorized system configuration. The system then starts and the LSM continues to set up secure communication channels.

Figure 13 illustrates how LOCKMA enables each subsystem with a key management (KM) function and an Advanced Encryption Standard (AES) encryption and decryption function. Subsystem A creates a key wrap containing a symmetric cryptographic key that is



**FIGURE 13.** In a LOCKMA security framework, a publisher (e.g., subsystem A) sends a key wrap only accessible by intended subscribers (e.g., subsystems D and E) to retrieve a session key (a). The publisher and subscribers are then able to carry out encrypted communication (b). Each subsystem contains an Advanced Encryption Standard (AES) and a key management (KM) function.

only accessible by authorized subsystems D and E, and establishes a communication channel. The channel users retrieve the common secret session key and use it for encrypted communications. The system is then ready to perform its mission objectives.

**Ongoing Work**

Security has an asymmetric nature—an attacker can compromise a system by discovering a single, unexpected vulnerability, while a defender must defend against all vulnerabilities. Because it is impossible to correctly predict every future attack, securing an embedded system to prevent attacks is not a guarantee of mission assurance. Being secure is not adequate; systems must also be resilient. Lincoln Laboratory is vigorously pursuing an answer to the essential mission-assurance question: If an attacker is successful

despite implemented defenses, what can be done so the mission can continue until completion?

Our objective is to define a standardized reference secure and resilient architecture for DoD embedded systems. We want to ensure that systems continue to function when a situation does not go as we expect. Our work is guided by the four stages of actions involved with the resiliency of an embedded system against cyber attacks: anticipate, withstand, recover, and evolve [10]. Our current research and development focuses on approaches that enable a system to defend against threats; withstand attacks and complete mission goals; recover from a degraded state and return to a normal state; and evolve to improve defense and resilience against further threats.

Our ongoing work also includes the development of mission-level resiliency metrics to answer the following question: Is the mission more likely to be successful

when our system is used? A system specification such as system restart time is a good design objective, but by itself does not provide information about system availability and mission assurance. We are developing a systematic approach to connect mission-level resiliency metrics to system specifications.

### Acknowledgments

The authors would like to acknowledge technical contributions from the team members: Thomas Anderson, Walter Bastow, Robert Bond, Brendon Chetwynd, Robert Cunningham, Ford Ennis, Benjamin Fuller, Michael Geis, Karen Gettings, Antonio Godfrey, Raymond Govotski, Kyle Ingols, Eric Koziel, Joshua Kramer, Theodore Lyszczarz, and Merrielle Spain. ■

### References

1. T.M. Takai, "Department of Defense Instruction, Subject: Cyber Security," 14 Mar. 2014, available at [http://www.dtic.mil/whs/directives/corres/pdf/850001\\_2014.pdf](http://www.dtic.mil/whs/directives/corres/pdf/850001_2014.pdf).
2. D.R. Martinez, R.A. Bond, and M. Vai, eds., *High Performance Embedded Computing Handbook: A Systems Perspective*. Boca Raton: CRC Press, 2008.
3. Department of Defense Open Systems Architecture Data Rights Team, "DoD Open Systems Architecture Contract Guidebook for Program Managers V. 1.1," June 2013, available at [http://www.acqnotes.com/Attachments/Open%20System%20Architecture%20\(OSA\)%20Contract%20Guidebook%20for%20Program%20Managers%20June%202013.pdf](http://www.acqnotes.com/Attachments/Open%20System%20Architecture%20(OSA)%20Contract%20Guidebook%20for%20Program%20Managers%20June%202013.pdf).
4. T. Huffmire, C. Irvine, T.D. Nguyen, T. Levin, R. Kastner, and T. Sherwood, *Handbook of FPGA Design Security*. New York: Springer, 2010.
5. Trusted Computing Group, "How to Use the TPM: A Guide to Hardware-Based Endpoint Security," 2009, available at [http://www.trustedcomputinggroup.org/files/resource\\_files/8D42F8D4-1D09-3519-AD1FFF243B223D73/How\\_to\\_Use\\_TPM\\_Whitepaper\\_20090302\\_Final\\_3\\_.pdf](http://www.trustedcomputinggroup.org/files/resource_files/8D42F8D4-1D09-3519-AD1FFF243B223D73/How_to_Use_TPM_Whitepaper_20090302_Final_3_.pdf).
6. Intel, "Software Guard Extensions Programming Reference," Sept. 2013, available at <https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf>.
7. National Security Agency, "Suite B Cryptography," 25 Sept. 2014, available at [https://www.nsa.gov/ia/programs/suiteb\\_cryptography/](https://www.nsa.gov/ia/programs/suiteb_cryptography/).
8. M. Spain, B. Fuller, K. Ingols, and R. Cunningham, "Robust Keys from Physical Unclonable Functions," *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2014, pp. 88–92.
9. H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein, "Finding Focus in the Blur of Moving-Target Techniques," *IEEE Security & Privacy*, vol. 12, no. 2, 2014, pp. 16–26.
10. D.J. Bodeau and R. Graubart, "Cyber Resiliency Engineering Framework," MITRE Technical Report, document number MTR110237, Sept. 2011, available at [https://www.mitre.org/sites/default/files/pdf/11\\_4436.pdf](https://www.mitre.org/sites/default/files/pdf/11_4436.pdf).

### About the Authors



**Michael Vai** is a senior staff member in the Secure Resilient Systems and Technology Group. From 2012 to 2015, he served as an assistant leader of this group. Previously, he was an assistant leader of the Embedded and Open Systems Group in the Intelligence, Surveillance, and Reconnaissance and Tactical Systems

Division. He has worked in the area of embedded systems and technology for more than 25 years, leading the development of advanced embedded systems and publishing his research extensively. Prior to joining Lincoln Laboratory in 1999, he was on the faculty of the Department of Electrical and Computer Engineering at Northeastern University. During his tenure, he conducted multiple research programs funded by the National Science Foundation, Defense Advanced Research Projects Agency, and industry. His current research interests include secure and resilient embedded systems and technology, particularly systems involved in tactical operations. He received his master's and doctoral degrees from Michigan State University, in electrical engineering.



**David J. Whelihan** is a technical staff member in the Secure Resilient Systems and Technology Group. He started at Lincoln Laboratory in 2002 as a summer intern working on advanced radar processing hardware while earning his doctorate in electrical and computer engineering from Carnegie Mellon University.

After graduating in 2004, he joined the startup company DAFCO, where he led the development of an advanced application-specific integrated circuit and field-programmable gate array debug tool. He then worked for a hardware cyber security startup. Prior to graduate school, he spent three years at Intel as an architectural validation engineer working on next-generation microprocessors. In 2010, he rejoined Lincoln Laboratory as a technical staff member, leading the Laboratory's efforts in the Defense Advanced Research Projects Agency's Photonically Optimized Embedded Microprocessors program and serving as the lead architect on the Self-Contained High-Assurance MicRO Crypto and Key-Management Processor system, which won a 2012 MIT Lincoln Laboratory Best Invention Award. His current work involves several programs dealing with secure and resilient hardware and software systems.



**Benjamin R. Nahill** is a technical staff member in the Secure Resilient Systems and Technology Group, specializing in secure hardware and embedded systems. He works on solving problems in secure processing and data protection by leveraging novel hardware architectures. Prior to joining the Laboratory in 2014, he

worked in a variety of areas, including embedded system design in resource-constrained environments, satellite communications, and signal processing. He received a bachelor's degree in computer engineering and a master's degree in electrical engineering, both from McGill University.



**Daniil M. Utin** is a technical staff member in the Secure Resilient Systems and Technology Group. Previously, he cofounded several successful Internet technology and gaming-related companies, including WorldWinner.com and Cambridge Interactive Development Corporation. He has substantial expertise

in developing secure, scalable, high-transaction-volume software systems coupled with dynamic, rich front-end graphical user interfaces. He joined Lincoln Laboratory in 2009 and has been designing cryptographic key management systems and utilizing his commercial software background to lead the development of secure usable solutions. He earned his bachelor's and master's degrees in computer science from Brandeis University.



**Sean R. O'Melia** is a technical staff member in the Secure Resilient Systems and Technology Group. He joined Lincoln Laboratory in 2008. His work to support secure communications in tactical networks includes the development of cryptographic key management technology for small unmanned aircraft

systems. Most recently, he has concentrated on the design of a key management architecture for future tactical satellite communications capabilities and the development of security for open-architecture airborne platforms. He received bachelor's and master's degrees in computer engineering from the University of Massachusetts, Lowell. His graduate research focused on instruction set extensions for enhancing the performance of symmetric-key cryptographic algorithms.



**Roger I. Khazan** is the associate leader of the Secure Resilient Systems and Technology Group. He has led programs and conducted research in the areas of systems and communication security. His expertise and research interests are in usable security, key management and distribution, and applied cryptography,

with specific focuses on disadvantaged tactical environments and embedded devices. He is passionate about creating technology that significantly simplifies the task of adding strong and usable cryptographic protections to software and hardware applications. He earned his master's and doctoral degrees from MIT in the Department of Electrical Engineering and Computer Science, and he completed a minor in management at the MIT Sloan School of Management. He earned a bachelor's degree from Brandeis University, with a double major in computer science and mathematics and a minor in economics.

# Secure and Resilient Cloud Computing for the Department of Defense

**Nabil A. Schear, Patrick T. Cable, Robert K. Cunningham, Vijay N. Gadepally,**

**Thomas M. Moyer, and Arkady B. Yerukhimovich**

Cloud computing offers substantial benefits to its users: the ability to store and access massive amounts of data, on-demand delivery of computing services, the capability to widely share information, and the scalability of resource usage. Lincoln Laboratory is developing technology that will strengthen the security and resilience of cloud computing so that the Department of Defense can confidently deploy cloud services for its critical missions.

» **Imagine a military commander who** urgently needs a specialized computing capability to analyze new intelligence, surveillance, and reconnaissance (ISR) data and integrate those data with existing ISR information. The commander directs his information technology (IT) staff and developers to design this capability. The staff quickly provision computing hardware from a Department of Defense (DoD) cloud and compose the software and services needed to ingest, enrich, create, and share knowledge from the data while ensuring that the resulting capability remains secure and resilient (i.e., able to continue operations after a disruption). Within days, the staff has an initial system for analyzing the ISR data up and running. In the following weeks, they enhance the system by creating new features and adding capacity for even more data. This vision for agile, inexpensive cloud computing could revolutionize the way the DoD operates, and Lincoln Laboratory is building the next-generation secure cloud computing systems that could enable that vision.

Marketers have made the term *cloud* synonymous with ubiquitous, convenient computing. Digging below this simplified description, we find that cloud computing is a model for deploying software and hardware resources at lower cost and with greater flexibility than deploying typical enterprise computing resources. The defining attributes of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity (i.e., ability to adapt quickly to changing

computational demands), and measured service (i.e., accounting and billing of resource usage) [1]. In cloud computing, computation and software capabilities are outsourced to a provider that delivers services to a cloud user (also called a tenant).

The DoD is looking to the cloud computing model as a means for lowering the costs and improving the flexibility of computing systems while delivering more capable services. But, the process of moving to the cloud is not without peril. The 2013 Defense Science Board (DSB) Report of the Task Force on Cyber Security and Reliability in a Digital Cloud recommended that the “DoD should pursue private cloud computing to enhance mission capabilities, provided that strong security measures are in place” [2]. The DSB study team, including experts from Lincoln Laboratory, the DoD, commercial cloud providers (e.g., Google and Amazon), and leading universities, found shortcomings in the security and resilience of clouds. The DSB report further highlighted the need for research addressing conditions of interest to a warfighter, whose computing resources may face an active cyber adversary, intermittent connectivity, and physical attacks on computing hardware.

Today’s cloud providers and the technology that underpins their clouds are focused on the availability and scalability of services and not on DoD-specific security needs. Commercial cloud security is typically proprietary and thus opaque to tenants. For example, tenants have no visibility into cloud network security or data access. The prevailing cloud computing model is based on users trusting their cloud providers; data are stored unencrypted inside the cloud and all processing is done on unprotected data. The only enforceable guarantees that tenants have are through legal service-level agreements that loosely define the security responsibilities of both providers and users. This legal model does not provide tenants with timely and controllable mechanisms with which to respond when adversaries strike. As the DoD seeks to utilize commercial cloud technology, current cloud security will leave the DoD unable to protect their cloud resources from external attack, their cloud provider, insiders, or malicious tenants.

To address these shortcomings in cloud security, Lincoln Laboratory has undertaken the Lincoln Laboratory Secure and Resilient Cloud (LLSRC) effort to shore up the technology behind the cloud. The LLSRC approach is to (1) define a more accurate threat model for DoD cloud computing, (2) research and build technology

that addresses that threat model, and (3) integrate the technology into a usable, secure, resilient cloud test bed. Underpinning this work is the semitrusted cloud threat model, which is built on the assumption that some of the cloud infrastructure or resources will be under the control of an adversary, but that there remains a portion of the cloud that can be inspected and trusted.

Our research and prototyping efforts are focused on four key components needed for a secure and resilient cloud: communication, storage, processing, and a high-assurance architecture that holds them together. In each area, our goal is to achieve security and resilience in the semitrusted cloud threat model. The vision for this technology is to create an ecosystem of services and capabilities that allows the DoD to build secure, resilient cloud mission applications. We are developing services and interfaces that can be recomposed to meet mission needs. Finally, we are combining these prototypes and services in a cloud test bed that reduces the risks for the DoD’s acquisition of secure, resilient cloud technology by providing proofs of concept, technology maturity, integration demonstrations, and security evaluations.

### **Cloud: A Primer**

Cloud computing changes how information services can be created and implemented. Before the cloud era, providing a new computing service (e.g., a large website or a file server) meant substantial capital expenses for data center space, network connectivity, and servers. After the capital investment, companies needed large teams of IT personnel and developers to manually build, install, configure, and maintain the supporting infrastructure. It took months for the teams to field the new service and considerable expense to operate and maintain it. In the cloud model, computing resources can be created on demand and composed into applications quickly.

The National Institute of Standards and Technology (NIST) defines the cloud as a “model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1]. NIST’s five essential characteristics of a cloud service are contrasted in Table 1 with those from the “old way” of enterprise computing .

**Table 1. Cloud versus Enterprise Computing Characteristics**

CLOUD COMPUTING	ENTERPRISE COMPUTING
On-demand self-service	User request and long implementation to provide services
Broad network access	Limited to local area/company networks only
Resource pooling	Dedicated resources; expensive resilient hardware
Rapid elasticity	Fixed, over-provisioned capacity; expensive to scale up or down
Measured service	Poor metrics; unmeasurable guarantees

**Table 2. Cloud Service Models**

SERVICE MODEL	WHAT'S PROVIDED	FLEXIBILITY	EXAMPLES
IaaS	Compute, storage, and network services	High	Amazon Elastic Compute Cloud (EC2), DISA milCloud, Google Compute Engine, Microsoft Azure
PaaS	Application program interfaces (API) and services	Medium	Amazon Elastic MapReduce, MathWorks Cloud, Red Hat OpenShift
SaaS	Full-fledged applications	Low	Google Gmail, Microsoft Office 365, Facebook

Cloud offerings fit into three different service models—infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS)—that target system administrators, developers, and end users, respectively (see Table 2). NIST identifies four environments in which cloud services exist: public, community, private, and hybrid. Public clouds allow anyone to use them. Community clouds may share tenants across a particular community of users (e.g., a cloud managed by the Defense Information Systems Agency [DISA]). Private clouds are generally limited within an organization (and their broad network access may be more limited). Hybrid clouds allow the mixing of private and/or community clouds with public ones. Some organizations have a private cloud for general use but may need to scale to broader cloud services for specific needs during certain periods.

When building cloud applications for these service models and environments, developers and designers must consider distributed systems issues, such as consistency,

availability, and network failure. Similarly, cloud developers and system administrators need to resolve how to automate the deployment and how to monitor the availability of an elastic distributed system. These issues represent a considerable departure from the vast majority of enterprise computing patterns. Indeed, many DoD cloud initiatives thus far have not broken the mold of enterprise computing; they are simply performing virtualization at a distant data center. Solving the challenges presented by distributed systems will benefit the future of DoD software by providing increased resiliency to the end users' missions.

