Analysis of Concurrent Moving Target Defenses

Warren Connell
George Mason University
Fairfax, Virginia
wconnel2@gmu.edu

Luan Huy Pham
George Mason University
Fairfax, Virginia
lpham6@gmu.edu

Samuel Philip
George Mason University
Fairfax, Virginia
sphilip4@gmu.edu

ABSTRACT

While Moving Target Defenses (MTDs) have been increasingly recognized as a promising direction for cyber security, quantifying the effects of MTDs remains mostly an open problem. One of the challenges facing MTD quantification efforts is predicting the cumulative effect of implementing multiple MTDs. No single MTD provides an effective defense against the entire range of possible threats. Each MTD has its own set of advantages and disadvantages. We present a scenario where two MTDs are deployed in an experimental testbed created to model a realistic use case. This is followed by a probabilistic analysis of the effectiveness of both MTDs against a multi-step attack, along with the MTD's impact on availability to legitimate users. Our work is essential to providing decision makers with the knowledge to make informed choices regarding cyber defense.

CCS CONCEPTS

• **Security and privacy** → **Network security**; **Web application security**; • **Computing methodologies** → *Model verification and validation*;

ACM Reference Format:
Warren Connell, Luan Huy Pham, and Samuel Philip. 2018. Analysis of Concurrent Moving Target Defenses. In *5th ACM Workshop on Moving Target Defense (MTD '18), October 15, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3268966.3268972

1 INTRODUCTION

The conventional dynamic in cybersecurity favors attackers over the defender. Defenders must protect systems which are relatively static and can be probed for information and weaknesses over a long period of time. Over time, attackers almost inevitably gather enough information and discover enough flaws in system which allow the attacker to conduct a successful attack. Moving Target Defense (MTD) has emerged in recent years to address this conundrum. In contrast to conventional static systems, systems which employ the concepts of MTD are dynamic and reconfigure characteristics over time [17]. Thus, any reconnaissance that an attacker gathers on a system implementing a defense with MTD features will be eventually rendered obsolete. Furthermore, MTDs also include techniques which involve refresh of system components. In the event that an attacker has compromised a system, a MTD using effective refresh techniques would return the system to a previously safe, uncompromised state, forcing the attacker to re-launch attacks to achieve the prior system foothold. The more rapidly the MTD system reconfigures its characteristics or refreshes state, the more rapidly the attacker's gathered intelligence becomes obsolete.

Several efforts have been made to quantify the effectiveness of MTDs [13, 26, 28, 30]. Without common, shared metrics for evaluation, there lacks a means to make informed decisions. In particular, [10] provides a framework that quantifies MTD effectiveness and allows for the possibility of multiple MTDs to be used to provide a broader defense against classes of threats. Indeed, the ability to have multiple MTDs available to protect against certain attacks is crucial. Most MTDs were originally developed to address a specific attack vector or set of attack vectors, such as DoS attacks [18, 19], data exfiltration [29], or SQL injection [7]. This means that a defender must incorporate a variety of MTD techniques in order to provide an effective solution against the broad spectrum of potential attacks.

However, the research and experimental analysis involving the deployment of multiple MTDs is severely lacking. To produce their intended effect, MTDs manipulate system or environment characteristics. Different MTD techniques may have some overlap in characteristics manipulated, resulting in unpredictable behavior. This may negate or compromise the security benefits of the MTDs or cause operational failure. Thus, it is preferred that the MTDs operate independently; that is, the MTDs do not manipulate the same system/environmental characteristics. This can occur when the MTDs operate at different levels of the software stack. Even with this stipulation, the effectiveness of combinations of MTDs is relatively unknown. The major contribution of this paper is a method to quantify the effectiveness of concurrent, independent MTDs deployed in the same system/environment.

MTD defenses also have disadvantages. Nearly all MTDs incur a performance cost [22]. During reconfiguration, the full capabilities of the system are rarely available. Increasing the rate of reconfiguration further increases this overhead cost and reduces the availability of resources, negatively impacting system performance. This effect can be compounded by the implementation of multiple MTDs. Decisions regarding the deployment of MTDs must balance the security benefits with corresponding MTD overhead and implementation costs. This paper does not neglect this aspect of security and offers a method to predict availability with multiple MTDs in operation as well.

Finally, because we are dealing with real-world problems and limitations that exist outside the theoretical realm, we must implement our own MTDs and test them against a realistic attack model.

