IST-152 Workshop on Intelligent Autonomous Agents for Cyber Defense and Resilience

Moving Target Defenses as a Resilience Strategy

Submitted by Argonne National Laboratory,
a U.S. Department of Energy National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439
630-252-2000

Primary Author:
Nathaniel Evans
Global Security Sciences
Argonne National Laboratory
Argonne, Illinois 60439
Phone: 630-252-3733
Fax: 630-252-2964
nevans@anl.gov

Co-Author:
Michael Thompson
Global Security Sciences
Argonne National Laboratory
Argonne, Illinois 60439
Phone: 630-252-1376
thompsonm@anl.gov

The work described in this paper is cleared for public release.

The paper is technically correct.

This work was supported by the U.S. Department of Energy, Office of Science under contract DE-AC02-6CH11357.

The paper is NATO/PfP Unclassified.

The paper does not violate any proprietary rights.

Introduction

Cyber-attacks pose a major threat to critical infrastructure [1]. For example, web servers are a significant attack vector for would-be malicious actors in cyberspace. Due to the ubiquity of web applications in modern computing, the server software that serves these applications is an attractive vector for would-be attackers.

Active defense describes the activities, methods, and emerging technologies that address the deficiencies of static security controls. Static controls focus on designing a secure architecture and then adding structures and processes — such as firewalls and anti-malware — to protect it from threats. These protections effectively thwart many threats, but threats evolve to exploit weaknesses in these controls or to evade them. Social engineering attacks such as phishing often succeed in evading static controls by enticing people to reveal their credentials or accidentally install malware inside their organization. Active defense also includes activities and technologies that provide additional defensive measures beyond passive and architectural protections. Recently, Moving Target Defense (MTD) strategies have grown in popularity in the computer-security community due to their ability to enhance resilience and force attackers into uncharacteristic behavior.

A common vulnerability in traditional cybersecurity systems is the static nature of defense mechanisms that are often used by programmers and IT personnel. Numerous cybersecurity experts in both academia [2] [3] [4] and the industry [5] have acknowledged the challenges of static defense and have suggested MTD as an ideal solution. MTD systems mitigate the limitations of static defense by creating a dynamic attack surface, which increases uncertainty from the perspective of the attacker(s) as well as the cost and effort that is required to launch an attack.

MTD systems broadly fall into two categories: proactive MTD and reactive MTD. In proactive MTD, possible adversarial behaviors are anticipated, and the corresponding defensive strategies are incorporated into the system design to thwart attacks proactively without disrupting operations. In addition, some elements of the defense system, such as Internet protocol (IP) addresses, port numbers, operating systems, etc., are diversified periodically to create a varying attack surface. In reactive MTD, systems react out of necessity to defend against a detected malicious attack.

MTD techniques are also categorized into shuffle, diversity, and redundancy [6]. The shuffle technique rearranges the configuration of operating systems or applications. The diversity technique randomizes the operating system or software stack components. The redundancy technique requires multiple replicas of network components where paths through the network are randomized. These techniques can be applied either proactively or reactively.

MORE MTD

As cyber threats become increasingly dangerous to critical infrastructure (CI), the development of proactive cyber defenses to enhance the resilience of CI becomes increasingly important. Multiple Operating System Rotational Environment (MORE) MTD acts as a proactive defense strategy that offers increased protection against an attacker being able to probe for and exploit

vulnerable operating systems (OSs). The main goals of MORE MTD are to reduce the number of zero-day exploits on client-server applications, reduce the impact of a successful exploit, and maintain application availability at all times.

The key to the MORE MTD technology is platform diversity; an application is designed such that it can run on multiple OSs. During runtime, various front-end hosts – each running different OSs – are, in turns, switched out of exposure. To accomplish this, we adjust the rotation of the OS for a given time. This ability to control the exposure time – the length of time during which a particular OS is public-facing and thus potentially exploitable – is a direct result of MORE. The results of preliminary experiments that use an exploitable OS to mimic a zero-day vulnerability show that the number of successfully exploited vulnerabilities is directly proportional to time; when the exposure time of an individual OS decreases, the number of successful exploits against the MORE MTD platform decreases.

Even if an attacker is successful in exploiting a host during the exposure time, that exploit lives only for the duration of that exposure time. After the host is rotated out of exposure, it can be isolated, cleaned, and/or forensically examined. Optionally, the host can be cleaned with every rotation so that it is always in an uncontaminated state at the beginning of its exposure window. Consequently, a MORE MTD deployment scenario will have an optimal deployment formula for an OS rotation at a given exposure time, which will affect the adversary's decisions throughout the attack-preparations cycle without affecting application's performance.