### Semitrusted Cloud Threat Model

Cloud services are attractive options for computing. They are easy to create, are usually straightforward to use, and offer flexibility and low cost; however, they carry significant security risks. Consider the example of Code Spaces [3]. In June 2014, attackers stole the credentials to the company's Amazon Web Services cloud account and

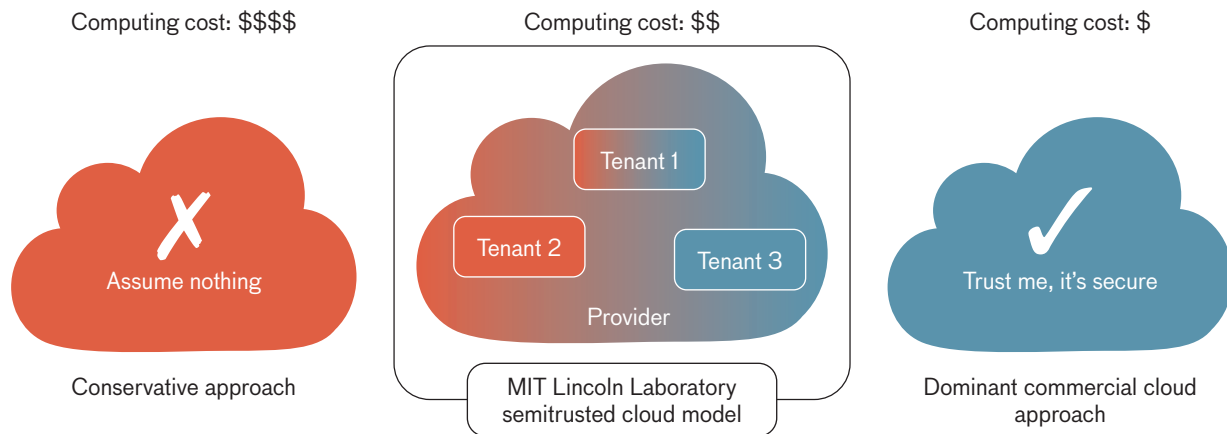
proceeded to destroy all Code Spaces’ virtual machines and customer data. Unable to recover, the company ceased operations shortly thereafter.

The first steps toward combating threats to the cloud are to understand and to codify assumptions about attacks and risk by examining the prevailing threat model. The dominant commercial cloud threat model is based on trusting the cloud providers and their system administrators. The layered service model of infrastructure, platform, and software as services allows a buyer of cloud services to abstract away the details of the lower layers. This model often results in security that is opaque to the end-user. For a number of reasons, security opacity is beneficial to providers. First, the providers’ infrastructure and associated mechanisms for the security of their offerings are their critical intellectual property that the providers are not incentivized to share. Second, by not specifying the details of their security implementations, providers are free to change the details as needed without violating any service-level agreement. Last, providers can espouse a shared security responsibility model in which attacks and vulnerabilities occurring at or below the level of the providers’ services are the providers’ responsibility, and attacks and vulnerabilities above the providers’ services are the responsibility of the cloud users. Rarely are sophisticated real-world attacks so cleanly separated across the layers of a computing stack; therefore, cloud providers can indemnify themselves of liability and blame the users for any security breaches that arise.

A further problem with the prevailing security model for clouds is that of mismatched priorities and control. Because cloud service providers require their users to outsource security to the providers, users must also give up control of how to respond to an attack, thereby allowing providers to both prioritize and formulate the responses. With only vague security service-level agreements in place as leverage, cloud users are at the providers’ mercy when an attack happens.

An alternate threat model is one in which cloud providers are not trusted at all because they are a third party to users’ resources. This conservative approach is taken by some users in the DoD who are engaged in sensitive missions and also by many academic researchers, especially those working in cryptography. The assumption that the cloud is completely insecure leads to the use of very inflexible solutions (e.g., encrypting data and not processing them at all in the cloud) or extremely expensive operations (e.g., using fully homomorphic encryption that performs computation without decrypting data [4]). As a result of confining technology to this threat model, many of the benefits of cloud computing are lost. Furthermore, a conservative user may avoid using the cloud at all and fall back to single-tenant enterprise computing.

We choose neither of these extremes for the LLSRC threat model. Instead, we use a trust model that we call the semitrusted cloud threat model (see Figure 1). In the semitrusted model, we assume that some fraction of the cloud resources is under the control of adversaries. We



**FIGURE 1.** This comparison of cloud threat models shows how the semitrusted threat model Lincoln Laboratory advocates accepts a reasonable amount of risk to maintain the cloud computing benefits of low computational costs and flexible, scalable services.

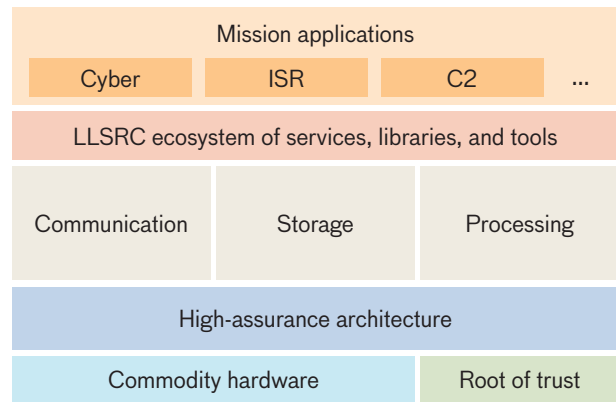


neither distinguish between external or insider attackers nor assume that we can always precisely identify which nodes are corrupted and which are unaffected. Applying this threat model, we expect and design for cloud compromise, but we assume that some trustworthy base of resources remains on which we can build secure and resilient systems.

### Secure and Resilient Cloud Architecture

Building a secure and resilient cloud system for the DoD necessitates some changes and additions to the standard cloud architecture. The LLSRC architecture stack (Figure 2) begins at the bottom with the same commodity, low-cost hardware present in today's clouds but with the addition a hardware root of trust, (i.e., a specialized cryptographic coprocessor that is trusted by the operating system). Atop that layer is a high-assurance trusted computing architecture that allows us to bootstrap (initiate) trusted cryptographic keys that will underpin higher layers. Because this layer also supports bidirectional control and visibility of the cloud infrastructure below, cloud tenants can obtain actionable situational awareness of the resources they are using. Above that trusted architecture, we build systems that enable the three core capabilities needed for cloud computing: communication, storage, and processing. We aim to develop systems that maintain security and resilience in the face of adversaries who control some of the cloud resources.

The LLSRC architecture fits both beneath and alongside existing insecure cloud software. As a result, we need a strategy for integrating LLSRC technology with the cloud services and applications that need to be secured. The LLSRC integration strategy is to utilize a suite of services and tools at various levels of the cloud stack. This strategy provides a tiered approach to integration that starts with limited-invasiveness, compatible solutions that can be deployed transparently to the applications. The next integration point is the security application program interfaces (API) and services that developers can compose. The final integration point consists of full replacements for end-user software. These integrations plug in roughly at the infrastructure, platform, and software layers to afford cloud tenants the maximum flexibility to design and compose appropriate solutions for their missions' needs.



**FIGURE 2.** At the base of the stack for the Lincoln Laboratory Secure and Resilient Cloud (LLSRC) are commercially available hardware and a root of trust. The high-assurance architecture allows the three core capabilities to take advantage of its security measures. The LLSRC application interfaces and the applications at the higher layers allow developers to design software for specific missions, such as intelligence, surveillance, and reconnaissance (ISR) or command and control (C2).

### High-Assurance Architecture

Today's cloud service providers do not furnish the building blocks necessary to establish a trusted environment for hosting mission-critical applications and data. Tenants have limited ability to verify the underlying platform when they deploy their software and data to the cloud provider and to ensure that the platform remains in a good state for the duration of their computation. Additionally, current practices restrict tenants' ability to establish unique, unforgeable identities for individual nodes that are tied to a hardware root of trust. Often, identity is based solely on a software-based cryptographic solution or unverified trust in the provider. What is needed are mechanisms to establish trusted cloud identities, rooted in hardware, and to maintain appropriate situational awareness and control over cloud nodes.

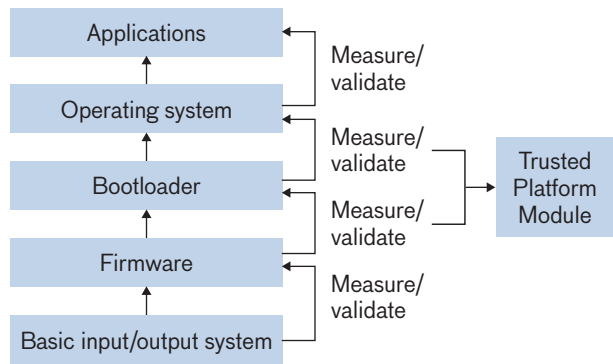
To establish a cryptographic node identity with a hardware root of trust, nodes validate their environment and, in turn, are validated by the environment before being issued a long-term identity. The hardware that is often used to establish a trusted environment in commodity systems is the Trusted Platform Module (TPM), a small cryptographic processor that provides a hardware root of trust. Figure 3 shows the process by which a system with a TPM will boot. This process, known as

secure boot, ensures that no files have been modified and halts the system when any unexpected changes are detected. The basic input/output system (BIOS) starts the process by measuring (or validating) the firmware within the system. The firmware then validates the boot loader, which in turn validates the operating system (OS). The OS then monitors the applications running on the system.

During normal operation, these collected measurements can be provided to remote systems as a means of proving, or attesting, the system’s integrity state, through a process known as integrity measurement. The TPM forms the hardware root of trust for secure boot and integrity measurement. This root of trust is expected to function correctly, and from this assumption, we can validate the entire set of applications running on the system, using the chained validation described in the preceding paragraph and Figure 3. These techniques ensure that any deviation from a known-good state can be detected so that appropriate responses can then be taken. This chained validation approach increases the difficulty for an adversary who is attempting to compromise a system while avoiding detection.

The first challenge to extending trusted computing to the cloud is virtualization. Because virtual machines are by definition separated from and unaware of the underlying hardware (e.g., the hardware of the virtual machine monitor, or hypervisor) on which they run, we need a way to tie the virtual environment to one rooted in hardware. The software Virtual Trusted Platform Module (vTPM) solves this problem by linking its attestations of the virtual environment to that of the underlying hardware [5].

The second challenge to trusted computing is effectively scaling the techniques to the thousands of nodes in the cloud. The sheer number of machines and limitations of the performance of hardware TPMs (e.g., a single digital signature, which is a fundamental building block in trusted computing, can take ~1 second to produce) make it infeasible to have each virtual machine attest to all other hosts with which it communicates directly. We propose using a cloud verifier to alleviate this problem by centralizing integrity measurement to a dedicated software service that verifies all the nodes belonging to a particular entity (tenant or provider) [6]. This integrity measurement asynchronously occurs separately from the communication channel and has no impact on the

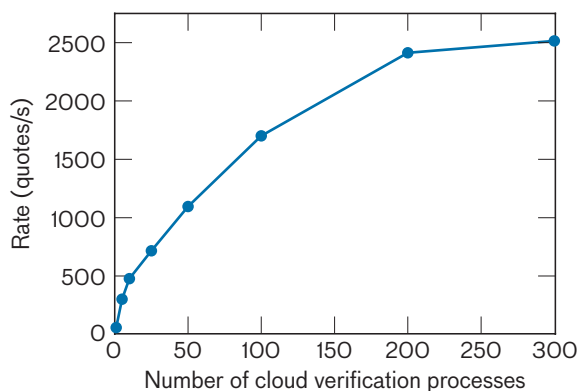


**FIGURE 3.** A trusted computing architecture validates the current environment during system boot to allow the system to generate proofs that show the current integrity state of the system. The validation is enabled by the Trusted Platform Module and sequences through all layers of the stack.

performance of the application’s communication. While centralizing the integrity measurement reduces the timeliness of detecting violations of a system’s integrity, the number of systems that can be attested is more scalable.

Using hardware-rooted node identities and the measurements collected by their individual cloud verifiers, tenants can now create and maintain their own individualized trusted membership lists. The long-term identities (i.e., public/private key pairs) in the trusted membership list bootstrap both long-term identity and ephemeral keys (temporary keys generated for each key establishment process) for higher-level services. Changes to a trusted membership list inform higher-level services of the trust level of each node in tenants’ environments. The trusted membership list forms the foundation and interface that enables secure communication, information flow tracking, and storage.

To demonstrate a method that allows applications on a cloud node to bootstrap trust, we created the Keylime library, a collection of services and an API for providing the cloud verifier (CV) service and the cloud node service that runs on each compute node. The ability to maintain a trusted list of cloud compute nodes depends on Keylime’s ability to continuously verify TPM quotes, i.e., cryptographically signed measurements of system integrity. To test the scalability of our library, we developed an automated test platform to create new virtual “compute nodes” and verify their integrity. Figure 4 shows that we were able to reach a rate of 2500 verifications (quotes) per



**FIGURE 4.** Lincoln Laboratory’s Keylime cloud verifier (CV) can quickly detect compromised cloud nodes and scale to continuously monitor 1000s of nodes. By increasing the parallelism of the CV (thus increasing the number of processes performed), we observed a sustained rate of 2500 quotes verified per second.

second. These results indicate that our system can scale to handle 1000s of nodes with low-latency detection of integrity violations.

### Secure Communication

Cloud providers, malicious insiders, or other tenants mounting a side-channel attack<sup>1</sup> may eavesdrop on cloud networks. To protect data in transit, we can rely on well-known techniques, such as Internet protocol security (IPsec) or transport layer security (TLS), which can provide both confidentiality and integrity of network traffic. In LLSRC, we use the long-term identity keys provided by the architecture to bootstrap the creation of ephemeral cryptographic keys for use with IPsec. By using this technique, we can secure communications without presharing keys to all nodes, and we do not need each node to be provisioned with the same keys. Each system has an agent that monitors changes to the trusted measurement list and can terminate IPsec tunnels when nodes are removed. This solution can be preconfigured as part of the cloud offering and transparently protects all inter-tenant cloud communications to provide an easy mechanism by which all point-to-point cloud network traffic can be encrypted. However, one limitation of this approach is the need for point-to-multipoint traffic.

<sup>1</sup>A side-channel attack is one that exploits nontextual information, such as timing information or power consumption statistics, generated by an encryption device.

Brokered publish/subscribe services offer a way to broadcast messages to multiple nodes. When nodes first join the service, they specify to a broker a list of topics in which they are interested. When nodes send broadcast messages related to those topics to the broker, the broker sends the messages to all the nodes that have subscribed to that topic. To enhance scalability and elasticity, tenants can leverage a broker operated by the cloud provider to multiplex multiple tenants’ messages across the cloud efficiently [7]. While this practice has many advantages, the broker presents adversaries or malicious cloud insiders with an easy target to attack.

To address the insecurity of brokered cloud messaging systems, we are building a cryptographic overlay that protects data passing over an untrusted broker. The system works by running a proxy on each cloud node that either publishes or receives messages from the cloud broker. These proxies use dynamic group keying to establish and distribute a cryptographic key that tenants can control for each topic [8]. The proxies then encrypt all data that transit the broker and subsequently remove any encryption before delivering the data to the destination application. The trusted membership list provides the keys needed to securely distribute the topic keys and the cues for when to rekey as the membership list changes. This solution is transparent to existing cloud applications (requiring only a configuration change to redirect the application to the proxy rather than to the real broker), and it maintains the elasticity and scalability of the cloud broker.

### Secure Cloud Storage

The ability to store and access data securely is core to developing a protected cloud infrastructure. Threats to data storage security abound in the cloud. Examples include insiders maliciously accessing physical disks, malware modifying critical files, providers improperly sanitizing reused media, and services providing insufficient access control. Again, we turn to cryptography and key management to simplify the trust assumptions we must make to ensure critical data are protected.

The simplest storage media to protect in the cloud is the local storage attached directly to a cloud compute node. While commercial products from HyTrust and SafeNet exist to transparently encrypt these volumes, these products fall short because they rely on software-based or password-based trust with no linkage to

a hardware root of trust or to the integrity state of the system. Using the LLSRC high-assurance architecture, we can mitigate this shortcoming. As with the communication component of cloud computing, long-term identity keys can be used to unlock cryptographic keys that protect local storage, and agents monitoring the tenant's trusted membership list can revoke access to encrypted volumes as the cloud node integrity state changes. By leveraging hardware acceleration and the bootstrapping capabilities of the trusted membership list, this solution can be deployed transparently to applications with minimal performance overhead.