MTD '18, October 15, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-6003-6/18/10…$15.00
https://doi.org/10.1145/3268966.3268972

Doing so will validate our model and identify important implementation details that might otherwise be overlooked. Our paper presents an experimental framework which models multiple MTDs and an automated attacker to validate our analysis. We then apply the results to a utility function to determine the reconfiguration parameters that offer the best tradeoff of security and availability for the system based on the user's objectives.

In summary, the primary contributions of this paper are: (i) A quantitative analysis of the security benefit of two or more MTDs when used concurrently; (ii) the corresponding availability tradeoff; (iii) an implementation of an experimental testbed incorporating multiple MTDs; (iv) experimental observations of attack scenarios with multiple MTDs; and (v) determining the reconfiguration parameters which offer optimal utility.

The rest of the paper is organized as follows: Section 2 discusses recent, related work in the field. Section 3 describes an overview of the attacker and defender models used, with Section 4 presenting a quantitative analysis in accordance with those models. Section 5 describes the experimental testbed created to implement multiple MTDs, with the results in Section 6. Finally, Section 7 includes concluding remarks and discusses direction for future research.

2 RELATED WORK

MTD techniques continuously change or shift the system attack surfaces, increasing attacker complexity and cost [11]. An attack surface is defined as a "subset of the system's resources (methods, channels, and data) that can be potentially used by an attacker to launch an attack" [21]. Considering these benefits, MTD research is a burgeoning field with a significant body of work already accomplished. Efforts have been made to survey this body of research, define taxonomies of MTDs, and classify existing MTDs using taxonomies based on the MTD technical details of implementation and other characteristics [8, 25].

The general structure of a cyber attack is well-studied; divided into stages collectively known as the Cyber Kill Chain [15]. The initial stage of an attack is especially critical, involving reconnaissance and probing of defenses. This initial stage is critical to allow attacks to gather information regarding network topology, services employed, and weaknesses. The attacker then employs gathered intelligence to tailor the attack against the target to achieve both success and stealth.

MTD techniques, depending on their design, mechanism of operation, and implementation, are designed to interfere with one or more stages of the Cyber Kill Chain. These techniques can be grouped into one of several different classes [25]. Different classes of MTDs are generally observed to focus more on different areas of the Cyber Kill Chain or have other strengths or weaknesses. For example, Dynamic Networks might stop reconnaissance while Dynamic Platforms might be more effective against disrupting attacks or maintaining persistence [24].

Dynamic Network MTDs encompass techniques which dynamically alter network properties, such as network protocols or IP addresses. Random Host Mutation [3], as introduced by Al-Shaer, is a prototypical example in which network communication is performed using ephemeral virtual IP addresses. These ephemeral IP addresses are assigned to individual network hosts and are changed at intervals. Attackers scanning the network or passively observing will observe the ephemeral IP addresses as opposed to actual IP addresses, disrupting reconnaissance efforts. This is a broad category of MTDs with specific implementations, including in Software-Defined Networks [16] and Mobile Ad-hoc Networks [4].

Dynamic Platforms encompass techniques which dynamically alter platform properties, which can include operating systems (type and version), software frameworks and other similar characteristics. In an example, Huang [14], introduces the concept of rotating web services. Users of a web-based application are directed to a load balancer which will randomly direct the user to a VM among a pool of available VMs. Each VM would implement the web application with separate underlying technologies and would periodically rotate VMs. However, this means of operation interferes with applications requiring the maintenance of a persistent state, thus the authors recommended that the technique be employed along with stateless frameworks and protocols. Huang employed a diverse web services layer including Apache, Nginx and Lighttpd web services. Meaningful diversity is critical to ensure that common vulnerabilities do not constitute a significant overlap across platforms.

However, implementing MTDs comes at a cost, either in in terms of additional overheard, lowered availability, or increased response time. These costs may be qualitative, as seen in expert responses in [12], or quantitative. One MTD technique commonly cited in literature is known as IP Hopping. Carroll, et al. studied the effects of this technique and captured the tradeoff in reconfiguration rate and connection loss for this technique [9]. In this work, we attempt to mitigate the losses from IP Hopping by introducing an additional MTD running concurrently and examine the combined effects of multiple MTDs.

In a previous work [10], we established an initial framework to quantify MTD performance and cost, allowing them to be comparatively evaluated. Prior to this framework, most MTD techniques addressed a very narrow set of attack vectors and employed custom metrics, making comparison difficult. This framework allows for combinations of multiple MTDs to be compared with each other. However, that framework assumes that MTDs are independent and does not capture any interactions between multiple MTDs. This work solves that problem by proposing a method for estimating the combined effects of MTDs, on attacker's success rate and availability of the system being protected.