Figure 1 illustrates how MORE MTD is implemented:

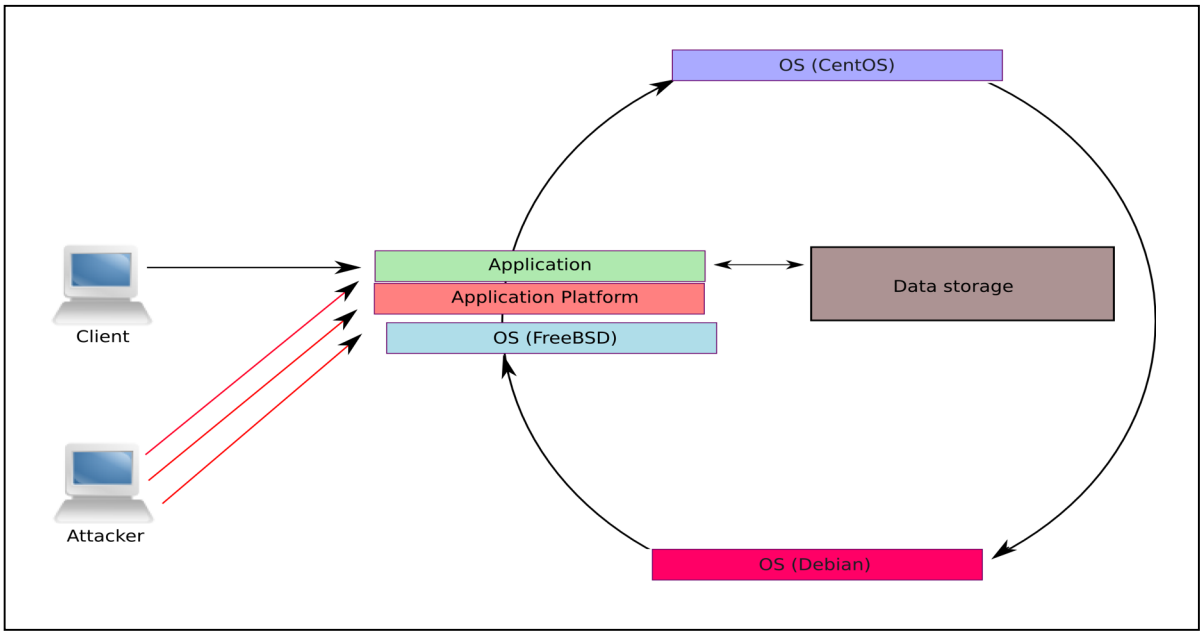

Figure 1: Implementation of the MORE MTD System

The MORE MTD technique is a periodic rotation of the various OSs. The protected application receives external traffic, which is passed to a private network inside the instance, and one of a number of hosts with different OS's is exposed to the web/network at any given time. Figure 2 demonstrates how the rotation works:

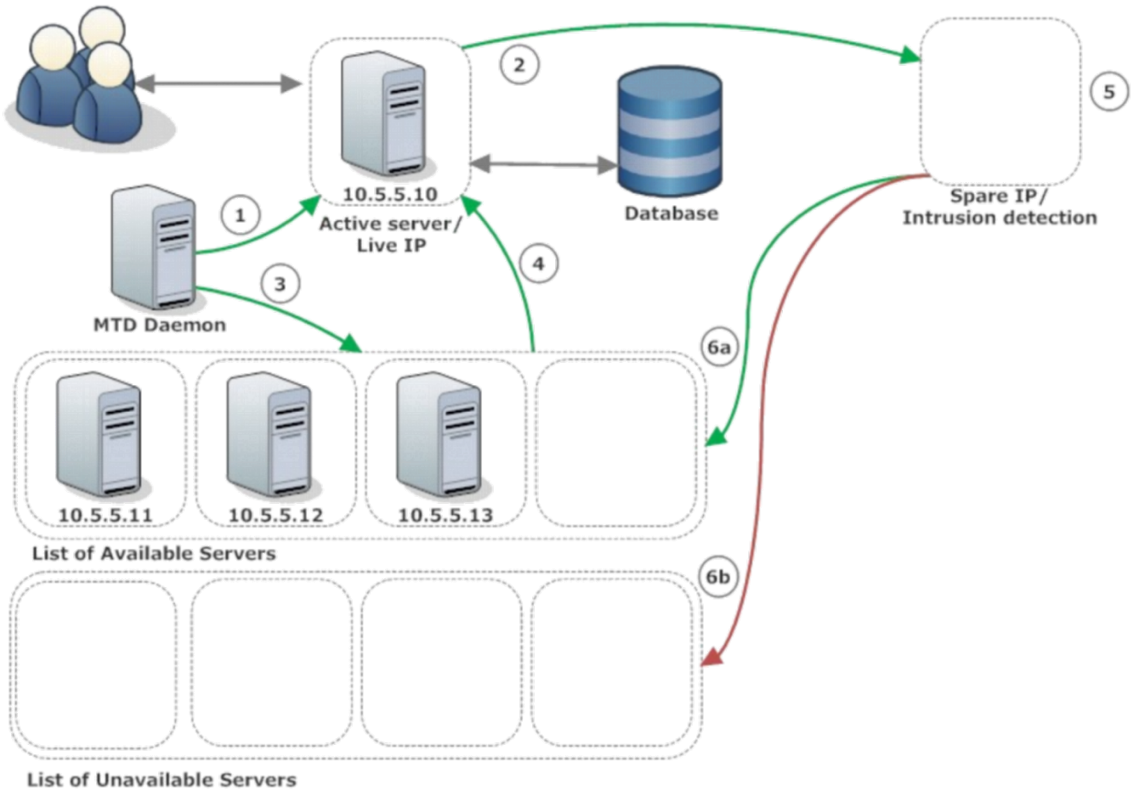

Figure 2: MORE MTD Server Rotation System

The following steps outline the rotation system of MORE MTD:

1. The Daemon establishes an SSH connection with the Live machine.
2. The Live machine is moved to the spare IP for intrusion detection.
3. The Daemon establishes an SSH connection with a selected machine in the list of available servers (selected as a queue or randomly).
4. The selected machine is moved to the active server.
5. Intrusion detection is run on the machine that was taken out of the Live IP.
6. If the Server is not compromised, it is added to the list of available servers.
7. If the Server is compromised, it is added to the list of unavailable servers and will not be placed into the rotation.

Testing on an internal instance of MORE MTD yields positive results and indicates that: (1) the likelihood of a successful attack is reduced by introducing confusion during the reconnaissance phase on a MORE MTD-enabled system; and (2) MORE MTD assures continuity of operations during "zero-day" announcements by allowing the removal and patching of the vulnerable systems. Moreover, initial testing of the techniques shows that the defensive deception mechanism is transparent to the users and introduces minimal functional and performance impacts, thus disrupting adversaries but not the friendly forces.

MORE MTD testing used manual techniques and automated tools to collect configuration attributes in order to plan and perform attacks. While some configuration data were identified,

an actual exploitation of the system failed despite repeated attempts. To verify whether the exploitation tools were functional, MORE MTD was deployed with a known vulnerable OS as one of its rotational hosts. A barrage of penetration tests that were attempted on this configuration confirmed several assumptions: the server exposure time is directly proportional to the success of a penetration attempt; and MORE MTD increases the difficulty in identifying system vulnerabilities, introducing real-world confusion for the attacker while allowing for offline compromised-host remediation.

Though the recent security landscape has been fraught with major vulnerabilities, MTD has shown itself to have validity as a defensive technique. Zero-day vulnerabilities are one of the most difficult problems that security professionals face, as there is no real defense against them. MORE MTD proceeds from the assumption that zero-days will be found in existing systems; however, its proactive nature increases the difficulty for attackers to exploit such vulnerabilities and lowers the consequences of any such exploitation.

Increasing attacker uncertainty and system resilience are themes that will continue to unite the cybersecurity community, and strategies such as MORE MTD that do both are valuable in that context. The next steps are to implement MORE MTD on a larger network. Although current testing showed that the size of the rotation window was the most significant factor in reducing the likelihood of an attack, future studies need to address other factors, such as varying the number of open ports on different platforms and randomizing the order of rotation.

DARE MTD

Dynamic Application Rotation Environment (DARE) MTD uses the two most common and freely available web servers, Apache and Nginx (see Figure 3). It runs a single application on both platforms, redirecting incoming traffic to one server or the other at a random interval. The goal is to mitigate any unknown vulnerability in one of these platforms by reducing the amount of time that platform is exposed to a would-be attacker. Like the MORE MTD strategy, this variability increases the cost of reconnaissance on a target and reduces the likelihood of exploiting any zero-day, or previously unknown, vulnerability.