Often, local cloud node storage is not used to store persistent state because nodes may come and go as the storage load varies. When persistent state is not stored in a cloud node, that state must be stored elsewhere and is typically shared with other cloud nodes. Object storage systems (e.g., Amazon Simple Storage Service, OpenStack Swift) or distributed file systems (e.g., Lustre, Ceph) often fill this need. These systems typically rely on access control lists that are enforced by the system. For example, file systems compliant with POSIX (a set of standards known as Portable Operating System Interface for UNIX) will offer owner, group, and other read-and-write permissions.

To move this reference monitor model of access control to one based in cryptography, we need both dynamic group keying and long-term identity keys provided by the architecture. Lincoln Laboratory has developed a prototype system that encrypts data in an object storage system and mediates access to shared resources by employing key management [9]. Using a method similar to the protection of topic keys with a secure publish/subscribe proxy, our system creates a randomly generated key, called the content key, for each object (i.e., piece of data) in the system and encrypts the object with the content key. The system then encrypts the content key using the long-term identity key of each entity that has permission to read the object. These encrypted keys, or key wraps, along with the encrypted object, are stored in the object storage system. The owner of the object can add permissions later and can revoke access by re-encrypting the object under a new key or encrypting under a new key only when new data are written to the object. An asynchronous agent running in the cloud manages when permissions should be updated to reflect changes to the membership list.

The final and most complex storage application to secure is that of databases supporting complex queries. Databases are critical to cloud computing because they allow applications to efficiently access small portions of large datasets. The standard sets of cryptographic algorithms (e.g., block ciphers, cryptographic hash functions, and public key cryptography) are insufficient for use with databases because these algorithms do not allow search operations on ciphertext. We need new cryptographic techniques, such as deterministic, order-preserving, and searchable encryption [10–12]. These cryptographic techniques have proven to be both secure and practical for relational or SQL-style<sup>2</sup> databases. These protections can be deployed with approximately a 10-fold decrease in processing overhead compared to the overhead involved in plaintext operations for a wide variety of advanced queries, including substring matching and ranges [13, 14].

Many cloud applications are moving to a schema and storage pattern that relaxes some of the constraints of SQL to create massively scalable databases. To address this shift, we have developed the Computing on Masked Data (CMD) toolbox, which employs the aforementioned encryption primitives (i.e., low-level algorithms) to provide a high-performance framework for securing data in NoSQL databases like Apache Accumulo [15, 16]. Specifically, to allow users to securely mask data stored in a database, the CMD toolbox makes use of encryption techniques such as semantically secure encryption (often called randomized encryption and shortened to RND, which only supports data retrieval), deterministic encryption (DET, which supports database matching queries), and order-preserving encryption (OPE, which supports database range and match operations).

When data are inserted into the cloud database, they are first converted to sparse representations known as associative arrays (sparse matrices with text-labeled rows, columns, and values). Prior to inserting data into the cloud, users select the appropriate level of masking (i.e., encryption with a secret key) that they will apply to support the security and functionality goals of their application.

To retrieve the masked data, users submit a masked query (with the same key used when they inserted data) or an analytic (such as correlation or thresholding). To view

---

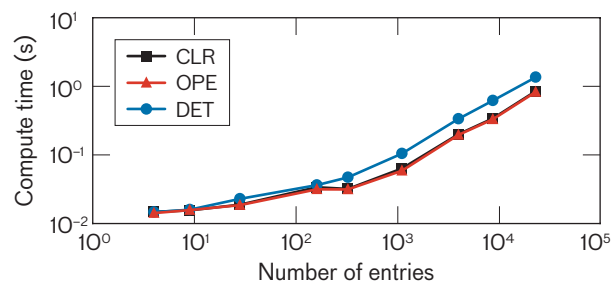
<sup>2</sup>SQL is short for Structured Query Language, a widely used programming language for managing databases.

the results, users unmask their data by using the same key they used to mask the data. The CMD prototype, which has been applied to healthcare data, network logs, and social media datasets, adds no more than twice the relative computational overhead. For example, Figure 5 describes the computational time taken to perform a correlation on masked data (DET and OPE) and plaintext data (CLR, for unencrypted data “in the clear”). We found the time required to perform masked correlation (represented by the red OPE line and blue DET line) is within a factor or two of the time taken to perform the correlation on plaintext data (represented by the black CLR line).

To address the integrity of data stored in cloud databases, we are developing a system to automatically and transparently append digital signatures to all data items in the NoSQL Accumulo database. In addition to developing techniques for protecting data integrity, we are developing a system that uses tools like Merkle hash trees to safeguard the soundness of results returned by the database [17, 18]. This system will ensure that a database cannot suppress or falsify results without detection. Both these systems are implemented in the software the client uses to access the database. If database clients reside in other cloud nodes (that are powering other applications), we can leverage the identity keys from the trusted membership list to authenticate nodes and distribute keys for signing and verifying data in the database.

### Secure Processing

The most challenging problem in cloud security is the protection of data while they are being processed. Because it may often be desirable for users to perform computations, such as statistical analyses, over data stored in the cloud, it is important for cloud providers to ensure that the data are secured throughout the computation. However, most clouds today only allow processing over unencrypted data to give the cloud the access necessary to perform the computation. This approach requires data owners to give up control over their data and trust the cloud to do the right thing. If even a single one of a tenant’s cloud nodes is compromised by an adversary, either through malware or the presence of a malicious insider, then both the confidentiality of the data and the integrity of the computed results may be compromised. Although the trusted computing techniques we have described can detect when the system reaches an

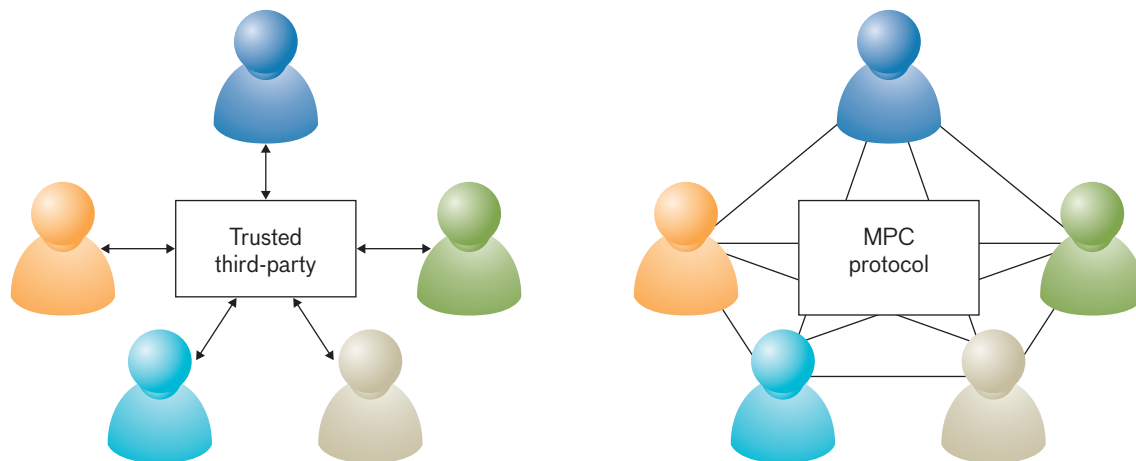


**FIGURE 5.** Computing on Masked Data (CMD) provides support for a number of encryption techniques. In this plot, we show that the time to perform a computation on data masked by using an order-preserving encryption technique (OPE, red line) or a deterministic encryption technique (DET, blue line) is either comparable to or acceptably close to the time to perform a computation on unencrypted data (CLR, black line).

unknown or malicious state, these techniques still leave the data vulnerable as they are being processed and stored in memory on a cloud system.

Consider the proposed semitrusted cloud threat model presented earlier. In this model, we explicitly assumed that some fraction of cloud machines is under adversarial control at any given time. Under this threat model, we cannot rely on trusting the entire cloud, and additional protections are necessary to keep data confidential during processing and to ensure correctness of computation results. A number of cryptographic techniques, such as homomorphic encryption, verifiable computation, and multiparty computation (see [19] for a brief survey), have been proposed by the academic community to protect data during processing in a semitrusted computing environment; however, more research is necessary for developers to understand the applicability and practicality of these techniques in a cloud setting.

To achieve secure processing on a semitrusted cloud, we are investigating the feasibility of secure multiparty computation (MPC) in the cloud. Secure MPC allows multiple parties to work together to compute a joint function on their private data without revealing those data to each other or any external parties and while ensuring that correct results are obtained even if a small number of the parties misbehave. The MPC arrangement is an effective substitute for a scenario in which a perfectly trusted third party would be needed (Figure 6). Distributed computation like MPC arises quite naturally in a cloud setting



**FIGURE 6.** Multiparty computation (MPC) on the right emulates a trusted third-party scenario (on left) to achieve comparable security.

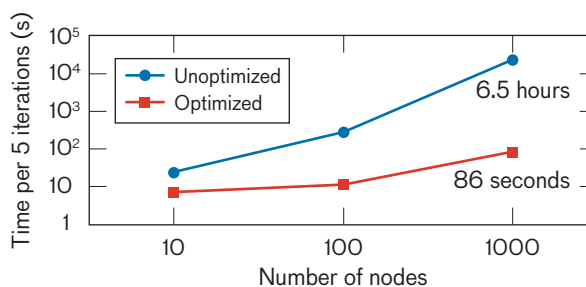
where data may be distributed over multiple cloud nodes or may belong to different cloud tenants. Using MPC, cloud tenants can perform computation over distributed sensitive data while protecting the confidentiality and integrity of the input data and the results, even if some fraction of the machines involved in the computation is corrupted. In fact, this security guarantee does not even require that the identities of corrupted parties be known.

The academic cryptography community originally developed secure MPC in the mid-1980s [20–23]. However, for more than 20 years, the computing community considered MPC protocols purely theoretical novelties and explored few, if any, real-world applications of the protocols. This perception that MPC is impracticable has been shattered in the past few years as multiple efficient protocols have been implemented and used for applications such as secure auctions [24] and private statistical analysis [25]. These demonstrated uses of MPC protocols have confirmed that MPC is nearing readiness for real-world applications; however, much work remains before these protocols can be employed in a cloud setting. Specifically, it is necessary to build protocols that can perform efficiently for computations that are distributed over a large number of cloud nodes and that are optimized for the computations typically performed in cloud settings.

Lincoln Laboratory is currently addressing both of these requirements for cloud MPC. First, we are investigating ways to decrease the number of nodes that must communicate to each other in an MPC protocol.

Reducing this communication locality is critical in a cloud network because the reduction lowers the total communication bandwidth necessary for MPC to run with many cloud nodes. Additionally, we are identifying and building MPC protocols for cloud-specific computations, such as shared cyber situational awareness and graph anomaly detection [26].

To better understand the usability and utility of MPC for such applications, we have developed an initial prototype of MPC for graph anomaly detection. Figure 7 shows the performance of both the unoptimized and optimized versions of our prototype. For the optimized prototype, we



**FIGURE 7.** Running times of multiparty computation (MPC) are shown for graph anomaly detection. The blue line represents an unoptimized initial prototype directly translating the algorithm into its MPC implementation. The red line represents an MPC implementation of the same protocol that uses several MPC-specific optimizations (e.g., fixed-point arithmetic, sparse matrices) to significantly improve performance.

considered both the mathematics of the graph anomaly detection algorithm and the overhead of those mathematical operations in MPC. We designed special-purpose optimizations that minimize the use of expensive MPC operations and increased the use of parallelism. Our results demonstrate that we can achieve improvements of several orders of magnitude by applying MPC-specific optimizations that transform existing algorithms into an appropriate form for secure computation. Efficient MPC for such computations will demonstrate its applicability in a cloud environment and will open the path to adding strong security to cloud processing.

### Achieving the LLSRC Vision

Developing the architecture and components discussed in this article represents a broad and sizable research agenda. Lincoln Laboratory is working on specific research and development (R&D) challenges within this space; however, we do not expect to solve all of them. We are leveraging promising R&D technology coming from academia, commercial companies, and other government labs and agencies. We are combining both our own technologies and those from other sources into an integration test bed.

Using the test bed as a platform for technology demonstration and evaluation, we hope to evangelize the semitrusted cloud security model. By showing that technology *can* achieve stronger security guarantees than those achievable with the current trust models in which trust is full, unverified, and absolute, we may drive the adoption of technology that employs the semitrusted threat model, and we may influence changes in commercial offerings.

### Acknowledgments

We appreciate the support of the following individuals who are researching and developing secure, resilient systems and who have advised us on our work and this paper: Mayank Varia, Jeremy Kepner, Emily Shen, Eric Evans, David Martinez, Albert Reuther, Bryan Richard, Sasha Berkoff, Sophia Yakoubov, Ben Fuller, Andrew Prout, Matt Hubbell, Roop Ganguly, Anna Simpson, Braden Hancock, Mike Calder, Chris Botaish, and Gene Itkis. ■

### References

1. P.M. Mell and T. Grance, "The NIST Definition of Cloud Computing," Technical Report SP 800-145. Gaithersburg, Md.: National Institute of Standards and Technology, 2011.
2. Defense Science Board, Report of the Task Force on Cyber Security and Reliability in a Digital Cloud. Washington, D.C.: Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2013, available at <http://www.acq.osd.mil/dsb/reports/CyberCloud.pdf>.
3. D. Goodin, "AWS Console Breach Leads to Demise of Service with 'Proven' Backup Plan," *ars technica*, June 2014, available at <http://arstechnica.com/security/2014/06/aws-console-breach-leads-to-demise-of-service-with-proven-backup-plan>.
4. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 2009, pp. 169–178.
5. S. Berger, R. Cáceres, K.A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," *Proceedings of the 15th Conference on USENIX Security Symposium*, vol.15, article 21, 2006.
6. J. Schiffman, T. Moyer, C. Shal, T. Jaeger, and P. McDaniel, "Justifying Integrity Using a Virtual Machine Verifier," *Proceedings of the 2009 Computer Security Applications Conference*, 2009, pp. 83–92.
7. Amazon Web Services, Simple Notification Service (SNS), available at <http://aws.amazon.com/sns/>.
8. R. Khazan and D. Utin, "Lincoln Open Cryptographic Key Management Architecture," Tech Note. Lexington, Mass.: MIT Lincoln Laboratory, 2012, available at [http://www.ll.mit.edu/publications/technotes/TechNote\\_LOCKMA.pdf](http://www.ll.mit.edu/publications/technotes/TechNote_LOCKMA.pdf).
9. G. Itkis, V. Chandar, B. Fuller, J. Campbell, and R.K. Cunningham, "Iris Biometric Security Challenges and Possible Solutions," *IEEE Signal Processing Magazine*, vol. 32, no. 5, 2015, pp. 42–53.
10. M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and Efficiently Searchable Encryption," chapter in *Advances in Cryptology—CRYPTO 2007*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 535–552.
11. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-Preserving Symmetric Encryption," *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 224–241.
12. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 79–88.
13. M. Varia, B. Price, N. Hwang, A. Hamlin, J. Herzog, J. Poland, M. Reschly, S. Yakoubov, and R.K. Cunningham, "Automated Assessment of Secure Search Systems," *Operating Systems Review*, vol. 49, no. 1, 2015, pp. 22–30, published by the ACM Special Interest Group on Operating Systems.