3 ATTACKER AND DEFENDER MODELS

The attack model employed in this paper adheres closely to the expected behavior described in literature through the aforementioned Cyber Kill Chain [15]. The attack consists of multiple steps including several successive reconnaissance steps to identify a vulnerable target, followed by an attack against that target. MTDs can disrupt this attack at several points in the kill chain and are chosen accordingly. The attack model and the corresponding defender model are described in more detail in the following sections.

3.1 Attack Model

The attacker searches the environment for vulnerable operating systems, then searches those targets for vulnerable services, and

then finally launches a corresponding exploit to establish control when such a service is discovered. The simulated attacker, operated by an automation script, performs the following actions, in order:

(1) Target Scan: The attacker employs the Nmap Network Mapper tool to scan the environment. During this initial reconnaissance step, we simulate an attacker which has configured Nmap to perform a relatively non-intrusive scan. This more accurately models the behavior of an attacker which wishes to maintain a degree of stealth while probing the network for weaknesses.
This initial scan process first attempts to detect whether an endpoint is active on a given IP address and the operating system it is running by analyzing the pattern of open ports. Based on the response, the attacker identifies targets for further reconnaissance efforts. Unresponsive targets or those not considered to be vulnerable are not scanned further. Likewise, the attacker does not attempt to perform more invasive scans such as service detection because the attacker seeks to minimize interaction to avoid detection during this step.
For the purposes of this paper, the simulated attacker has identified the characteristics of the Metasploitable virtual machines as promising targets and further actions would only be performed against those targets.
(2) Service Detection: In this step, our simulated attacker has identified promising targets with active services and seeks to probe these targets further, performing a service scan to determine the name and version of the services operating on the node.
For the purposes of our paper, we have focused on web services, as they are a common avenue of attack and many exploits are available. However, it is plausible that a real-world attacker would wish to compromise other active services. Continuing our example, the Metasploitable target profile has port 80 open, corresponding to HTTP web services, and contains a version of Apache with a PHP vulnerability present by default.
If the service running is not considered to be vulnerable, the attacker proceeds to the next target previously identified during the target scan. If the attacker finds that the service running is vulnerable, it proceeds to the next step.
(3) Launch Exploit: If a vulnerable service is discovered, the attacker will launch an exploit against the service. In our example, this results in a reverse connection back to the attacker. If it is not successful, the attacker proceeds to the next target identified by the target scan.
(4) Attacker Success: The phase represents the end state where the now-compromised host has established a reverse connection back to the attacker. This closely corresponds to an attacker which has established full command and control over the target host. In this end state, data can be exfiltrated and other arbitrary actions undertaken. At this point, the overall attack is considered to a success.

Figure 1 depicts these steps in a process flow as performed against a single target IP address. After an attack is complete, if there were other hosts using the target OS found during the initial scan, these hosts are subsequently scanned and attacked in a serial fashion. This models realistic scenarios of attacker/defender behavior. An attacker would be discouraged from launching attacks indiscriminately for several reasons. Indiscriminate attacks against the network space would likely result in more rapid detection of attacker activities. More rapid identification of data breaches has been shown to result in significantly reduced costs to the defender [2].

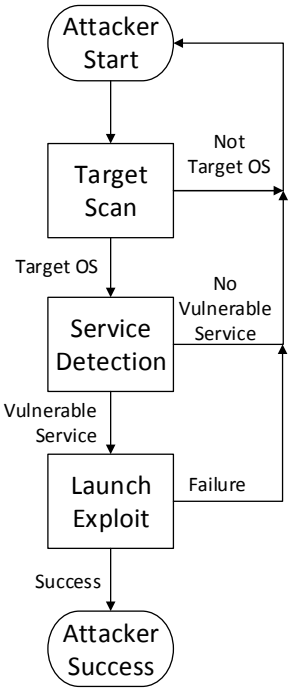

Figure 1: Attack Model

During the attack process, reconfigurations may occur (described in the following section) which interrupt services to the attacker. In these cases, the attack is considered to be a failure.

3.2 Defender Model

Our attacker model requires the attacker to obtain several pieces of information before an attack can commence. This allows for several opportunities for deploy a MTD to disrupt the attacker. The attacker must find a vulnerable service running and must also have an IP address to attack. If the attacker does not obtain both of these, it will move on to the next potential target, if possible. If the attacker does not detect a vulnerable target, then the attack halts and is considered a failure.

