REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 12/12/2016	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* 07/01/2015-08/31/2016

4. TITLE AND SUBTITLE Design and Implementation of Decoy Enhanced Dynamic Virtualization Networks	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER N00014-15-1-2396
	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
Sophisticated adversaries usually initiate their attacks with a reconnaissance phase to discover exploitable vulnerabilities on the targeted networks and systems. This attacking strategy works well due to the static nature of the network topology. To mitigate the effectiveness of reconnaissance attacks, we propose to develop a defensive mechanism that dynamically mutates network topology with a large number of decoys to invalidate the attacker's knowledge from network scanning. In this work, we focus on solving two major challenges associated with dynamic network topology, namely, service availability to legitimate users and service security against unauthorized users.

15. SUBJECT TERMS
Moving Target Defense, Decoy, Network Function Virtualization

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

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

"Design and Implementation of Decoy Enhanced Dynamic Virtualization Networks"

PROJECT: ONR-BAA-15-001
Contract #: N00014-15-1-2396

Final Technical Report
Reporting Period: 1 July 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 **3**

2 Accomplishment Under Goals **3**

2.1 CyberMoat: Camouflaging Critical Server Infrastructures with Large Scale Decoy Farm [1] *4*

2.2 SPEAKER: Split-Phase Execution of Application Containers [2] *6*

3 Training Opportunities **7**

4 Results Dissemination **7**

5 Honors and Awards **7**

6 Participants **7**

7 Publications **8**

1 Major Goals

The relatively static configurations of networks and their hosts allow attackers to gather intelligence, perform planning, and execute attacks at will. This project's overall objective is to protect the real system running on a host computer via mutable virtualized large-scale network containers. Specific objectives include (1) increasing the attack surface and hiding the real system in a large number of decoys, and (2) developing dynamic virtualized network topology on a host computer. If successful, our host immunity system could be used by the Navy to better protect its computer system from the existing attacks and enhance the resilience of the services and applications when it is under attacks. Even if an attacker gains some access to the host, his ability to exploit the penetration is limited because what he obtained is no longer true. As time goes on our system knows more about the attacker while he knows less about our system.

This project focuses on developing 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. The software will be delivered as a series of software drops with incremental capabilities. 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 two major tasks between 07/01/2015 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, deception based cyber defense mechanisms have been largely constrained by the low decoy fidelity, the poor scalability of the decoy platform, and the static decoy configurations. Attackers greatly benefit from the unsophisticated decoy design, which allows them to identify and bypass the deployed decoys. In this paper, we develop a decoy-enhanced defense framework that can proactively protect critical servers against targeted attacks through deception. To achieve high decoy fidelity and scalability, our system follows a hybrid architecture that separates lightweight yet versatile front-end proxies from backend high-fidelity decoy servers. Moreover, our system can further invalidate the attackers' reconnaissance through dynamic proxy address shuffling. To guarantee service availability, we develop a transparent connection translation strategy to maintain existing connections during shuffling. Our evaluation on a prototype implementation demonstrates the effectiveness of our approach in defeating attacker reconnaissance and remote exploitations and shows that it only introduces small performance overhead.

Second, Linux containers have recently gained more popularity as an operating system level virtualization approach for running multiple isolated OS distros on a control host or deploying large scale microservice-based applications in the cloud environment. The wide adoption of containers as an application deployment platform also attracts attackers' attention. However, Docker lacks a method to obtain and customize the set of necessary system calls for a given application. Moreover, we observe that a number of system calls are only used during the short-term booting phase and can be safely removed from the long-term running phase for a given application container. In this paper, we propose a container security mechanism called SPEAKER that can dramatically reduce the number of available system calls to a given application container by customizing and differentiating its necessary system calls at two different execution phases, namely, booting phase and running phase. For a given application container, we first separate its execution into booting phase and running phase and then trace the invoked system calls at these two phases, respectively. Second, we extend the Linux seccomp filter to dynamically update the available system calls when the application is running from the booting phase into the running phase. Our mechanism is non-intrusive to the application running in the container. We apply SPEAKER to the most popular web server and data store application containers from Docker hub, and the experimental results show that it can reduce more than 50% and 35% system calls in the running phase for the data store and the web server application containers with negligible performance overhead.

2.1 CyberMoat: Camouflaging Critical Server Infrastructures with Large Scale Decoy Farm [1]

Recent years have witnessed the explosion of targeted attacks against the critical server infrastructures within both government organizations and businesses, and the number of data breaches increases tremendously with the continuous proliferation of exploitable zero-day vulnerabilities. Particularly, an advanced persistent threat (APT) may enable an unauthorized attacker to bypass the traditional security measures such as firewalls and intrusion detection systems (IDS) and stealthily gain access to the sensitive data on the victim servers.