14. R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, 2011, pp. 85–100.
15. J. Kepner, V. Gadepally, P. Michaleas, N. Schear, M. Varia, A. Yerukhimovich, and R.K. Cunningham, "Computing on Masked Data: A High Performance Method for Improving Big Data Veracity," *Proceedings of the 2014 High Performance Extreme Computing Conference*, 2014, pp. 1–6.
16. V. Gadepally, B. Hancock, B. Kaiser, J. Kepner, P. Michaleas, M. Varia, and A. Yerukhimovich, "Computing on Masked Data to Improve Big Data Security," *IEEE International Symposium on Technologies for Homeland Security*, 2015, available at <http://arxiv.org/abs/1504.01287>.
17. P.T. Devanbu, M. Gertz, C.U. Martel, and S.G. Stubblebine, "Authentic Third-Party Data Publication," *Fourteenth Annual Working Conference on Database Security: Data and Application Security, Development and Directions*, 2001, pp. 101–112.
18. M. Goodrich and R. Tamassia, "Efficient Authenticated Dictionaries with Skip Lists and Commutative Hashing," Technical Report, Johns Hopkins Information Security Institute. Baltimore: Johns Hopkins, 2001.
19. S. Yakubov, V. Gadepally, N. Schear, E. Shen, and A. Yerukhimovich, "A Survey of Cryptographic Approaches to Securing Big-Data Analytics in the Cloud," *Proceedings of the 2014 High Performance Extreme Computing Conference*, 2014, pp. 1–6.
20. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computation," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 1–10.
21. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty Unconditionally Secure Protocols," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 11–19.
22. O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority," *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 1987, pp. 218–229.
23. A.C. Yao, "How to Generate and Exchange Secrets," *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 162–167.
24. P. Bogetoft, D.L. Christensen, I. Damgård, M. Geisler, T.P. Jakobsen, M. Krojgaard, J.D. Nielsen, J.B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, "Secure Multiparty Computation Goes Live," in *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 325–343.
25. D. Bogdanov, L. Kamm, S. Laur, and V. Sokk, "Rmind: A Tool for Cryptographically Secure Statistical Analysis," International Association for Cryptologic Research, Cryptology ePrint Archive no. 2014-25525, 2014.

26. K.M. Carter, N. Idika, and W.W. Streilein, "Probabilistic Threat Propagation for Network Security," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 9, 2014, pp. 1394–1405.

#### About the Authors



**Nabil A. Schear** is a senior staff member in the Secure Resilient Systems and Technology Group at Lincoln Laboratory, where he leads efforts to create secure cloud computing environments for the Department of Defense and the federal government. His research interests include intrusion tolerance and prevention, network traffic analysis, data tracking and isolation, and anonymous and covert communications systems. Prior to joining the Laboratory in 2011, he was a member of the technical staff at Los Alamos National Laboratory, performing research and development in vulnerability analysis, Internet privacy, high-speed network traffic monitoring, sensor networking, and secure sanitization. He holds a bachelor's degree from the Georgia Institute of Technology, a master's degree from the University of California at San Diego, and a doctorate from the University of Illinois at Urbana-Champaign, all in computer science.



**Patrick T. Cable** is a member of the technical staff in the Secure Resilient Systems and Technology Group. His interests include cloud computing and the implementation of distributed systems in cloud environments. Prior to joining the group in 2014, he architected and managed computing resources for a network used in the creation of novel hardware. He has also worked on various free-space optical programs at Lincoln Laboratory. For the USENIX Large Installation System Administration Conference, he served on the program committee and was the chairperson for invited talks. He holds a bachelor's degree in computer network and information systems from Wentworth Institute of Technology and a master's degree in system and network administration from the Rochester Institute of Technology. His graduate project focused on the safe use of containers on bastion hosts for remote access to sensitive networks.





**Robert K. Cunningham** is the leader of the Secure Resilient Systems and Technology Group. He is responsible for initiating and managing research and development programs in information assurance and computer and platform security. His early research at Lincoln Laboratory focused on machine learning, digital

image processing, and image and video understanding. As part of this effort, he contributed to early drafts of the real-time message passing interface (MPI/RT) specification. Later, as a member of the technical staff in the Information Systems Technology Group, he pursued system security research and development, initially investigating intrusion detection systems that do not require advance knowledge of the method of attack, then moving on to consider detection and analysis of malicious software. He has patented security-related technology, presented and published widely, and served as general chair or program chair for many conferences and workshops. He has also served on several national panels, such as the U.S. Army Cyber Materiel Development Strategy Review Panel, and led national teams, such as the National Security Agency's working group for computer network defense research and technology transition. He holds a bachelor's degree in computer engineering from Brown University, a master's degree in electrical engineering from Boston University, and a doctorate in cognitive and neural systems from Boston University.



**Vijay N. Gadepally** is a member of the technical staff in the Secure Resilient Systems and Technology Group and a researcher at the MIT Computer Science and Artificial Intelligence Laboratory. His research is focused on big data and the Internet of Things systems, machine learning, high-performance computing,

advanced database technologies, and pattern recognition. Prior to joining Lincoln Laboratory in 2013, he worked as a postgraduate intern with the Raytheon Company, a visiting scholar with the Rensselaer Polytechnic Institute, and a student intern with the Indian Institute of Technology. He holds a bachelor's degree in electrical engineering from the Indian Institute of Technology in Kanpur, and master's and doctoral degrees in electrical and computer engineering from The Ohio State University. His doctoral dissertation on signal processing focused on developing mathematical models to accurately estimate and predict driver behavior for autonomous vehicle applications.



**Thomas M. Moyer** is a member of the technical staff in the Secure Resilient Systems and Technology Group. His current research addresses the development of secure infrastructures for cyber systems through the application of trusted computing technologies and secure data provenance. Prior to joining

Lincoln Laboratory in 2011, he worked as a research assistant at the Pennsylvania State University, investigating problems in cyber security, including secure cloud computing infrastructures and storage security. He holds a bachelor's degree in computer engineering and master's and doctoral degrees in computer science and engineering, all from the Pennsylvania State University. His graduate research focused on high-integrity web applications using commodity trusted hardware.



**Arkady B. Yerukhimovich** is a member of the technical staff in the Secure Resilient Systems and Technology Group. His research interests include theoretical cryptography and the application of cryptography to securing real-world systems and applications. His current projects at Lincoln Laboratory involve developing

cryptographic techniques for securing data and processing in large netcentric systems. He holds a bachelor's degree in computer science from Brown University and master's and doctoral degrees in computer science from the University of Maryland. His graduate research focused on understanding the limitations of standard cryptographic techniques and assumptions.

# Securing the U.S. Transportation Command

**Jeffrey M. Diewald, Kajal T. Claypool, Jesslyn D. Alekseyev, George K. Baah,**

**Uri Blumenthal, Alfred Cilcius, William L. Pughe, Joseph A. Cooley, Robert K.**

**Cunningham, Jonathan R. Glennie, Edward F. Griffin, and Patrick J. Pawlak**

The U.S. Transportation Command moves soldiers, equipment, and supplies around the world to support U.S. military and disaster relief operations. To help ensure that this critical supply chain is functioning efficiently, Lincoln Laboratory is working with the command to develop a software architecture that will provide the command with an enterprise network with ample computational power, strong cyber security, and resiliency to attacks and disruptions.



*You will not find it difficult to prove that battles, campaigns, and even wars have been won or lost primarily because of logistics.*

— General Dwight D. Eisenhower

**History is rich with examples that showcase the power** of military logistics and its influence in the outcome of wars. Hannibal crossing the Alps with foot soldiers, horsemen, and elephants to gain a string of victories in central Italy between 218 and 204 BCE relied on logistical planning, cutting supply lines for Roman forces and seizing Roman supply depots [1]. The six years of the Battle of the Atlantic in World War II were a struggle to get a million tons of imported material to Britain every week, fighting German efforts to sink as many of the cargo ships as possible [2]. As the battle raged in the Atlantic Ocean, Allied bombers were destroying German access to oil refineries and synthetic fuel factories. By 1944, the Germans did not have enough fuel for aircraft to protect the oil facilities that remained or for the fleet of submarines that had caused so much damage in the Atlantic [3]. These are just two examples that demonstrate the effectiveness of a military strategy to disable an enemy's supply lines.

The United States' extraordinary and unique ability to rapidly project national power and influence are a direct result of its transportation command—the United States Transportation Command (USTRANSCOM). Uninterrupted and efficient operation of the U.S.

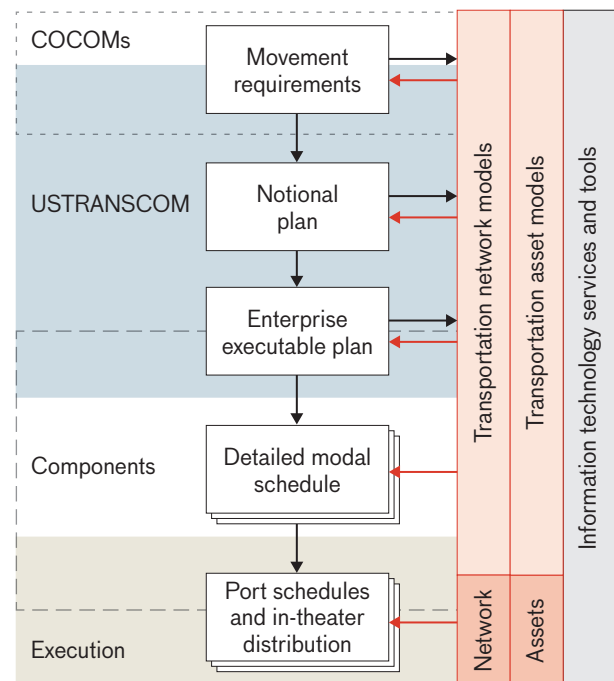
transportation supply chain is critical for ensuring the nation's ability to deploy forces for military actions or humanitarian aid and disaster relief. USTRANSCOM, whose transportation systems have evolved out of many separate distribution systems and programs, is now looking to consolidate and refactor these disparate components and to secure its computing and storage infrastructure. Lincoln Laboratory has been enlisted to help architect this next-generation USTRANSCOM enterprise.

The Laboratory's efforts are addressing fundamental operational and cyber security issues to improve the overall USTRANSCOM defensive posture and cyber visibility across the command. The goal is to facilitate the development of an enterprise that is robust, secure, and resilient to disruptions, whether from cyber attacks, geopolitical turmoil, meteorological events, or natural disasters.

### USTRANSCOM: Background and Enterprise Needs

USTRANSCOM is one of nine unified commands for the Department of Defense, providing air, land, and sea transportation in times of peace and war. USTRANSCOM, established in 1987, serves as the single manager of the U.S. global defense transportation system, supporting troop deployment and sustainment, air refueling, medical evacuations, presidential movements, as well as humanitarian aid and disaster relief missions. In a typical week, USTRANSCOM executes roughly 1900 air missions, 25 ship movements, and 10,000 ground shipments across 75% of the world's countries [4]. All USTRANSCOM missions are conducted worldwide and employ military and commercial transportation assets coordinated through the three USTRANSCOM Transportation Component Commands: the Air Force's Air Mobility Command, which provides aerial refueling capabilities and air transport for people and supplies; the Army's Surface Deployment and Distribution Command, which plans and executes surface deliveries of supplies and equipment; and the Navy's Military Sealift Command, which directs sea transportation [5]. Seventy percent of these movements are subcontracted to commercial partners, such as Maersk, United Parcel Service, and small local shipping companies [6].

As shown in Figure 1, a typical mission is initiated by a request, including a set of movement requirements, from a combatant command (COCOM). A



**FIGURE 1.** The USTRANSCOM operational flow begins with a request for a mission from a combatant command (COCOM) and progresses through the stages of planning and scheduling until the appropriate Transportation Component Command executes the mission, relying on the networks and assets available, including commercial carriers.

movement requirement specifies the type, quantity, source, destination, and timeframe for movements of goods and personnel. A notional plan is produced by USTRANSCOM Fusion Center personnel, in collaboration with the combatant and Transportation Component Commands [7, 8]. This plan is developed through an iterative process that evolves on the basis of the availability of the planes, ships, trains, and trucks needed to move goods and personnel from source to destination. The result of this process is an enterprise executable plan, which can be broken down into explicit instructions, i.e., a modal schedule, for each of the component commands.

This initial planning is sufficient to start the movements, but the world is an uncertain place, given the threat of cyber attack, the politics of military movements, the unpredictability of weather, the inevitability of mechanical failures, or the chaos created by a natural disaster. Thus, replanning is a fundamental process, perhaps the core process, in USTRANSCOM

movements. Ripples flow up and down the hierarchy in Figure 1; sometimes the higher-level planners can adjust their plans according to timely and accurate situational awareness, but often there is no time, and solutions must be found at the lower levels.

Because USTRANSCOM processes and distribution systems evolved separately and independently, coordinating and securing the operations of the various systems is a challenging task. Each command in the USTRANSCOM portion of the process has separate business processes and information systems, each with its own information representations, business rules, and constraints. Although these processes and systems are still functioning to accommodate USTRANSCOM's round-the-clock, all-year-long schedule of missions, they are inefficient and grow less secure as adversaries find and build new exploits to infiltrate computer networks. Differences in information systems necessitate individualized cyber security solutions to meet Department of Defense (DoD) security requirements and, therefore, increase the cost and effort needed to administer these systems. In addition, because the majority of its movements are executed by commercial vendors, USTRANSCOM is compelled to exchange information in schemas defined by those vendors and with information systems outside of the cyber security purview of the DoD. Many of these vendors are based outside the United States, run by foreign nationals who are operating their businesses across the open Internet.

Furthermore, USTRANSCOM's plans, as well as all the information underpinning those plans, are crucial to the United States; therefore, these plans are of great interest to U.S. adversaries. The most advanced of these adversaries are constantly probing for a foothold inside USTRANSCOM as part of their search for more permanent access to U.S. secrets. These adversaries also look to attack and compromise commercial vendors' systems as a pathway into the USTRANSCOM enterprise. The list of vendors targeted by cyber attackers includes cleared defense contractors that build and maintain applications used by USTRANSCOM. Adversaries hope that if they can compromise an application at a contractor's site, USTRANSCOM will not detect the exploitable capability inserted into a system and will then install it as part of a regular upgrade performed to synchronize USTRANSCOM systems with those of the contractor.

Any changes to USTRANSCOM systems, for modernization or enhanced cyber security, cannot delay ongoing missions. Nevertheless, USTRANSCOM recognizes that an incremental modernization and consolidation of their distributed architecture could bring significant improvements to the enterprise:

- Faster response to external events, improving the efficiency of operational plans and timelines
- More flexibility to overcome access challenges, such as bad weather, geopolitical uncertainties, and active anti-access/area denial efforts by adversaries
- Better throughput in wartime or crisis operations
- More efficient operations that would lower fuel expenses, contract costs, and maintenance costs for planes, ships, and other fleet vehicles
- Reduced data storage costs achieved by moving to the cloud or cloud-ready technologies
- Increased cyber security through a reduced, reproducible, consistent, and measurable cyber attack surface
- Improved cyber security for the software development supply chain

Several groups across two divisions at Lincoln Laboratory are collaborating to find architectural solutions that will allow USTRANSCOM to take advantage of these improvements. The Laboratory is providing these answers through prototypes, demonstrations, recommended technologies, and tactics, techniques, and procedures (TTP) that improve cyber security, all enabled by an architecture based on three key tenets:

- *Platform as a Service (PaaS)/Infrastructure as a Service (IaaS) lifecycle security*: Service lifecycle security focuses on defining system and software protections and visibility that should exist in the cloud to isolate malware and adversarial actions while maintaining resilient, visible operational systems. This effort has resulted in recommended technologies and TTPs that segment applications across the entire cloud-based software stack and that help counter the many threat vectors that exist in cloud computing. These recommendations include methods to keep data confidential, to guarantee data cannot be tampered with as they move from cloud to user, and to verify data only goes to authorized users. Lincoln Laboratory's approach has been to develop a high-assurance multitenant cloud environment that can be physically distributed across cloud infrastructures.

- *Data lifecycle security*: Data lifecycle security focuses on maintaining data protections and visibility while USTRANSCOM and third parties store, communicate, and manipulate enterprise data. This effort has led to the development and implementation of a strategy to maintain real-time visibility of the enterprise attack surface from the perspective of data as they flow across boundaries within and outside of USTRANSCOM. In particular, the Laboratory's strategy uses data provenance, which is a record of the history of the evolution of data in a computing system [9], for both understanding the critical enterprise data flows and protecting the data.
- *Authentication and authorization*: Each user and system is authenticated before being granted access to USTRANSCOM resources; this process is in keeping with the reference monitor idea first expressed by Anderson [10], and consistent with the National Institute of Standards and Technology's Identity and Access Management plan [11]. Whereas today USTRANSCOM uses a range of authentication and authorization approaches arising from the organic growth of the organization, Lincoln Laboratory is working to transition USTRANSCOM to an architecture that consistently and methodically provides protection. The Laboratory's approach focuses on the use of data provenance to automatically generate a consistent set of authentication and authorization security policies.