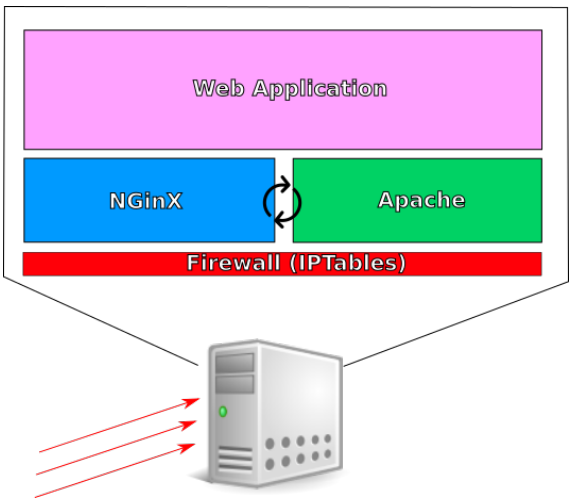

Figure 3: The DARE MTD Mechanism

One virtual machine (VM) is selected at a given time to handle all network traffic, and it is known as the active VM. At a predefined interval, which may be as short as 15 to 30 seconds, the active VM is switched. When a VM becomes inactive, the integrity of the file system is checked for signs of attack and removed from rotation if any integrity compromise is detected. The procedure mirrors MORE MTD, which set the lower rotation window to 60 seconds. The idea of both strategies is to rotate at a suitable interval in order to accomplish the following:

1. Prevent accurate fingerprinting and identification of entry points;
2. Thwart persistent attacks by reducing the exposure of vulnerable software; and
3. Reduce the viability of any gain to the attacker by consuming extra time and resources.

The DARE MTD prototype was deployed in as realistic of an environment as possible. To accomplish this, we used common software and configurations from a typical production environment and tested using common penetration-testing utilities (see Table 1 for testbed components). The DARE MTD Mechanism rotates between two actively running web servers on a single host, both supporting a single, stateless web application.

VM	Software	Purpose
MTD VM	CentOS	Operating System
	Apache HTTPD Server	Web Server #1. Port 82
	Nginx Server	Web Server #2. Port 81
	WordPress	Stateless Web Application
	Web Server Rotation	**Custom Service which Implements MTD using IPTables**
Reconnaissance VM	Kali OS	Operating System
	Nessus/Nmap/OpenVAS	Fingerprinting
	Metasploit Framework	Penetration Testing

Table 1: DARE MTD Testbed Components

For testing purposes, DARE MTD and penetration testing software were deployed on separate virtual machines that were contained on an individual host machine. This allowed for a consistent networking environment and a versatile template that could later be deployed to other testbeds and production environments. Figure 4 describes the networking environment as it existed for tests as outlined below:

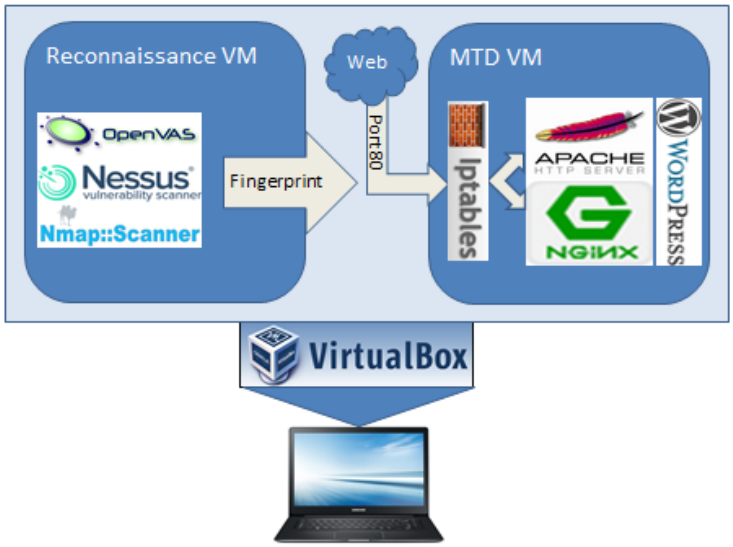

Figure 4: The DARE MTD Networking Environment

DARE MTD takes incoming web traffic (on standard TCP (Transmission Control Protocol) port 80) and redirects it to one of two ports: (1) non-standard TCP port 81 for traffic served by Nginx or (2) non-standard TCP port 82 for traffic served by Apache. The MTD mechanics are as follows:

- Host-based firewall configuration: Linux IPTables is used to redirect incoming web traffic either to the Nginx port or to the Apache HTTPD port.
- Web servers: both Nginx and Apache HTTPD are configured to listen to a localhost on their respective non-standard TCP ports.
- Rotation service: custom script (running as a service), which performs the IPTables rule, updates to redirect the web traffic from TCP port 80 either to the Nginx or the Apache HTTPD TCP port. This update occurs at a random yet bounded time interval.

A production-environment web server typically listens for incoming traffic on port 80 (HTTP). As such, the MTD mechanism can be installed on typical Linux production servers with minimal effort and thus adheres to the MORE MTD ideal. The web servers are configured to listen to localhost, which makes them unreachable on the network. IPTables is configured in a way that only allows traffic on the defined TCP ports. In our design, we have made no assumptions in the enterprise configuration of a system and therefore do not rely on network firewalls or other mechanisms to aid in the Moving Target Defense.

DARE MTD offers significant promise as a proactive defense against web server application-level vulnerabilities. It succeeds both in the goals of increasing uncertainty (as shown in the VM fingerprinting tests) and in increasing resilience (as shown in the exploit mitigation tests). There are a number of unsolved problems with DARE MTD and MTD solutions in general. Though in the present set of experiments we showed that performance is actually increased over a static Apache web server, performance during rotation remains an area that requires further exploration. Additionally, there are several unanswered questions regarding session maintainability, stateless transport mechanisms such as the user datagram protocol, and overall

maintainability with regard to patching and system overhead. However, the conclusion remains that MTD technologies such as DARE MTD have the ability to offer significant benefits for protecting high-value targets.

SS MTD

Cyber infrastructures can be attacked with relative ease, as most of the current infrastructures have configurations with a single, static network stream for communication between any two nodes at a particular time. The aim of Stream Splitting (SS) MTD is to explore and design a TCP moving target defense technique by splitting the payload (data) over multiple TCP streams, making it difficult for the attacker to access the entire communication payload and gain meaningful information. TCP stream splitting (SS) splits a network stream into multiple streams (see Figure 5), making it difficult for an attacker to attack the system by eliminating the advantage of fixed-system configurations and network architecture.