Consequently, deception-based techniques have re-emerged as an additional line of defense to supplement the traditional preventive security measures. Instead of relying on monitoring abnormal attack patterns, deception-based techniques use advanced luring techniques and engagement servers to entice an attacker away from the valuable servers. As the average time to identify and resolve a data breach for malicious attacks is 82 days, we can utilize specially crafted decoys to detect attackers throughout the kill-chain cycle and prevent them from completing their missions. Specifically, a decoy provides a carefully isolated environment for misdirecting attackers and feeding them disinformation in the form of falsified data such as fake encryption keys, database entries, and OS fingerprinting information. It also creates a trapped environment to gather information about the attack and trigger an alarm of the forthcoming intrusion.

Most existing deception-based defenses adopt traditional honeypot technology to develop decoy systems for attack detection and information gathering. However, their effectiveness has not met the expectations of the security practitioners due to two major limitations. First, the number of believable decoys is constrained by the limited system resources. There is a trade-off between the fidelity of the decoys and the limited system resources. Low-fidelity decoys require less system resources but may be easily identified by attackers. In contrast, high-fidelity decoys may not be easily identified by attackers but require more system resources to mimic the real server. Second, decoys are statically deployed. With sufficient support, APT attackers may finally identify all static decoys through either timing-based or fingerprinting-based analysis. It is critical to design believable, scalable, and dynamic decoys to solve these two challenges.

We design a decoy-based deception mechanism named CyberMoat that can protect critical servers against targeted attacks with a large number of high-fidelity decoys, which can be dynamically created and managed using limited system resources. CyberMoat adopts a hybrid architecture that separates an extensive layer of front-end proxies that focus on network stack processing from back-end decoy servers that focus on serving the service requests. Since the size of the proxies is much smaller than the decoy servers, we can create a proxy pool that consists of hundreds (or even thousands) of lightweight proxies, which transparently redirect the network traffic between the attackers (or legitimate users) and the backend decoy servers (or protected servers). We named the proxies redirecting normal traffics as secret proxy, and the proxies redirecting malicious traffics as public proxy. We note that both secret proxies and public proxies have the same network functions. The high-fidelity decoy servers closely resemble real servers with full-fledged operating system and services but only provide false information. Due to the resource constraints, we may only create a small number of high-fidelity decoy servers in a decoy server farm.

To further defeat APT attackers, we dynamically mutate the proxies' addresses to invalidate attackers' efforts of identifying and then blacklisting the decoy servers. By shuffling the proxy addresses, we can not only diffuse the traffic targeting at an overloaded proxy, but also invalidate the attacker's knowledge gained from prior network scanning. Indeed, network address shuffling can reduce the effectiveness of reconnaissance for vulnerability discovery and further raises the bar of remote exploitation based attacks. Since we shuffle both the secret proxy and the public proxy, existing network connections will be disrupted when shuffling the proxy's IP and MAC addresses. For legitimate users this implies degraded user experience; while for potential attackers, it creates an obvious channel of discovering our shuffling enhanced deception mechanism so that they can retreat early to avoid exposing their attack strategies. To guarantee service availability, we develop a transparent connection migration strategy so that the previously alive sessions are not interrupted during proxy address shuffling.

We implement a CyberMoat prototype. First, we use ClickOS [3] as the front-end proxies and use full-fledged Xen virtual machines as the back-end decoy servers. The small size of ClickOS enables us to create hundreds of proxies on one computer. Moreover, it has short boot time. It also ensures secure isolation among the proxies by the hypervisor. The proxies redirect connection requests to the back-end decoy servers that install the same set of software stack as the real server. Second, based on the control plane of the software-defined networking, we design a centralized CyberMoat control plane to manage the proxy address shuffling and seamless connection migration. It is responsible for coordinating the creation of a hybrid decoy platform by communicating with agents in the proxy pool and the decoy server farm. It also performs traffic monitoring to direct the dynamic proxy address shuffling and ensure transparent network connection migration for maintaining existing network connections during the proxy shuffling. We evaluate the effectiveness of our prototype in disrupting network reconnaissance and remote exploits and show that our system introduces small performance overhead.