This article presents our development of the Lincoln Secure Environment (LSE), a private cloud hosted at Lincoln Laboratory and offering high-assurance PaaS and IaaS support for multiple tenants. We provide an overview of the threat model and security architecture of the LSE, which is part of the USTRANSCOM test range being implemented at Lincoln Laboratory and which serves as the development and demonstration environment for the Laboratory staff working on the USTRANSCOM project. The LSE also serves as a model for USTRANSCOM's software development environment.

The article also describes our work on Using Provenance To Expedite MAC Policies (UPTempo), a tool that uses collected data provenance for the generation of authentication and authorization security policies, and showcases UPTempo's use of data provenance to identify critical enterprise data flows and to generate mandatory access control (MAC) policies for improved access protection.

## USTRANSCOM Next-Generation Architecture Overview

Figure 2 depicts the Laboratory's proposal for a next-generation USTRANSCOM architecture that provides a mature, secure information technology enterprise. The overall architecture has been developed with security-first principles; cyber security is integrated as a key driver of solutions within each layer.

This IaaS cloud is a high-assurance multitenant architecture that is designed to be vendor-agnostic and can be distributed across multiple physical infrastructures. The architecture consists of several layers, with clean, well-defined interfaces between them.

### PLANNING AND ANALYSIS APPLICATIONS

At the very top layer are USTRANSCOM-specific applications, built with the business logic of the USTRANSCOM enterprise. These applications are the domain of planners, analysts, and cyber situational awareness at USTRANSCOM.<sup>1</sup>

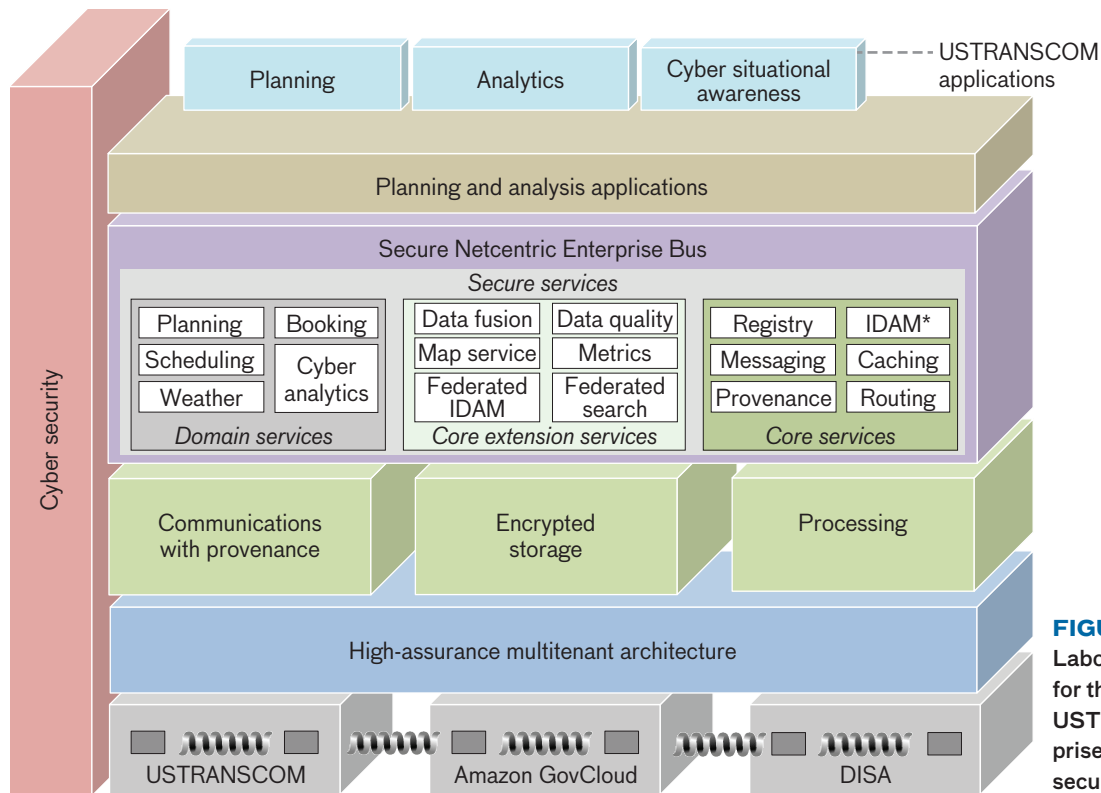
### SECURE NETCENTRIC ENTERPRISE BUS AND ITS SERVICES

This layer is a secure interoperable messaging system that provides a standard service-oriented architecture (SOA) that is federated across the USTRANSCOM enterprise. The message bus provides a simple, consistent interface to a wide range of small, common, sharable, and composable services that the applications can use to build their business logic. This layer of services also provides a federated and unified data-sharing environment for USTRANSCOM and its component commands. By breaking down the "stovepipes," i.e., the rigid implementation barriers that lock existing data in isolated databases, USTRANSCOM should be free to integrate existing data into new and useful combinations. The federated secure SOA provides the foundational global standard from which to support comprehensive interoperability to meet USTRANSCOM and its components' business needs.

This architectural approach has been successfully used at other government organizations. Lincoln

---

<sup>1</sup>Another large part of Lincoln Laboratory's USTRANSCOM project is experimenting with new planning algorithms to add robustness and resiliency for building plans. A third part of this USTRANSCOM project is developing new ways to capture and view cyber situational awareness information so as to detect adversaries earlier, minimizing their damage.



**FIGURE 2.** Lincoln Laboratory’s proposal for the next-generation USTRANSCOM enterprise architecture builds security into every layer.

\*IDAM—Identity and Access Management

Laboratory has used this design pattern at COCOMs for providing improved cyber situational awareness, at the Federal Aviation Administration to provide integrated weather information in support of human-in-the-loop decision support, and in several other Lincoln Laboratory projects [12–16].

**FUNDAMENTAL SERVICES**

All of the services above this layer share three common needs—communications, storage, and computational power—that are served by this layer. First, adding data provenance to the communication services will provide input for UPTempo (detailed in the section titled “Data Provenance in the Lincoln Secure Environment”), leading to stronger cyber security policies and a forensic trail for all communication paths. Second, ensuring that all data at rest are automatically encrypted adds another layer of protection against a determined adversary. Finally, harnessing the power of the cloud gives more processing power than before, allowing applications to experiment with new algorithms and techniques unavailable or impractical in conventional information technology environments.

**HIGH-ASSURANCE MULTITENANT ARCHITECTURE**

This layer provides a common interface to the underlying cloud infrastructures to allow for moving to a *hybrid cloud* model.<sup>2</sup> There are several cloud options currently available, each with a different set of security guarantees. While most of USTRANSCOM data are unclassified, some sensitive information is classified. Consequently, this classified information must stay within the confines of USTRANSCOM’s private cloud. When the data are not sensitive, it should be possible to move that information into a more public cloud,<sup>3</sup> where USTRANSCOM can make use of the available economies of scale of commercial service providers.

<sup>2</sup>A hybrid cloud is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds) [17].

<sup>3</sup>The Amazon GovCloud is one example of a more public cloud that provides additional security assurances beyond those of the completely public Amazon Cloud offerings. GovCloud use is restricted to U.S. government projects.

A number of challenges must be addressed before this type of hybrid cloud infrastructure<sup>4</sup> can become a reality. The article “Secure and Resilient Cloud Computing for the Department of Defense” in this journal highlights the current state of the art in this domain and the ongoing work at Lincoln Laboratory to address some of these challenges [18].

#### IAAS PHYSICAL INFRASTRUCTURES

At the bottom layer of the architecture are the underlying IaaS cloud infrastructures. In Figure 2, three possible infrastructures are shown: the USTRANSCOM private cloud, the Amazon GovCloud restricted public cloud, and the Defense Information Systems Agency (DISA) DoD cloud.

The current instantiation of the LSE is a private cloud and serves as a model for a potential USTRANSCOM private cloud; it also serves as the development environment Lincoln Laboratory is using to build a prototype of this next-generation architecture. This physical infrastructure must also provide a “root of trust” for the cyber security system. This root of trust is a set of functions, defined in specialized hardware, that provides security guarantees about the system. It must be possible to know that the lowest layers of the system have not been corrupted, i.e., that the boot process is free of malware. Guarantees of security at the lowest levels of the system can provide assurances at higher levels of abstraction—assurances that demonstrate that the cyber security system that crosses all layers is working as designed.

#### Lincoln Secure Environment

The LSE is a prototype environment designed to serve as a (1) sandbox that can be employed to test out options for secure software development and (2) an operational high-assurance, multitenant development environment that can be tested and evaluated for usability on the basis of actual development efforts. As a security sandbox, the prototype environment can be used by Lincoln Laboratory staff to investigate techniques that mitigate some of the risks of system intrusion and theft of sensitive materials. As a development environment, this prototype of

a usable, secure system, with concrete requirements, designs, and implemented technology, can be transferred to USTRANSCOM and incorporated in its efforts to build a large-scale Common Computing Environment (CCE).

#### LSE Architecture

The LSE architecture design was driven by the CCE requirements, Lincoln Laboratory’s developer requirements, and the LSE threat model. The LSE was designed to achieve the following goals:

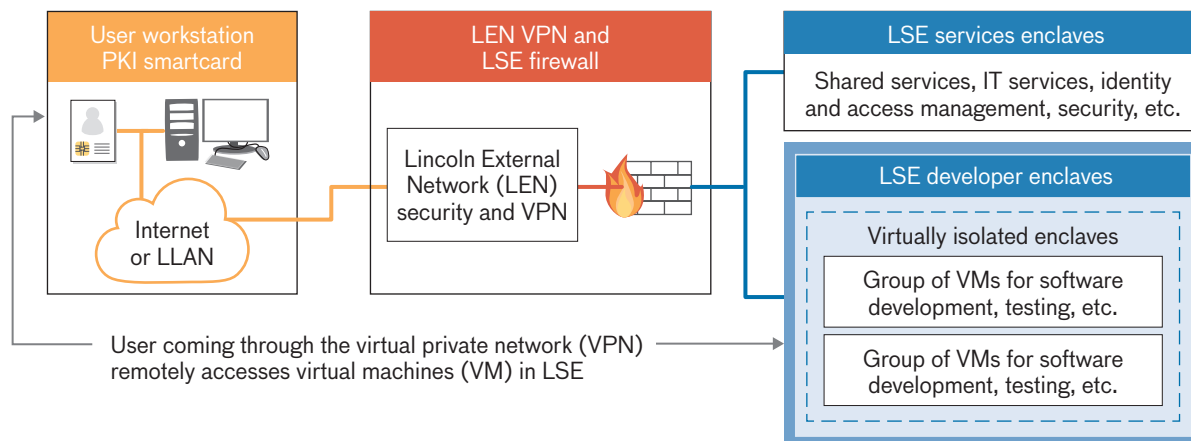
- Provide a secure and usable environment, capable of hosting more than one group of tenants
- Provide each group of tenants with one or more secure development enclaves for their work
- Provide shared, persistent storage within an enclave, with a consistent, roaming profile for each developer
- Secure each developer enclave so that work and data are not visible from other enclaves
- Provide a rich set of shared tools, so that users have the flexibility, within an enclave, to choose their preferred tool chain for source control, build management, release management, and software assurance
- Enforce a secure “single front door” to the LSE and its enclaves
- Enforce two-factor authentication with a hardware-based token for access
- Ensure an automated, configuration-controlled environment, minimizing system administration efforts and guaranteeing a known, reviewed, tested, secured, and malware-free deployment for every element in the LSE
- Allow an evolutionary path for the LSE, so that it can grow from virtualization to IaaS and, eventually, to PaaS

This architecture maps to the lowest three layers in Figure 2. Conceptually, the LSE is a set of isolated enclaves that run on virtual and physical resources. A user sees each enclave as a collection of virtual machines that are accessed from a remote host, through a virtual firewall, as shown in Figure 3.

The LSE is partitioned into two types of enclaves:

1. The developer enclaves are isolated enclaves used to build and test software. Users can spin up virtual machines within their enclave as needed in the development process. The Lincoln Laboratory USTRANSCOM development team “lives” in one of these enclaves. A separate test enclave is used for exercising the latest builds and running the full test suite.

<sup>4</sup>The Lincoln Laboratory team conducted a study to assess the current state of the art for cloud technologies and the gaps at USTRANSCOM. The result of this study is a key recommendation for USTRANSCOM to move toward a hybrid cloud option, deploying to a secure government cloud wherever feasible for unclassified data.



**FIGURE 3.** Each user has one secured path into the Lincoln Secure Environment (LSE), leading to the compartmentalized developer enclaves for each specific project. Developers can collaborate with others on their project, completely isolated from other development projects in the LSE. All developers have access to a common set of services.

2. The services enclaves provide tools and services that are shared by all developers. For example, the software services enclave houses a shared Git multitenant repository, a Nexus server, a Jenkins build server, and other development-oriented services. Common system-wide services, such as Network Time Protocol (NTP), Domain Name Service (DNS), and the like, are provided by the shared infrastructure services enclave.

Virtual firewalls implement additional traffic segmentation between enclaves within the LSE. These enclaves are also isolated from each other via hardware virtualization and network isolation techniques.

In order to promote consistency, repeatability, and configuration management, the LSE leverages automated provisioning tools to virtualize and deliver the infrastructure as a service. This automation, a key contributor to the security of the LSE, provides a scalable, repeatable, auditable method for ensuring secure configurations that are preapproved and enforced within the LSE. The LSE uses configuration-controlled Salt scripts [19] to define the environment programmatically, enabling the orchestration and interconnection of the LSE components. Other Salt scripts ensure that the security controls meet required guidelines, implementing the DoD Security Technical Implementation Guides mandatory for systems at USTRANSCOM. The collection of Salt scripts allows the system administrator to ensure correct, secured components are configured and installed. These scripts also enable system administrators to quickly wipe the

entire system and recreate it in a known-good state after malware has been detected or a cyber attack has occurred. Finally, these scripts define the LSE; upgrade a script and the entire LSE is upgraded.

**Threat Model**

Because USTRANSCOM’s critical planning information, generated plans, applications, and algorithms are valuable to U.S. adversaries, USTRANSCOM faces two important challenges to ensuring its continued ability to perform the U.S. military’s logistics role.

First, the three classes of networks to which USTRANSCOM connects have varying levels of protection. The least well-protected networks are those that are directly connected to the Internet and include the networks of commercial partners that ship nonsensitive materiel. Some commercial companies provide good cyber protection, but not all companies are diligent. The second class of networks includes those used by the military and government to provide USTRANSCOM with requirements about missions.<sup>5</sup> The U.S. government protects these networks from cyber attack by using a blend of commercial and government-developed tools and techniques. The third class of networks consists of those within USTRANSCOM and its Transportation Component

<sup>5</sup> These networks may include nonmilitary government agencies, such as the Centers for Disease Control and Prevention and the Federal Emergency Management Agency, which is involved in U.S. humanitarian aid or disaster relief efforts.



Commands that host the software and data used to plan and execute logistics operations. These networks are the most sensitive and require extra protection.

Second, USTRANSCOM needs to address adversaries who have a wide range of capabilities and who represent the three classes of sophistication (each grouped into a pair of tiers) described in the Report of the Defense Science Board Task Force on Resilient Military Systems and the Advanced Cyber Threat [17]. At the lowest class are the Tier I-II practitioners who rely on others to develop malicious code. These cyber attackers are mainly nuisances, looking to attack public-facing USTRANSCOM websites to demonstrate their capabilities as “hackers.” The DoD uses a variety of techniques, not unique to USTRANSCOM, to limit the access and impact of these nuisances. In the middle class are the Tier III-IV adversaries who can develop their own tools to exploit known vulnerabilities and to discover unknown existing vulnerabilities. These actors appear across all networks, internal and external, usually in search of some financial gain. The final class, Tier V-VI, comprises other great powers, who have sufficient resources to create vulnerabilities in systems and who focus their efforts on USTRANSCOM’s most sensitive networks and data. These Tier V-VI adversaries are constantly seeking entry to USTRANSCOM systems as a way to gain continued access to U.S. classified information. These actors also attempt to compromise external vendor systems as a “back door” into USTRANSCOM’s enterprise. To effectively execute all missions all the time, one would need to protect against the top-tier threats. However, not all data are of equal value. Clearly, data that indicate military plans are of the highest value, and information that discloses the delivery of essential materiel within short time windows, suggesting upcoming military operations, are of great importance.

