REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
12/12/2016	Final	01/01/2015-08/31/2016

4. TITLE AND SUBTITLE
Cyber Moat: adaptive virtualized network framework for deception and disinformation

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-15-1-2012

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Sun, Kun

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The College of William and Mary, Office of Sponsored Programs, P.O. Box 8795, Williamsburg, VA 23187-8795

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research

875 N Randolph St, Arlington, VA 22217

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is Unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Given limited resources including CPU cycles, memory, disk storage and network bandwidth, we will design a scalable decoy system that can not only minimize resource usage (i.e. a large number of decoys can be deployed simultaneously), but also provide effective protection for the real system. In general, there are three design goals of our decoy system: scalability, fidelity, and isolation. This project will develop a set of optimized, validated, and fully documented algorithms and mechanisms. A successful system prototype will be delivered and lead to a powerful new capability for using moving target defense mechanism to build resilient, adaptive, and secure systems.

15. SUBJECT TERMS
Moving Target Defense, Decoy, Dynamic Virtualized Network

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kun Sun
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	8	703-993-1715

"Cyber Moat: Adaptive Virtualized Network Framework for Deception and Disinformation"

PROJECT: ONR-BAA-15-001
Contract #: N00014-15-1-2012

Final Technical Report
Reporting Period: 1 January 2015 – 31 August 2016

Principal Investigator: Dr. Kun Sun
Department of Computer Science, P.O. Box 8795
College of William and Mary
Williamsburg, VA 23187-8795
757-221-3457 (voice); 757-221-1717 (fax); ksun@wm.edu

Project Manager:
Dr. J. Sukarno Mertoguno
Mathematics Computers and Information Research
Office of Naval Research
875 N. Randolph Street, Suite 1425
Arlington, VA. 22203

Contents

1 Major Goals

Current enterprise networks are usually built with static topology and fixed network configuration. An attacker has long enough time to scan and collect the network information and develop sophisticated exploits for future attack. We propose to design and develop an adaptive deception and disinformation system called ***Cyber Moat*** that protects a real system through hiding it in a continuously morphing virtual network and serves disinformation (false statements) with a novel Decoy Master/Agent mechanism to defeat both fingerprinting-based and timing-based decoy detection techniques and thus convince the trapped attackers of untruth. The effort for providing virtual indistinguishable properties for the mirage nodes can be very expensive, as each mirage node needs to function as a proper and active computing system. To achieve a tradeoff between system security and system performance, we propose a hybrid decoy system where one high interaction *decoy master* ***(DM)*** is responsible for generating responses to the original requests from the adversary and forwarding them to a group of low interaction *decoy agents* ***(DA)***, which then performs a unique personality transformation function to prepare the packet headers for responses. Using decoy master/agent mechanism, we can have a large number of low interaction decoys that are indistinguishable from real systems. Moreover, we need to guarantee that compromises of decoy agents will not lead to the compromise of the decoy master.

Given limited resources including CPU cycles, memory, disk storage and network bandwidth, we will design a scalable decoy system that can not only minimize resource usage (i.e. a large number of decoys can be deployed simultaneously), but also provide effective protection for the real system. In general, there are three design goals of our decoy system:

(1) Scalability. The number of deployed decoys should be large so that the probability of an attacker identifying the real system is minimized.

(2) Fidelity. The decoy system should appear to the attackers almost the same as the real system. Otherwise, the attacker can easily discover the real system based on the difference from the decoys.

(3) Resource and security isolation. The resources of the decoy and real system should be well isolated so that an attacker has no means of identifying their co-location through correlation analysis. Since the scalability requirement indicates the adoption of resource multiplexing and sharing in our implementation, we need to ensure that this would not create covert channels through which the real system can be disclosed.

This project will develop a set of optimized, validated, and fully documented algorithms and mechanisms. A successful system prototype will be delivered and lead to a powerful new capability for using moving target defense mechanism to build resilient, adaptive, and secure systems. The primary deliverable of this effort will be the actual secure system based on moving target defense mechanisms, for deployment on commodity computers. There are no proprietary claims associated with our deliverables.

This report is approved for public release; distribution is unlimited.