In summary, we have made the following contributions. First, we develop a dynamic deception mechanism that protects critical production servers with a large scale high-fidelity decoy pool, which can be dynamically configured to further trap and misinform targeted attacks. Second, we develop a highly scalable decoy system that combines an extensive layer of proxies with high-fidelity decoy servers. Through this hybrid decoy design we can successfully misdirect the attackers and entice them away from the protected servers. Third, we introduce a proactive approach of randomizing the decoy network addresses to invalidate the attacker's reconnaissance efforts and diffuse the network flooding against a single proxy. Moreover, we outline a strategy for seamless connection migration to maintain those existing connections during the randomization process. Fourth, we implemented a system prototype for our defensive framework, and our evaluations demonstrate that the proposed approach can effectively evacuate critical servers from remote exploit based attacks and introduce small performance overhead.

2.2 SPEAKER: Split-Phase Execution of Application Containers [2]

We use Linux container to create a large number of decoy agents and develop a fine-grained control on the allowable system calls in each container. Linux containers have emerged as one popular operating system level virtualization approach for running multiple isolated Linux systems on a host OS or deploying large scale microservice-based applications in the cloud environment. State-of-the-art container technologies such as Docker and Rocket are enabling the wide adoption of application containers. According to the latest cloud computing trend, the overall Docker container adoption has doubled through the past year.

A number of security mechanisms have been proposed or adopted to protect containers. Since the system calls are the entry points for processes in the container into the kernel, seccomp has been integrated into the most popular container management tool Docker to effectively limit the system calls available to the container. In Docker, a whitelist of available system calls is recorded as a seccomp profile file in the json format. However, Docker lacks a method to obtain and customize the set of system calls in the seccomp profile for a given application. It only provides a coarse-grained setting recommendation that allows 313 system calls for all application containers.

We observe that an application container may require different sets of available system calls during the lifetime of this container. Since most of the application containers are used to run long-term services such as web servers and database servers, the lifetime of those containers can be generally divided into two phases, the booting phase and the running phase. The booting phase is responsible for setting the container environment and initializing the service. Typically it takes less than 70 seconds. In the running phase, the service begins to accept service requests and send back responses. Due to the different functions demanded in these two phases, a number of system calls invoked in the booting phase may no longer be needed in the application running phase. For instance, compared to the default 313 available system calls through the entire lifetime of a Docker container, our experiments show that 116 system calls are invoked in the container booting phase and only 58 system calls are necessary for the long-term running of the MySQL database server. However, it is a challenge to dynamically change the sec- comp profile to support different sets of systems calls, since Docker only allows the seccomp profile to be set once when the container is being launched.

We develop a dynamic system call reduction mechanism called SCARCE to enhance the security of application containers. The basic idea is to first trace the system call profiles by monitoring the container execution and then dynamically constrain the available system calls when the container boots up and runs. Both system call profiling and dynamic setting run outside the container. For a given application container, it first traces the only necessary system calls required for the booting phase and the running phase, respectively. In the Linux kernel, the available system call list can be represented as an seccomp filter. Therefore, when one container runs, we may dynamically configure the seccomp filter of the container with two sets of available system calls for two different phases. Linux kernel provides two sys- tem calls prctl and seccomp for one container to change its own seccomp filter. However, due to the security reason, we cannot allow processes inside one container to change its available system calls. Instead, SCARCE changes one container's seccomp filter from an outside process running on the host OS. Since all processes inside one container share the same seccomp filter, we can change the seccomp filter of one process to update the available system calls for the entire container. Since the changes happen outside the container, we need to fill the semantic gap to find the data structure of seccomp filter before we can make changes.

We apply our mechanism on two popular categories of application containers from Docker hub - web server containers and data store containers. We select the top four web server container images (i.e., nginx, Tomcat, httpd, and php) and the top four data store container images (i.e., MySQL, Redis, MongoDB, and Postgres), which cover most of the popular web server platforms and data store plat- forms. The experimental results show that SCARCE can reduce more than 50% and 35% system calls for the data store containers and the web server containers, respectively. The number of system calls for web server containers may vary when deploying different web applications; however, they share most of system calls since the primary functions of web servers such as processing HTTP requests and web pages are the same. Actually, for all website applications tested in our experiments, about 80% system calls will be invoked for just fetching one web page.

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, Kyle Wallace

7 Publications

[1]. Jianhua Sun and Kun Sun, "CyberMoat: Camouflaging Critical Server Infrastructures with Large Scale Decoy Farm". Under submission to conference. December 2016.

[2]. Lingguang Lei, et al., "SPEAKER: Split-Phase Execution of Application Containers". Under submission to conference. November 2016.

[3]. J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "Clickos and the art of network function virtualization," in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 2014.