Broadly speaking, USTRANSCOM needs to ensure that its commercial partners practice good authentication and authorization, perform regular “cyber hygiene” to ensure their systems are patched and up to date, leverage virus detection, and use software developed by a team of people who know how to develop code resilient to cyber attacks. USTRANSCOM’s connections to and from these partners need to be done over secure channels. These precautions address the low-level Tier I-II attackers.

USTRANSCOM needs to do more to address the Tier III-IV attackers. To thwart these adversaries, techniques are needed to securely store data and to verify that data are not modified as they traverse the system. Data must be tracked by using secure provenance techniques [20], and the software should be verified by employing trusted or secure boot techniques [21]. A secure development environment like the LSE provides additional protections [22].

For the serious Tier V-VI adversaries, it must be assumed that, with their skills and persistence, they will succeed in penetrating the USTRANSCOM enterprise. Thus, the aim is to improve the detection and deterrence of attacks, effectively raising the costs for adversaries to reach their goals. This objective is difficult to achieve through technical means because adversaries only need to successfully exploit one vulnerability to gain access to the enterprise while the enterprise’s cyber defenders must protect against all vulnerabilities. By coupling strong authentication and authorization with data provenance, one can better attribute certain attacks to certain actors. Knowing who is responsible can provide insights into what tactics the adversary may use next, leading to additional defenses and a stronger response. This attribution can be shared with other DoD and intelligence community cyber defenders to improve their situational awareness and defensive posture. Improving cyber defensive capabilities for thwarting Tier V-VI adversaries is an ongoing effort by the DoD, as well as in the continuing collaboration between USTRANSCOM and Lincoln Laboratory.

### Security Architecture

The security requirements of the USTRANSCOM CCE form the basis of the LSE requirements. USTRANSCOM CCE requirements include those driven by the DoD. The Joint Information Environment defined by DISA is a part of the DoD’s strategic plan that supplies requirements for the CCE. The Joint Information Environment defines its Single Security Architecture (SSA) with a vision for “a single joint enterprise IT [information technology] platform that can be leveraged for all DoD missions” [23]. Table 1 shows how SSA design principles are satisfied by LSE design artifacts.

The LSE implements a *least privilege* model to control access and determine how to elevate access for users. In this model, users get the smallest set of user rights and

privileges necessary for performing their work. This model is applied on a per-user-per-project basis. Users may have elevated privileges for one project, i.e., in one enclave, but not for another project in another enclave.

**LSE Usability Results**

A key objective of the LSE is usability; if a system is difficult to use because of security measures, users will find a way to work around the difficulties, thereby weakening security. The team developed a survey to determine three things:

- Is the LSE usable for typical development work?
- Does the LSE implementation achieve a balance between security and usability?
- Are there common elements affecting ease of use and productivity in the LSE?

The survey was created by taking demographic questions specific to users in the LSE, together with questions intended to compare the LSE environment with the users' typical work environment, and integrating the industry-standard System Usability Scale (SUS) [24] (i.e., the questions listed in Table 2). The survey took a snapshot of the work being conducted within the LSE, with the goals of (1) rating the process of bringing someone on board and into the LSE, (2) comparing software development in the LSE with Lincoln Laboratory software development outside the LSE, and (3) assessing the ease of conducting daily development tasks within the LSE. The SUS was incorporated to evaluate perceived user satisfaction with the LSE and provide a score that can be compared against a large number of industry examples. The pattern of questions in the SUS, with a positive question followed by a

**Table 1. Joint Information Environment Single Security Architecture (SSA) Principles and Lincoln Secure Environment (LSE) Design Artifacts**

SSA PRINCIPLE	LSE DESIGN ARTIFACTS
Resiliency	<ul style="list-style-type: none"> <li>• All LSE nodes are redundant to allow services to migrate from one virtual machine to another and from one hardware host to a different one.</li> <li>• All the networks are separated from each other by using physical, logical, and cryptographic separation.</li> <li>• All the communications protocols are standard. Specific authentication mechanisms are VMware-specific because currently there is no interoperability between the vendors that support Common Access Card (CAC)-based authentication. DoD-issued CACs provide a required second factor for authentication.</li> </ul>
Maneuverability	<ul style="list-style-type: none"> <li>• LSE administrators can control and configure authentication mechanisms to address emerging needs and security conditions.</li> <li>• Access is granted to data and services, not servers.</li> <li>• Everything inside the LSE is virtualized as much as practical.</li> </ul>
Accessibility	<ul style="list-style-type: none"> <li>• Every user and every device will be authenticated.</li> <li>• Configuration is policy based and is enforced.</li> <li>• All transactions are authorized through access control.</li> </ul>
Visibility	<ul style="list-style-type: none"> <li>• All the network and service-providing hosts are monitored continuously.</li> <li>• All the alerts and other artifacts of the monitoring are fed to a separate network operations center for analysis and for incorporation into a shared situational awareness picture.</li> <li>• All the enclaves and all the nodes within each enclave will conform to the relevant network-related and host-related DoD Security Technical Implementation Guides.</li> </ul>

negative question, was deliberately designed to reduce response bias. Respondents indicated their scores for each question on a five-point Likert scale that ranges from a low score of “strongly disagree” to a high score of “strongly agree” [25]. The survey was conducted twice: once to evaluate the initial version of the LSE and later to assess the LSE after improvements had been made.

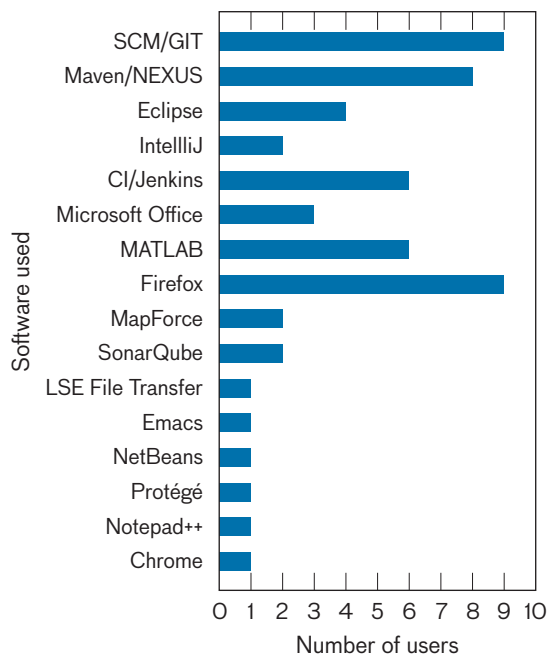
Eleven LSE users (nine developers and two analysts) responded to the initial survey. Fourteen LSE users (thirteen developers and one analyst) responded to the second survey; a majority of these respondents had used both instantiations of the LSE. As of the second survey, nine of the thirteen developers were using the environment for a majority of their development tasks. For six of the thirteen developers, the LSE was their only development environment, with 100% of their work conducted

within that environment. Figure 4 provides a census of some of the software applications the developers and analysts used in their work within the LSE.

Overall, users found that the LSE supported their tasks well (average score of 3.21 on a 5-point Likert scale) and considered the system more secure when compared to their desktop system (average score of 3.85 on the 5-point scale). Additionally, ten of the returning users reported that the system had been improved between surveys. The overall SUS score for the latest release of the LSE was 52 (compared to 44 for the first instantiation of the LSE). Figure 5 shows a comparison of the SUS scores by user between the first and the current versions of the LSE. The SUS score is generated by summing and scaling calculations that equalize the impact of each question. This nonlinear score can be normalized and compared with thousands of other SUS results. The overall mean SUS score for a wide range of open software that users consider usable is 68 [26]. It should be noted that usability scores for closed systems such as the LSE will likely never match the scores for an open system because the requisite security measures for closed systems add complexity

**Table 2. System Usability Scale (SUS) Questions**

NUMBER	QUESTION
Q1	I think that I would like to use this system frequently.
Q2	I found the system unnecessarily complex.
Q3	I thought the system was easy to use.
Q4	I think that I would need the support of a technical person to be able to use this system.
Q5	I found the various functions in this system were well integrated.
Q6	I thought there was too much inconsistency in this system.
Q7	I would imagine that most people would learn to use this system very quickly.
Q8	I found the system very cumbersome to use.
Q9	I felt very confident using the system.
Q10	I needed to learn a lot of things before I could get going with this system.



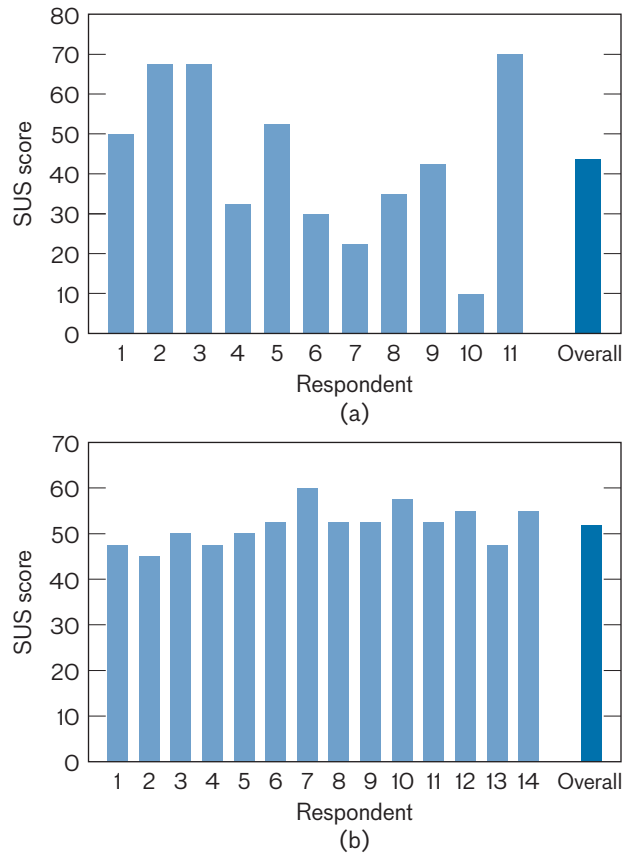
**FIGURE 4.** The software applications used in the Lincoln Secure Environment (LSE) by the developers and analysts who took the survey on the environment’s usability show the range of programs the LSE could support.

and procedures that make a system less convenient and intuitive for operators to use. However, aiming for a high usability score assures that usability is a key consideration in the ongoing development of the LSE.

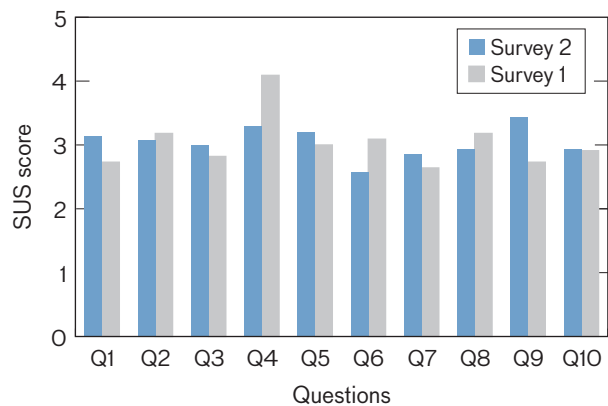
Recent system modifications have resulted in most of the users (>75%) reporting that these improvements have increased usability significantly, and gains were made to lessen the perceived need for the support of a technical person (Figure 6, Q4) and to increase reported overall confidence in system use (Figure 6, Q9). Additionally, SUS scores among respondents to the second survey are more consistent, signaling that improvements may have addressed the most impactful usability issues and that new users do not feel as though they are at a significant handicap. However, the SUS scores indicate that there is still a need for further improvement. Periodic assessment will ensure that LSE development has a continuing focus on the needs of its users, and utilizing the standard SUS scoring mechanism will ensure a fair comparison with future surveys of LSE usability.

**Data Provenance in the Lincoln Secure Environment**

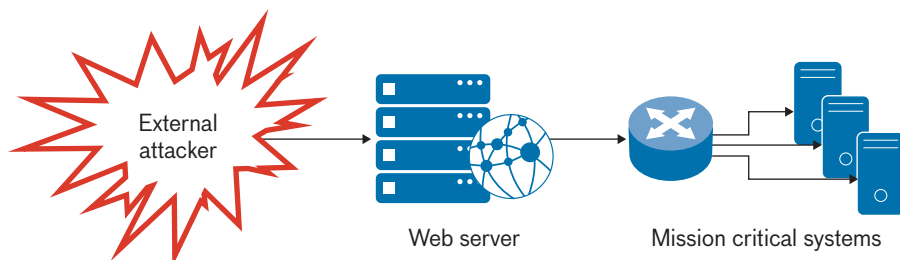
One technique we leverage to address lower-tier and middle-tier adversaries uses data provenance, the recorded evolution of data in a system [9], to produce a set of mandatory access control (MAC) rules. The challenge in using data provenance is instrumenting systems and applications to generate and capture provenance information. USTRANSCOM has, over the years, built, grown, evolved, and acquired a rich, powerful set of legacy applications, systems, and information technology infrastructure to support its business processes. These legacy applications access, create, use, manipulate, and store transportation-related data necessary for analyzing, planning, scheduling, assembling, and executing a mission. Attempting to instrument USTRANSCOM applications to generate application context-based provenance would provide the best information, but would be expensive in time and effort. Kernel-level provenance, in which information is tracked by actions in the operating system, such as reads and writes to files and sockets, quickly and easily produces coverage of provenance events, but at a low level, without much application context.



**FIGURE 5.** System Usability Scale (SUS) scores by users in the Lincoln Secure Environment usability surveys; the initial survey results are in (a) and post-improvements results are in (b). A score of 68 is the average usability score reported in the many usability studies conducted by industry and research institutes.



**FIGURE 6.** System Usability Scale (SUS) scores by question in the Lincoln Secure Environment usability survey. Significant improvements are shown in Q4 (perceived need for technical assistance) and Q9 (perceived confidence of use). Note that even-numbered questions are phrased as negative statements; therefore, a lower score indicates a better usability rating.



**FIGURE 7.** The adversary on the left accesses a targeted web server via a vulnerability in its code and then is able to gain admittance to other mission-critical systems from which the adversary can exfiltrate data.

Typically, when an adversary takes over an application through a security vulnerability, the adversary has all the rights associated with that application. For example, if the application is used to access a database whose contents should be kept confidential, then the adversary has access to the contents of that database and can exfiltrate or modify the data. Figure 7 shows an example of how an adversary can gain access into an organization's network. The cyber attacker exploits a vulnerability in the web server by introducing malicious software and consequently obtains all the rights and privileges of a system administrator assigned to the web server. The attacker can now use these privileges to access other mission-critical systems.

Efficiently collecting provenance data at the kernel level [9, 27] enables us to get a complete picture of the behaviors of subjects (e.g., users, applications, processes that request access) and objects (e.g., files, databases, computers that are accessed) in a system. In a system that collects provenance, the adversary in Figure 7 leaves a trail of evidence for detection and forensic analysis—but that trail is identified too late, and the adversary has gained access. The provenance information generated by an uncompromised system can be used to build SELinux (Security-Enhanced Linux, a version of the Linux operating system that supports access control policies) MAC policies that block the adversary from compromising the web server in the first place.

Table 3 shows a fragment of an SELinux MAC policy for the Firefox web browser. This fragment is designed to prevent an initial browser compromise, which is the first step in compromising the web server. The left and right columns show line numbers and SELinux rules, respectively. The goals of the fragment are to define a limited `e4684_firefox_t` domain and to ensure that users move from the unconstrained `unconfined_t` domain into the restricted `e4684_firefox_t` domain.

Manually writing such a MAC policy is difficult: many of the rules for such a policy can span multiple pages; missing a single rule can cause the application to function incorrectly; writing the policy requires the policy writer to have in-depth knowledge of system and application behavior; and often the interaction of rules within the policy are unknown. Manually developing MAC policies often results in policies that are either too restrictive or overly permissive.