2 Accomplishment Under Goals

We perform three major tasks between 01/01/2015 and 08/31/2016 before the PI moved from College of William and Mary to George Mason University (GMU) in 09/2016. Related grant has been transferred to GMU on 09/30/2016.

First, the system should sustain service availability to the authenticated clients when the system changes its topology. The network connections should be maintained when the IP or Port number is changing. Meanwhile, the service downtime should be minimized. We develop a virtual dynamic network framework, which enables network administrators to dynamically change the network topology and the network configuration based on the attackers' activities. We propose a VM-based seamless TCP connection migration scheme to support live VM migration without losing the existing network connections. We port the VM migration kernel module from Linux kernel version 2.4 to 2.6. Moreover, we propose a novel defense mechanism for protecting the identity of nodes in Mobile Ad Hoc Networks and defeat the attacker's reconnaissance efforts. To preserve communication among legitimate nodes, we modify the network layer by introducing a translation service for mapping virtual identities to real identities, a protocol for propagating updates of a node's virtual identity to all legitimate nodes; and a mechanism for legitimate nodes to securely join the network. We show that the proposed approach is robust to different types of attacks, and also show that the overhead introduced by the update protocol can be controlled by tuning the update frequency.

Second, we present the design and development of a cache-assisted secure execution framework, called CaSE, on ARM processors to defend against sophisticated attackers who can launch multi-vector attacks including software attacks and hardware memory disclosure attacks. CaSE utilizes TrustZone and Cache-as-RAM technique to create a cache-based isolated execution environment, which can protect both code and data of security-sensitive applications against the compromised OS and the cold boot attack. To protect the sensitive code and data against cold boot attack, applications are encrypted in memory and decrypted only within the processor for execution. The memory separation and the cache separation provided by TrustZone are used to protect the cached applications against compromised OS.

Third, we need to mitigate the threat from insiders. Though the system only allows authenticated clients to locate and access it service, insiders (as authenticated clients) can still connect to the real system and perform attacks. We develop an anti-crawler mechanism called PathMarker to detect and constrain the distributed persistent inside crawlers that have valid credentials to access the web services. The main idea is to add a marker to each web page URL and use the URL path and user information contained in the marker to help accurately detect crawlers at its earliest stage. PathMarker can dramatically suppress the efficiency of distributed crawlers and effectively reduce the crawling speed of individual persistent crawler.

2.1 DESIR: Decoy-Enhanced Seamless IP Randomization [1]

Sophisticated adversaries usually initiate their attacks with a reconnaissance phase to discover exploitable vulnerabilities on the targeted networks and systems. We develop a decoy-enhanced seamless network address randomization mechanism called DESIR to defeat network reconnaissance attacks and ensure service availability. First, we fortify the IP randomization technique with a large number of decoys to protect the servers against reconnaissance attacks.

Besides the real servers, we deploy a number of decoy nodes that will change their IP addresses along with the real servers. Decoys have been widely used to distract attacker's attention from the real system; however, APT attacks may eventually identify the decoy nodes based on their response time and fingerprint analysis after interacting with the decoys. In our solution, in addition to deploying a large number of decoys, we randomly shuffle the IP address space of the target network including both the real servers and the decoys. Therefore, though the attacker may create a blacklist of decoy IP addresses through reconnaissance, this blacklist becomes invalid after the next round of IP randomization, and the attacker has to start over the reconnaissance process. In other words, we combine both IP randomization technique and decoy technique to effectively defeat persistent reconnaissance attacks, though neither of them can achieve this goal by itself only.

Our theoretical analysis shows that decoy-enhanced IP randomization can effectively prolong the attacker's scanning time. Suppose one real server is protected by n IP addresses, where n – 1 IP addresses are occupied by decoy nodes. When the attacker is not aware of our defense mechanism, it may only scan the entire IP address space once either sequentially or randomly. In this scenario, our IP randomization technique can increase the average number of probes from 0.5n to 0.63n. When the attacker knows that the real system is protected by our defense system, it may scan the entire IP address space multiple times, and it will increase the average number of probes from 0.5n to n.