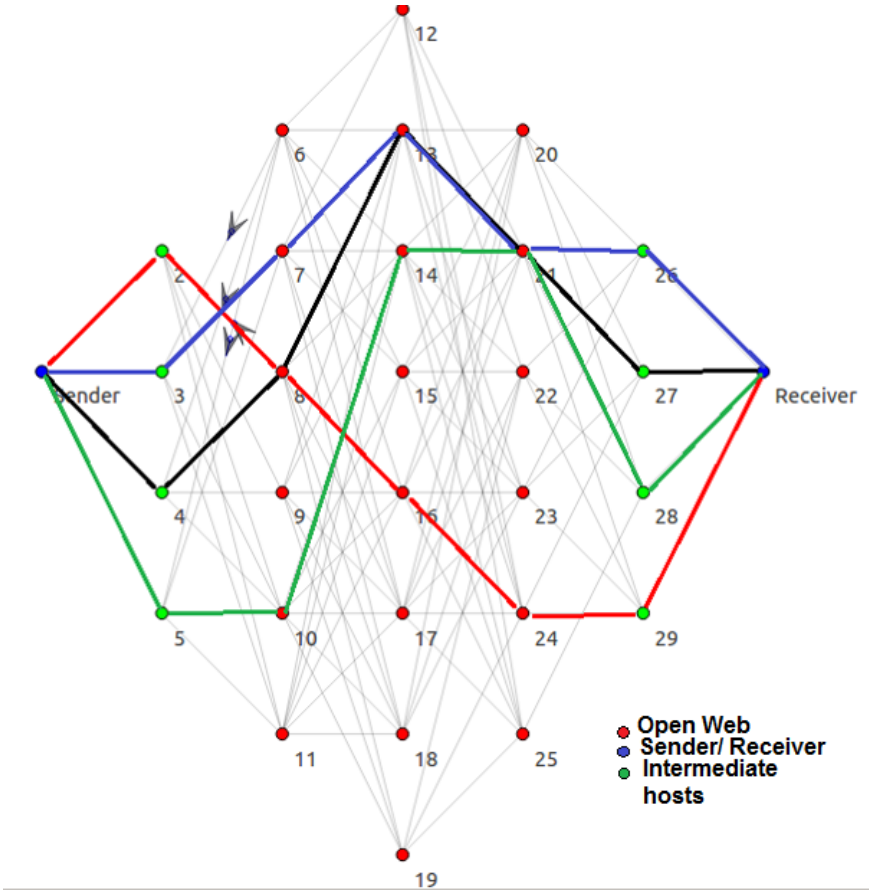

Figure 5: Splitting TCP Network Streams

By mathematically defining a diversity quotient and a desirable state, it follows that we would want to design our software to attempt to reach the desired state.

- At the launch of a network transmission, we compute a diversity quotient Q and compare it to a user-configured threshold.

- If Q is greater than the threshold, the software attempts to increase diversity by adjusting the number of sub-streams.

Counterintuitively, in some cases, reducing the number of sub-streams may actually reduce Q (increasing diversity).

- This could happen when the number of intermediary hosts is limited.
- To combat this phenomenon and allow for increased diversity, we propose the use of on-demand cloud provisioning to increase intermediary host quantity and geographical diversity (many cloud providers allow a user to choose geographical locality).

SS MTD is designed to be media-agnostic. Although initial prototypes focus on traveling over different paths on the Internet using TCP/IP, the protocol itself is designed to operate at the application layer, over any protocol, and over any media. To accomplish this, when not using a stateful, lower-level protocol like TCP, we have a dual-layer reassembly algorithm. Our full-stream reassembly algorithm still focuses on the integrity of the original communications stream, but our sub-stream reassembly algorithm allows for the retransmission of individual packets or chunks without triggering a failure of the channel.

SS MTD is similar to multipath TCP (MPTCP), though it has several key differences. SS-MTD does not aim for layer-4 backward compatibility but rather runs totally at the application layer. This means that there are no flags in the TCP headers to signal that this stream is running over multiple paths. In addition, MPTCP has no way to measure the diversity of the paths on which communication passes.

A proactive MTD anticipates adversarial behaviors by continuously changing in some randomized fashion, while a reactive MTD adapts when an attack occurs as a preventative measure. As discussed above, a proactive approach to cyber defense effectively maintains the integrity and accessibility of the application(s) in use.

Conclusion

This paper presents multiple MTD solutions for hardening security by increasing adversarial uncertainty of the target system. Existing MTD solutions are not practical, and their ability to protect networks from attacks has received little attention in literature; the proposed solutions fill these gaps. Experimental results show that the presented MTD techniques are effective in protecting the server environment from adversarial attacks by reducing the likelihood of reconnaissance and other attacks and by limiting an attacker's access time to a compromised host.

The findings have several implications. First, converting from static defense to dynamic defense through the rotation of system components improves security by reducing the value of, and the window of opportunity for, gathering information and deploying exploits while increasing the cost of doing so. Second, the MTD techniques can be improved further by diversifying other elements of the networking environment, such as applications and database servers.

References

[1] US House, Subcommittee on Cybersecurity, Infrastructure Protection, and Security Technologies, "Examining the Cyber Threat to Critical Infrastructure and the American Economy," Washington: Government Printing Office, 2012.

[2] Cox, B.; Evans, D.; Filipi, A.; Rowanhill, J.; Hu, W.; Davidson, J.; Knight, J.; Nguyen-Tuong, A.; Hiser, J., editors. N-variant systems: a secretless framework for security through diversity: Defense Technical Information Center; 2006.

[3] Colbaugh, R.; Glass, K. Proactive defense for evolving cyber threats. Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on: IEEE; 2011. p. 125–130.

[4] Zhuang, R.; Zhang, S.; DeLoach, S.A.; Ou, X.; Singhal, A. Simulation based Approaches to Studying Effectiveness of Moving-Target Network Defense. National Symposium on Moving Target Research; 2012.

[5] Zank, A. Moving Target Defense: Coronado Group. Available from: http://www.coronadogroup.com/images/Moving-Target-Defense- Coronado.pdf. Accessed June 18, 2012.

[6] Colbaugh, R. and Glass, K., *Proactive Defense for Evolving Cyber Threats*, 2011 IEEE International Conference on ISI, p. 125-130, 10-12 July 2011.