The difficulties in manually developing MAC policies have driven research into several approaches that partially or fully automate the process [28, 29]. Some of these approaches record the interactions of an application with the operating system, gathering and analyzing data from that interaction to build policies. The drawback of these approaches is that gaps might exist in the data. Gaps in the data make it impossible to generate a complete set of rules, resulting in MAC policies that are excessively restrictive or too permissive. To use data provenance to build MAC policies, the provenance data for an application in a system must be complete.

Using Provenance To Expedite MAC Policies (UPTEMPO) is a Lincoln Laboratory framework that utilizes kernel-level data provenance to expedite the generation of MAC policies, thereby automating the securing of computing systems. The MAC policies that UPTEMPO builds ensure that the integrity of data is preserved and limits the damage adversaries can do when they are able to compromise an application. UPTEMPO automatically generates policies conforming to the Biba integrity model [30] by using provenance data on subjects and objects in a computing system. Figure 8 shows the five stages in UPTEMPO: (1) provenance collection, (2) policy generation, (3) policy refinement, (4) policy translation, and (5) policy enforcement.

In the first stage, UPTEMPO collects provenance data on the subjects and objects in a system and stores the data in a provenance data store. In the second stage,

UPTEMPO analyzes the provenance data and uses the results of the analysis to generate a graph representation of the final MAC policy. In the third stage, UPTEMPO refines the graph to remove redundant edges and nodes. In the fourth stage, UPTEMPO translates the refined graph into a MAC policy. Finally, in the fifth stage, UPTEMPO enforces the MAC policy. UPTEMPO addresses the problem of overly restrictive or overly permissive policies by routinely iterating through the five stages.

A production computing environment that uses UPTEMPO to generate policies protects the web server and the mission-critical systems accessed by the server. UPTEMPO policies raise the cost of an attack by slowing the adversary at every step. If the adversary manages to find an effective compromise for the web server that works around UPTEMPO’s policies, additional policies protect the mission-critical systems.<sup>6</sup> Applications are only allowed to access the data they need to function correctly.

UPTEMPO collects provenance information as the system is used. On a regular basis, a system administrator would use UPTEMPO to incrementally generate an updated set of MAC policies. Initially, the human in

the loop would be responsible for generating the policies. In the longer term, this function could be an automated, regular occurrence, removing the human except as someone to sanity check the results. This recurrent policy updating improves security by denying adversaries the time to construct workarounds to MAC policies.

**UPTEMPO Evaluation**

A common attack scenario relies on a compromised web server and a vulnerable program that visits that web server. When a vulnerable program visits the compromised web server, the program is then compromised. Through this compromised program, the adversary subsequently gains access to mission-critical systems in order to exfiltrate data stored there.

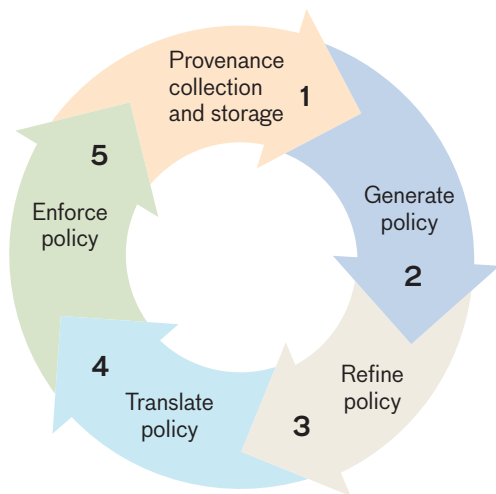
To demonstrate the feasibility of the UPTEMPO approach, we considered the Firefox web browser as a proxy for a mission-critical application and utilized user files as the exfiltrated data. The goal of this initial evaluation was to demonstrate the use of UPTEMPO to restrict a compromised Firefox browser’s functionality to web browsing only, thus prohibiting an adversary from accessing user files.

We followed a four-step process for this evaluation:  
**Step 1:** Collect provenance data from an uncompromised Firefox browser.

<sup>6</sup>This technique, known as “defense in depth,” uses defenses (or walls) that are not just around the perimeter of the system. Because we expect top-tier adversaries to get inside those walls, we build more defenses inside.

**Table 3. An Example Fragment of an SELinux Policy for the Firefox Web Browser**

LINE	SELINUX RULE
1	allow e4684_firefox_t NetworkManager_t:dbus send_msg;
2	allow e4684_firefox_t bin_t:dir {read search open getattr};
3	allow e4684_firefox_t bin_t:file {read execute open getattr};
4	allow e4684_firefox_t bin_t:lnk_file {read getattr};
5	allow e4684_firefox_t bin_t:unix_stream_socket connectto;
6	allow e4684_firefox_t config_home_t:dir {write remove_name search add_name};
7	allow e4684_firefox_t config_home_t:file {rename write getattr read create unlink open};
8	allow e4684_firefox_t d2799_je23930_t:dir {read search open getattr};
9	allow e4684_firefox_t dns_port_t:udp_socket name_connect;
10	type_transition unconfined_t e4684_firefox_exec_t:process e4684_firefox_t;



**FIGURE 8.** The UPTEMPO process is a continual cycle of policy generation and enforcement.

**Step 2:** Use UPTEMPO and the provenance data collected in Step 1 to create an SELinux policy for limiting the use of Firefox.

**Step 3:** Disable SELinux on the system and start Firefox. We verified that Firefox was able to browse the web and read user files.

**Step 4:** Enable SELinux on the system. SELinux, by default, denies all accesses to objects by subjects on a system. Without the UPTEMPO policies in place, Firefox would not be able to browse the web or read user files. We verified that the policies constructed by UPTEMPO allowed Firefox to browse the web but prevented any attempt to read user files.

UPTEMPO generated 246 types, classes, and rules for the policy constraining Firefox. The case study evaluation outlined above exercised 173 of those 246 types, classes, and rules. This initial evaluation shows UPTEMPO has great promise as a tool to further secure an environment. The next steps for this work are (1) more rigorous evaluation of the algorithms and techniques utilized to generate the policy; (2) the development of automated testing procedures that provide extensive coverage of policy cases and, when augmented by clever human-designed tests, result in a thorough assessment of the validity of these complex policies; and (3) automated provisioning of SELinux VMs in the LSE. The LSE does not currently support SELinux, which limited the integration of UPTEMPO into the LSE.

## Future Directions

Enhancing the effectiveness and cyber security of USTRANSCOM's enterprise is a significant ongoing effort that requires interaction among a diverse set of organizations at various levels. Collaboration between Lincoln Laboratory and USTRANSCOM, primarily at the research and development level, is a key driver for advancing the cyber security of the more than 70 USTRANSCOM Programs of Record. LSE, for example, has been transitioned to USTRANSCOM and is being used as a template for the development of its Common Computing Environment (CCE).

Research and development efforts must be continued to ensure improved cyber security of the USTRANSCOM enterprise as it moves from a largely private, stove-piped infrastructure to a unified, cloud infrastructure. This transition should employ advanced technology, such as moving target techniques, sophisticated key management, and heightened provenance collection and mining, at all levels of the application stack.

The continuing development and use of the LSE is providing insights into a secure, usable development environment; these activities also offer a valuable foundation for future work. In 2012, the DoD issued a directive to transfer computing into the cloud where feasible [31]. Moving the LSE into the cloud would help USTRANSCOM understand the implications of moving their CCE to the cloud. The automated provisioning of the LSE should make a transfer to the cloud relatively simple. The cloud can deliver additional capabilities because of its scalability, and the LSE would no longer be limited by its current hardware. Developers using the LSE in the cloud would furnish key SUS usability metrics that could be compared with the SUS scores already gathered. Securing the LSE in the cloud could leverage other cloud cyber security work being done at Lincoln Laboratory.

Building the LSE with an integrated UPTEMPO will generate an environment with improved security, but much more provenance work remains to be done to assure this improvement. An enhanced UPTEMPO could also be used as the first pass at understanding the security properties of the planning and analysis tools being developed at Lincoln Laboratory as part of the USTRANSCOM project.

These tools, which are designed to test new planning algorithms for robustness and for plan resiliency, read from databases and generate new plans. All of these sensitive

data and plans are sought by adversaries. A clever adversary in the environment might act subtly; for example, tampering with data to add a few delays in the transport of essential men and materiel could be far more effective in hampering a U.S. mission than crashing an entire system might be. Protecting data means keeping them confidential and guaranteeing their integrity. Therefore, understanding effective, efficient ways to cryptographically protect those data at rest, in motion, and in use is an issue Lincoln Laboratory is looking at on other projects. A practical USTRANSCOM set of applications and data provides researchers with valuable real-world examples and requirements that can be shared with other Laboratory projects.

In addition, determining the application-level provenance of the data in these tools can yield new cyber security insights. UPTEMPO provides a wealth of low-level provenance information. Adding application-level provenance to the tools can supply much better context to the use of data by the tools. Collecting the application provenance and merging it with the kernel-level provenance from UPTEMPO to explore provenance's potential implications to cyber security is another exciting area of research.

The partnership between USTRANSCOM and Lincoln Laboratory has produced practical prototypes and transferrable technology that will have value to programs beyond USTRANSCOM's. The collaboration is generating additional important questions, and, with both the LSE and UPTEMPO, a foundation is already available for investigating those interesting new questions. ■

## References

1. Wikipedia, "Hannibal's Crossing of the Alps," available at [https://en.wikipedia.org/wiki/Hannibal%27s\\_crossing\\_of\\_the\\_Alps](https://en.wikipedia.org/wiki/Hannibal%27s_crossing_of_the_Alps).
2. Wikipedia, "Battle of the Atlantic," available at [https://en.wikipedia.org/wiki/Battle\\_of\\_the\\_Atlantic](https://en.wikipedia.org/wiki/Battle_of_the_Atlantic).
3. "Why Germany Really Lost World War II," *The European Union Times*, posted 22 Jan. 2010, available at <http://www.eutimes.net/2010/01/why-germany-really-lost-world-war-ii/>.
4. United States Transportation Command website, "About USTRANSCOM," available at <http://www.ustranscom.mil/cmd/aboutustc.cfm>.
5. United States Transportation Command, Publication P35-1 "United States Transportation Command," available at <http://www.ustranscom.mil/cmd/fpindex.cfm>.
6. T. Rorabaugh, "Evaluating the Use of Cloud Hosting for Consolidating USTC IT Infrastructure," The MITRE Corporation, 2014.
7. USTRANSCOM, "AT21 [Agile Transportation for the 21st Century] Time-Phased Optimization Capabilities Storyboard White Paper Air Centric (Draft)," TCAC-F/TCAC-SL, 13 Sept. 2011.
8. USTRANSCOM, "AT21 Time-Phased Optimization Capabilities Storyboard White Paper Deployment Sealift Centric (Draft)," TCAC-F/SDTE-ST, 5 Dec. 2011.
9. D.J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-Fidelity Whole-System Provenance," *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 259–268.
10. J.P. Anderson, "Computer Security Technology Planning Study," Volume II of the Report of the Computer Security Technology Planning Study Panel, 1972.
11. Defense Information Systems Agency, Identity and Access Management (IDAM) webpage, accessed Dec. 2015, available at <http://www.disa.mil/Initiatives/Identity-Access-Mgmt>.
12. D. Staheli, V.F. Mancuso, M.J. Leahy, and M.M. Kalke, "Cloudbreak: Answering the Challenges of Cyber Command and Control," *Lincoln Laboratory Journal*, vol. 22, 2016, pp. 60–78.
13. W. Moser, "Design and Development of the TFDI Information Management Architecture," *Proceedings of the Integrated Communications, Navigation and Surveillance Conference*, 2009, pp. 1–12.
14. V. Mehta, S. Campbell, J. Kuchar, W. Moser, H. Reynolds, T. Reynolds, and R. Seater, "The Tower Flight Data Manager Prototype System," *Proceedings of the 30th IEEE/ALAA Digital Avionics Systems Conference*, 2011, pp. 2C5-1–2C5-15.
15. K. Claypool, "Common Support Services Information Management," MIT Lincoln Laboratory Air Traffic Control Workshop, Washington, D.C., Dec. 2012, available at [http://www.ll.mit.edu/mission/aviation/publications/publication-files/ms-papers/Claypool\\_2012\\_LLATC\\_MS-72923\\_WW-26438.pdf](http://www.ll.mit.edu/mission/aviation/publications/publication-files/ms-papers/Claypool_2012_LLATC_MS-72923_WW-26438.pdf).



16. C. Kelly, "CIWS Data Distribution Service," MIT Lincoln Laboratory Air Traffic Control Workshop, Washington, D.C., Dec. 2012, available at [http://www.ll.mit.edu/mission/aviation/publications/publication-files/ms-papers/Kelly\\_2012\\_LLATC\\_MS-72864\\_WW-26438.pdf](http://www.ll.mit.edu/mission/aviation/publications/publication-files/ms-papers/Kelly_2012_LLATC_MS-72864_WW-26438.pdf).
17. Department of Defense, Report of the Defense Science Board Task Force on Resilient Military Systems and the Advanced Cyber Threat, J.R. Gosler and L. Von Thaeer, task force cochairs, Jan. 2013.
18. N.A. Schear, P.T. Cable, R.K. Cunningham, V.N. Gadepally, T.M. Moyer, and A.B. Yerukhimovich, "Secure and Resilient Cloud Computing for the Department of Defense," *Lincoln Laboratory Journal*, vol. 22, 2016, pp. 123–135.
19. SaltStack, SaltStack Automation for CloudOps, ITOps and DevOps at Scale, available at <http://saltstack.com>.
20. A. Bates, D. Tian, K.R.B. Butler, and T. Moyer, "Trustworthy Whole-System Provenance for the Linux Kernel," *Proceedings of the 24th USENIX Security Symposium*, 2015, pp. 319–334.
21. R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and Implementation of a TCG-Based Integrity Measurement Architecture," *Proceedings of the 13th Conference on USENIX Security Symposium*, vol. 13, 2004, p. 16.
22. Department of Defense, Report of the Defense Science Board Task Force on Cyber-Security and Reliability in a Digital Cloud, E.D. Evans and R. L. Grossman, task force cochairs, Sept. 2012.
23. Defense Information Systems Agency, "Shaping the Enterprise for the Conflicts of Tomorrow," 2014, available at [http://www.disa.mil/~media/Files/DISA/About/JIE101\\_000.pdf](http://www.disa.mil/~media/Files/DISA/About/JIE101_000.pdf).
24. J. Brooke, "SUS: A Quick and Dirty Usability Scale," Chapter 21 in *Usability Evaluation in Industry*, P.W. Jordan, B. Thomas, B.A. Weerdmeester, and I.L. McLelland, eds. Bristol, Penn.: Taylor & Francis, 1996.
25. R. Likert, "A Technique for the Measurement of Attitudes," *Archives of Psychology*, vol. 22, no. 140, 1932, pp. 1–55.
26. J. Sauro, "Measuring Usability with the System Usability Scale," *Measuring Usability* website, 2 Feb. 2011, available at <http://www.measuringu.com/sus.php>.
27. A. Bates, K.R.B. Butler, and T. Moyer, "Linux Provenance Modules: Secure Provenance Collection for the Linux Kernel," Technical Report CIS-TR-201407-01, University of Oregon, 2014.
28. N. Provos, "Improving Host Security with System Call Policies," *Proceedings of the 12th Conference on USENIX Security Symposium*, vol. 12, 2003, p. 18.
29. B.T. Sniffen, D.R. Harris, and J.D. Ramsdell, "Guided Policy Generation for Application Authors," MITRE Technical Papers, The MITRE Corporation, 2006.
30. K.J. Biba, "Integrity Considerations for Secure Computer Systems," The MITRE Corporation, Technical Report 76-372, 1977.
31. "Cloud Computing Strategy," Report issued by the Chief Information Officer, U.S. Department of Defense, 5 July 2012.

## About the Authors



**Jeffrey M. Diewald** is a member of the technical staff in the Secure Resilient Systems and Technology Group at Lincoln Laboratory. Since joining the Laboratory in 2011, he has been working on the development of secure, usable software for building and managing certificate authorities, provenance, data integrity, and dynamic key management. His recent efforts are in helping to add provenance technologies to the USTRANSCOM project code base, to bring usable security to the Lincoln Secure Environment, and to understand how cloud technologies can help USTRANSCOM. Previously, he worked for several notable companies, including Digital Equipment Corporation, Compuware, and Avid Technology. During that time, he developed and delivered compilers, debuggers (resulting in two U.S. patents), software performance tools, and user interfaces for several significant products. He holds bachelor's degrees in computer engineering and electrical engineering, and a master's degree in electrical engineering, all from the University of Michigan.