We implement a virtual machine (VM)-based prototype that integrates decoy-enhanced IP address randomization with seamless connection migration. The experimental results show that the overheads for both decoy deployment and IP randomization are reasonably low and can defeat the practical scanning attacks using tools such as Nmap or ZMap. The results of this work have been published in IEEE Infocom 2016 [1].

2.2 CaSE: Cache-Assisted Secure Execution on ARM Processors [2]

To enhance the security of embedded systems, ARM provides a hardware security extension named TrustZone to protect sensitive code and data of applications in an isolated execution environment against a potentially compromised OS. However, the design of TrustZone cannot prevent physical memory disclosure attacks such as cold boot. Since mobile phones are frequently stolen, when attackers have physical access to the mobile devices, they can gain unrestricted access to the contents in the DRAM. Unfortunately, TrustZone does not enforce encryption of memory in the privileged environment like SGX. As a result, sensitive information, such as cryptographic key material, is not secured even if it is stored in TrustZone protected physical memory when adversaries have physical access to the mobile device.

In this work, we propose a cache-assisted secure execution system called CaSE that can protect against both software attacks and physical memory disclosure attacks on ARM-based devices. The basic idea is to create a secure environment in the CPU cache and use TrustZone to prevent the potentially compromised OS from accessing the secure environment. Thus, CaSE can protect both confidentiality and integrity of the application's code and data against both software attacks and physical memory disclosure attacks. To protect against physical memory disclosure attacks, CaSE creates an execution environment inside the ARM processor by loading and executing an

application completely within the CPU cache. Cache is designed to be a hardware mechanism that is transparent to the system software except for a small number of maintenance instructions. Therefore, we solve several challenges to create a cache-assisted execution environment.

First, to make computation SoC-bound, the application code, data, stack and heap have to be stored in and only in the cache. The memory for each component in the application address space has to be allocated carefully to eliminate cache contention. Unfortunately, none of the publicly available ARM documents details the mapping from memory addresses to cache line indexes. In order to correctly place and optimize application memory in the cache, we design and perform experiments to obtain cache mapping schemes of the targeted hardware platform.

Second, once the application is loaded in the cache, we make use of the hardware-assisted cache locking function to pin down portions of the cache, without significantly impacting the system performance. With the ability to control eviction policy on cache lines that store the sensitive data, it is possible to enable context switching between the protected application and the rest of the system without concerning the execution of other programs will cause eviction of the sensitive contents from cache to DRAM.

Third, since the application is still encrypted when loaded into DRAM, it needs to be decrypted completely within cache before being executed. In many processor architectures, including ARM, instruction cache and data cache are not guaranteed to be coherent. When an application decrypts its own code back into the process address space, instruction cache and data cache become incoherent. Such issue of incoherent cache caused by self-modifying programs is often resolved by flushing the cache. In CaSE, flushing the cache fails our efforts of running applications entirely inside the SoC. To solve this problem, we synchronize the incoherent data cache and instruction cache by utilizing the unified last level cache in the processor.

TrustZone is used to protect the cache-assisted isolation environment against an untrusted OS. Cache lines in TrustZone-enabled ARM processors are built with an extra non-secure (NS) bit to indicate whether the line belongs to the secure world or the normal world. Therefore, the rich OS in the normal world cannot access or manipulate the cache lines used by the secure world. The secret key to decrypt the application is saved in the secure world cache. Without the key, a compromised rich OS cannot decrypt the application code, which may be misused by attackers to reverse engineer proprietary algorithms or find potential vulnerabilities. CaSE offers two running modes depending on whether secure world cache or normal world cache is used to create the environment for the SoC-bound execution. These two modes provide a trade-off between the system security and the run-time performance.

We implement a prototype of CaSE on the i.MX53 running ARM Cortex-A8 processor. Using the CaSE, we show that it is possible to execute a kernel integrity checker and a suite of cryptographic algorithms including AES, RSA, and SHA1 in the cache with small performance impacts. The results of this work have been published in IEEE Security& Privacy conference 2016.