**Kajal T. Claypool** is currently the assistant leader for the Informatics and Decision Support Group and the program lead for Lincoln Laboratory's USTRANSCOM project. She has also worked on several of Federal Aviation Administration's Next-Generation Air Transportation System programs during her tenure at Lincoln Laboratory and is actively engaged in research and technology development in big data management. She has been active in the database research community for more than 15 years, and her interests include information integration, information retrieval, data analytics, and secure architectures. She has authored numerous publications in these and related areas. Prior to joining the Laboratory in 2007, she worked as a member of technical staff at Oracle Corporation and served as an assistant professor in the Department of Computer Science at the University of Massachusetts, Lowell. She holds a bachelor of technology degree in computer engineering from the Manipal Institute of Technology, Karnataka, India, and a doctoral degree in computer science from Worcester Polytechnic Institute.



**Jesslyn D. Alekseyev** is a member of the technical staff in the Informatics and Decision Support Group. She is currently a human-factors researcher and user-experience designer on the USTRANSCOM and Biosurveillance Ecosystem projects. Her work at Lincoln Laboratory has focused on interface design, information architecture, and user research and analysis. Prior to joining the Laboratory, she worked as a designer, with an interest in process and resource management for exhibit, environmental, and wayfinding projects

and with additional experience in transportation logistics through her work with the United Parcel Service. She has a bachelor's degree in illustration from Syracuse University, with additional coursework in communication design, interaction design, and psychology. She is currently working toward a master's degree in human factors in information design at Bentley University.



**George K. Baah** is a technical staff member in the Cyber Analytics and Decision Systems Group. Prior to joining Lincoln Laboratory in 2013, he was a post-doctoral fellow at the Georgia Institute of Technology. His research interests include cyber security, program analysis, machine learning, and causal analysis. He holds a

bachelor's degree (magna cum laude) in computer science, with a minor in mathematics, from Pace University and a doctoral degree in computer science from the Georgia Institute of Technology.



**Uri Blumenthal** is a member of the technical staff in the Secure Resilient Systems and Technology Group. Since joining Lincoln Laboratory in 2007, he has worked on secure communication protocols, computer applications, and infrastructures. His professional activities have focused on information assurance, cyber assessments,

identity management and access control in complex configurations, and distributed security architecture. Other areas of interest include cryptographic algorithms and protocols, network management, and theory of music. Prior to joining the Laboratory, he worked at several research facilities, including IBM Research, Bell Labs, and Intel Research Labs. He has published papers and books on cryptography, public key infrastructure, network management, and cyber situational awareness. He holds a master's degree in applied mathematics from Odessa State University and a master's degree in divinity from The Interfaith Seminary of New York.



**Alfred Cilcius** is a member of the Informatics and Decision Support Group. Since joining Lincoln Laboratory in 2013, he has been supporting the USTRANSCOM project by working on the Lincoln Secure Environment, developing DevOps (methodology derived from a collaboration between operations staff and

development engineers), automated provisioning, secure network and enclave isolation, and open-source software. Before joining the Laboratory, he was the chief software architect at AgilePath and worked for other notable companies, including TYBRIN Corporation (now a segment of Jacobs Technology), The MITRE Corporation, Wang Laboratories, and various startups. He holds a bachelor's degree in physics from Bates College and a master's degree in systems and advanced technology from the State University of New York, Binghamton.



**Joseph A. Cooley** is a member of the technical staff in the Informatics and Decision Support Group. Since joining Lincoln Laboratory in 2000, he has worked on a wide range of computer system problems, including durable archival storage, distributed systems, Internet protocol networking in disadvan-

taged environments, applied cryptography, enterprise security, and energy systems. As a Lincoln Scholar from 2009 to 2011, he completed a master's degree in computer science at Dartmouth College under the direction of Dr. Sean W. Smith. His graduate research focused on improving usable security in the domain of medical informatics. He also holds a bachelor's degree in computer science from the University of Minnesota.



**Robert K. Cunningham** is the leader of the Secure Resilient Systems and Technology Group. He is responsible for initiating and managing research and development programs in information assurance and computer and platform security. His early research at Lincoln Laboratory focused on machine learning, digital

image processing, and image and video understanding. As part of this effort, he contributed to early drafts of the real-time message passing interface (MPI/RT) specification. Later, as a member of the technical staff in the Information Systems Technology Group, he pursued system security research and development, initially investigating intrusion-detection systems that do not require advance knowledge of the method of attack, then moving on to consider detection and analysis of malicious software. He has patented security-related technology, presented and published widely, and served as general chair or program chair for many conferences and workshops. He has also served on several national panels, such as the U.S. Army Cyber Materiel Development Strategy Review Panel, and led national teams, such as the National Security Agency's working group for computer network defense research and technology transition. He holds a bachelor's degree in computer engineering from Brown University, a master's degree in electrical engineering from Boston University, and a doctorate in cognitive and neural systems from Boston University.



**Jonathan R. Glennie** is an IT systems administrator for the Air Traffic Control Systems and the Informatics and Decision Support Groups. He began working at Lincoln Laboratory in 2006 after earning a bachelor's degree in computer networking and systems administration from Michigan Technological University. Prior to

joining the Laboratory, he focused primarily on Windows desktop and datacenter technologies. After being introduced to VMware virtualization in 2010, he quickly became familiar with the architecture and administration of the key components and served a central

role in expanding VMware use within the groups he supports. In 2013, he helped set up the Homeland Protection and Air Traffic Control Division's multitenant virtualization infrastructure that is used by all the groups in the division, and he now serves as the lead administrator for the infrastructure. His current work focuses on virtualization technologies for the data center and desktop, data center computing, and network engineering with an emphasis on automation techniques for today's cloud environments.

management for the Air Traffic Control Systems and the Informatics and Decision Support Groups. His team has applied the techniques used in supporting the FAA programs to the design and setup of the Lincoln Secure Environment. He holds a bachelor's degree in computer science from Saginaw Valley State University.



**Edward F. Griffin** is a system engineer in the Air Traffic Control Systems Group. He joined Lincoln Laboratory in 1998 after receiving a bachelor's degree in engineering from the University of Southern California. His focuses are on the design and implementation of data center networking and on computing and

storage technologies to facilitate the operation and reliability of the various programs in the Air Traffic Control Systems and the Informatics and Decision Support Groups. He has contributed to many Laboratory-wide initiatives and committees, helping to shape the direction of computing and networking in the Laboratory. Using the skills derived through these activities, he assisted in the design and implementation of the technology used in the Lincoln Secure Environment.



**William L. Pughe** is a member of technical staff in the Informatics and Decision Support Group. He joined the Weather Sensing Group at Lincoln Laboratory in 1996 as a software developer working on weather-detection algorithms for the Terminal Doppler Weather Radar and Weather Systems Processor programs.

Since switching to the Informatics and Decision Support Group, he has been involved in developing the Lincoln Secure Environment, a software-development test bed, and in investigating cyber security issues for industrial control systems. He holds a bachelor's degree in physics from the University of Massachusetts, Amherst.



**Patrick J. Pawlak** is a member of the technical staff in the Air Traffic Control Systems Group. His focus is on the development of robust networks that meet the group's demands for large, externally facing, real-time prototypes that enable the validation of technology in operational settings. Since joining Lincoln Laboratory

in 1988, he has been a key member of several teams responsible for the transfer of experimental systems to the Federal Aviation Administration (FAA) for transition to an operational capability. He is currently part of the teams developing real-time prototypes for the Corridor Integrated Weather System, Consolidated Storm Prediction for Aviation, and National Weather Processor programs. He leads the team responsible for systems and network design and

# Transitions

A ROUNDUP OF LINCOLN LABORATORY TECHNOLOGY TRANSFER OPPORTUNITIES IN CYBER SECURITY

## Technology Transfer

As a federally funded research and development center, Lincoln Laboratory is chartered to make technology available to both government agencies and commercial entities. Here is a sampling of some of the technologies that are ready for this transition.

For more information on these technology transfer opportunities, please email [cyber-tech-transfer@ll.mit.edu](mailto:cyber-tech-transfer@ll.mit.edu).

### Scalable Cyber Analytic Processing Environment (SCAPE)

Network defense requires rapid sensemaking of large amounts of data from disparate sources. This sensemaking is especially challenging in networks that are established ad hoc or that lack enterprise network monitoring and security, and an event management infrastructure. The core problem that the Scalable Cyber Analytic Processing Environment (SCAPE) solves is the following: Given a number of available network sen-

sor data sources, such as NetFlow records from routers, logs from web proxies, and alerts from intrusion-detection systems, how can an analyst with little prior knowledge about the network or the data sources immediately begin ingesting and analyzing the data while continuously refining his or her understanding to enhance downstream data query and analysis?

SCAPE provides data storage and a suite of knowledge engineering, query, analysis, and visualization capabilities to meet an analyst's needs. The technology uses Accumulo, a Bigtable-like NoSQL database, as the storage back-end and provides a variety of out-of-the-box parsers and utility functions for ingesting cyber data. SCAPE pioneers a knowledge-engineering framework, called the Knowledge Registry, that uses domain-specific data types and descriptive tags to describe data sources. For a domain-specific view of the data, SCAPE provides an application programming interface (API) that leverages the information in the Knowledge Registry. As the analyst gains intuition about the data sources through ad hoc data exploration, he or she can

expand the Knowledge Registry with metadata about the sources and, in turn, assert fine-grained control over how data are interpreted and exposed through the API. This iterative process allows the analyst to home in on interesting data by continually tuning the data processing pipeline as new data source relationships are discovered.

SCAPE was developed under the Lincoln Laboratory cyber situational awareness program that is funded by the Assistant Secretary of Defense for Research and Engineering line. It is now available as an open-source project.

### Lincoln Open Cryptographic Key Management Architecture

There is a strong market need for cryptographic technology that is secure and efficient. While modern cryptography offers proven ways to secure applications and devices, it lacks easy-to-deploy and easy-to-use key-management solutions. Making cryptographic keys available to authorized remote devices when needed and securing the keys in storage and in transit are complicated tasks. Existing cryptographic software libraries provide only a partial solution, lacking built-in support for key management and authorized user-identity management. Developers must figure out how to combine low-level cryptographic functions into a secure design that supports all of the high-level security functions required by the application, such as data protection, cryptographic user-identity management, and key management,

and that prevents key development errors resulting in insecure applications and security breaches.

Lincoln Open Cryptographic Key Management Architecture (LOCKMA) provides a seamless solution by combining the following functions into a self-contained, rigorously architected and verified component: powerful cryptography to enable applications to protect their data at rest and in transit over communication channels; standards-based identity management to help applications create, establish, and verify identity credentials; and advanced key-management functions for generating, protecting, and securely distributing cryptographic keys to authorized recipients.

With a simple, intuitive interface, LOCKMA handles all low-level cryptographic functions “under the hood” in a design successfully realized in several advanced military communication applications. LOCKMA is highly portable, is extremely resource efficient, and is decoupled from specific types of operating systems and communication channels. It is beneficial to a wide variety of applications, such as military operations, household management automation, and network security.

LOCKMA focuses on making the addition of strong, usable cryptographic protections to applications as easy and as inexpensive as possible. As such, LOCKMA implements only those algorithms approved by the National Institute of Standards and Technology and the National Security Agency. Furthermore, unlike existing key-management enterprise solutions,

LOCKMA enables devices and applications to secure their data end to end, without having to trust any centralized key servers.

LOCKMA was honored with an R&D 100 Award, was realized as a field-programmable gate array core, was submitted for two U.S. Patent and Trademark Office patent applications, and won an MIT Lincoln Laboratory Best Invention Award.

---

### **Proactively secure Accumulo with Cryptographic Enforcement**

The Proactively secure Accumulo with Cryptographic Enforcement (PACE) project uses cryptographic techniques to enhance the security of the Accumulo database against a malicious server or system administrator. Accumulo, a scalable distributed database, offers fast ingest rates and cell-level access control, giving users the ability to quickly store large datasets and to establish fine-grained access control for authorized users. Because Accumulo is widely used within the federal government, it is important that its stored data cannot be learned, modified, or leaked at the whim of an adversary. The PACE team uses efficient, well-understood cryptographic algorithms to secure Accumulo against threats to stored datasets.

PACE has two primary focuses. The first is to enable users to validate the integrity of their stored data and the results of their queries, ensuring that these results contain only the correct requested data. The second focus is to guarantee the confidentiality of users’ data by providing a flexible encryp-

tion library for Accumulo cells and cryptographically enforcing Accumulo’s access control. The PACE team is also developing a seamless interface with Accumulo’s API to enable users to cryptographically secure aspects of their Accumulo servers with minimal changes to their existing code. This seamless integration, combined with the PACE software’s security guarantees, has already allowed some of the PACE integrity work to be transitioned to a government customer. Looking ahead, Lincoln Laboratory plans to transfer the rest of the PACE technology to the same seamless interface while finding new ways to deliver powerful, usable security to Accumulo users.

---

### **Self-Enforcing Security for the Cloud with Cryptographic Access Control**

Commercial cloud storage offers many benefits, such as data ubiquity, data backups, and low storage costs. However, many users are reluctant to relinquish their data to the cloud because of security concerns, fearing a loss of control over data access and protection. Most current cloud storage services rely on explicit or implicit trusted third parties to protect data and to enforce access control policies that define who can obtain the data and the type of access, e.g., read-only or write-only permission. However, third parties may not be trustworthy because they can be corrupted by insider threats and security breaches.

The cryptographic access control (CryptAC) framework returns data control to users. CryptAC

## Transitions

redefines authentication and authorization by making permissions into “self-enforcing” cryptographic objects, negating the need for a trusted third party. CryptAC can also improve local storage resilience against insider threats; for example, system administrators can manage files on a local system but cannot access the files’ contents without the owner assigning the administrators explicit permissions. If attackers gain access to stored data, they cannot read the information; they can only destroy the data. To mitigate data destruction, CryptAC uses erasure codes that encode data in blocks so that if some blocks are erased (up to a predefined threshold), the data can be reconstructed from the remaining blocks. CryptAC employs erasure codes to efficiently distribute data over multiple clouds, allowing the data to be efficiently retrieved even if some clouds are unavailable or the data are corrupted.

CryptAC works seamlessly with the cryptographic keys stored on Department of Defense (DoD) Common Access Cards (CACs), allowing DoD users to authenticate and manage permissions with ease. Keys derived from other sources, including passwords and biometric data, are also compatible with CryptAC.

CryptAC provides secure support even for traditional access policies, such as file permissions in standard operating systems. More importantly, it not only improves current policy enforcement but also enables technology for developing and supporting novel trust infrastructures that are flexible, explicit, and secure.

### **Timely Randomization Applied to Commodity Executables at Runtime (TRACER)**

When cyber attackers exploit typical programming bugs in common applications, they can obtain and leak sensitive information that determines how a program runs and how it is protected. Researchers have observed numerous advanced persistent threats that bypass modern operating systems’ (OS) defenses. The resulting data leakages are especially difficult to mitigate in proprietary OS, e.g., Windows, and closed-source applications because existing defensive techniques rely on analyzing the source code.

Timely Randomization Applied to Commodity Executables at Runtime (TRACER) is a prototype technology that prevents information-leakage attacks by frequently rerandomizing the encoding of sensitive program data in closed-source applications. The rerandomization is tied to program outputs, e.g., network packets. When the program generates and releases an output, allowing an attacker the opportunity to steal and potentially leak information, all sensitive regions of the program are rerandomized. As a result, any leaked program data immediately become stale and unusable.

TRACER can work with proprietary applications, such as Adobe Reader, Internet Explorer, and Java, on top of Windows without requiring the source code or modifying the OS. TRACER also minimally impacts performance; the current prototype does not add a noticeable slowdown to protected applica-

tions. It has been tested on a variety of popular applications, including Adobe Reader, Internet Explorer, Firefox, and Adobe Flash.

TRACER prevents sophisticated attacks that can otherwise bypass OS defenses as has been observed in many persistent attacks.