2.3 PathMarker: Defending Persistent Malicious Crawlers [3]

With the prosperity of Internet, data collection from the network has gained increasing demands. As one type of bots, web crawlers have been leveraged by search engines (e.g., Googlebot by Google) to popularize websites through website indexing. However, the number of malicious bots is increasing too. To regulate the behavior of crawlers, most websites include a file called "robots.txt" that contains the rules on what information cannot be collected by the web crawlers. The file may even set different rules for different crawlers. However, "robots.txt" only provides a guideline, and almost all malicious robots ignore it. Moreover, since this file is publicly available, malicious crawlers may use it to infer the locations of valuable web contents that the servers want to protect.

For websites that require open user sign-up and login to access more website contents (e.g., sourceforge.net, stackoverflow.com), they don't want to see a small number of free user accounts can successfully download all their web contents. For websites that may contain confidential information or paid documents (e.g., www.sciencedirect.com, ieeexplore. ieee.org), though only authorized or paid users are allowed to access the websites, they still need to prevent authorized users as insiders from collecting the entire website. Particularly, it is a challenge to detect and constrain distributed persistent crawlers against those websites. First, since the crawlers are persistent, they can afford to lower the download rate and better mimic the access behaviors of real users. Thus, it is critical to detect a persistent crawler accurately at its earliest stage. Second, a number of users may coordinate and use a divide-and-conquer strategy to speed up the crawling tasks. Defenders should suppress the efficiency of distributed crawlers to minimize the information leakage before the crawlers are detected.

In this work, we develop an anti-crawler mechanism called PathMarker to detect and constrain the stealthy distributed persistent crawler. The main idea is to add a marker to each web page URL and use the web page path and user information contained in the markers to help identify and confine crawlers. Given one website, we can automatically append a marker to each web page's URL, and we call it URL marker. A URL marker records the information about its parent web page's URL and the user ID who collects the URL. Thus, when distributed crawlers share collected links in a pool, we can detect them through a user ID mismatch, since the user who collects the page may not be the same as the one who visits the URLs contained in this page. Moreover, we encrypt the URL marker along with the original URL except the root URL to further protect the website structure against distributed crawlers, who will see different web links about the same web page to different user IDs.

With the aid of URL marker, we can calculate the depth and width of each page and build accurate URL visiting path based on parent URL recorded in the URL marker. Next, we can leverage machine learning techniques to detect crawlers based on the different URL visiting path patterns and URL visiting timings between human beings and malicious crawlers. We adopt Support Vector Machine (SVM) to model the normal users and crawlers using the features related to the URL marker. Moreover, to lower the false positive rate, we rely on CAPTCHAs to differentiate normal users from crawlers.

We develop a PathMarker prototype on an online forum website. Since our solution needs to make small changes on the source code of web servers, it is difficult for us to test on public large-size or medium size websites. Instead, we setup a forum website from scratch and integrate our anti-crawler mechanism. We first train a SVM model based on the logging data collected

from more than 100 normal users and 6 in-house crawlers, and then test the model using 6 open source crawlers and another set of normal user data. The experimental results show that our anti-crawler technique can effectively detect all the crawlers. Moreover, two external crawlers, Google bot and Yahoo Slurp, are also detected.

3 Training Opportunities

Nothing to report.

4 Results Dissemination

Based on this project, we submitted another proposal to Cisco Systems, Inc. Since Moving target defense is an emerging critical research area, Cisco awarded the PI an unrestricted gift of $100K to help develop the final hybrid decoy system.

5 Honors and Awards

Nothing to report.

6 Participants

PI: Kun Sun
Postdoc: Lingguang Lei
PhD students: Jianhua Sun, Shengye Wan, Ning Zhang

7 Publications

[1]. Jianhua Sun and Kun Sun, DESIR: Decoy-Enhanced Seamless IP Randomization. To appear in IEEE International Conference on Computer Communications (INFOCOM), San Francisco, CA, April 10-15, 2016.

[2]. Ning Zhang, Kun Sun, Wenjing Lou, and Tom Hou. "CaSE: Cache-Assisted Secure Execution on ARM Processors." To appear in the 37th IEEE Symposium on Security and Privacy (S&P), SAN JOSE, CA, MAY 23-25, 2016.

[3]. Shengye Wan, Yue Li, Kun Sun, PathMarker: Defending Persistent Malicious Crawlers, Under submission to conference. November 2016.