Therefore, a reconfiguration which occurs during reconnaissance is likely to cause the attack to fail. Reconfigurations from each MTD occur randomly with interarrival times following an exponential distribution, which is commonly used when determining defender actions because of its memoryless property which prevents the attacker from predicting defender behavior [20, 23]. We also make use of this property in our analysis. The average interarrival time for

reconfigurations is denoted by μ, which may vary between MTDs. In this paper, we may also refer to the reconfiguration rate, which is inversely proportional to the interarrival time and is denoted by $\lambda = \frac{1}{\mu}$.

3.2.1 Service Reconfiguration. For the purposes of Service Reconfiguration, we developed a simple service randomization scheme, very similar to existing dynamic platform-based MTDs [25]. The MTD periodically changes the underlying implementation of the service in a manner transparent to the user.

A target node employing the MTD has multiple implementations of a service available. Upon startup, the MTD randomly chooses one of the available services to run. The service remains active until the automation script initiates a reconfiguration at the interarrival time μ_S. As a part of reconfiguration, the previously-running service is stopped and a new random service started. To increase the randomness and prevent an attacker from predicting patterns in the service changes, the new service chosen may be the same one as before.

All web services were run in the Metasploitable environment, which comes pre-loaded with a version of Apache with a PHP vulnerability present, with the other two web services installed afterward and not containing that specific vulnerability. The following web services were employed:

(1) Apache (containing an unpatched PHP vulnerability)
(2) Nginx
(3) Lighttpd

This particular scheme is heavily influenced the MTD originally proposed by Huang [14], utilizing many of the same web services Huang selected in his own work. A diagram showing the relationship between the attacker and this MTD is shown in Figure 2.

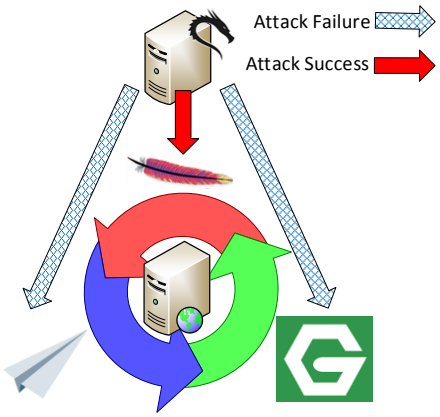

Figure 2: Service Randomization

3.2.2 IP Reconfiguration. For the second MTD evaluated, we implemented IP Reconfiguration using a simple IP randomization technique similar to other known dynamic network-based MTDs [25]. As with the Service Reconfiguration MTD, IP Reconfiguration is accomplished via an automation script. In this scheme, the VM changes its IP address with reconfigurations occurring with an average interarrival time of μ_{IP}. The pool of available IP addresses is equal to that of a Class C network (256), minus a small set of reserved IP addresses for the hardware environment.

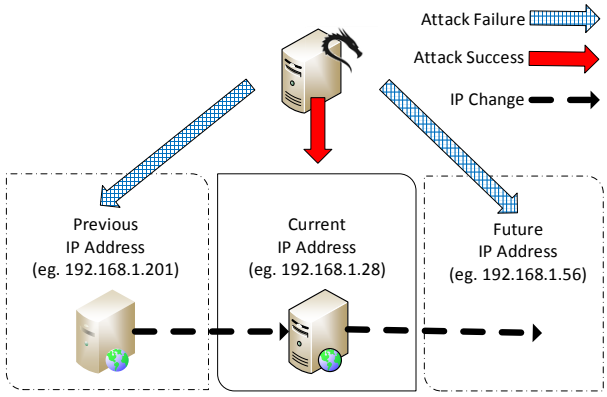

Figure 3: IP Randomization

3.2.3 Concurrent MTDs. The two MTDs must be capable of operating without interference. If the IP were to change while the service is being reconfigured, the service may not restart properly, resulting in an additional loss of service. This scenario becomes increasingly likely as the reconfiguration rates increase. To ensure MTD compatibility, the two MTDs are implemented in a multi-threaded application with mutual exclusion (mutex) locks around the reconfiguration operations. This prevents the two MTDs from reconfiguring at the same time. The implementation of the MTDs working in conjunction with each other is shown in Algorithm 1.

It should be noted that these MTD defenses, as implemented, may not be practical for all web applications. In particular, web applications which require a stable, persistent connection would likely be disrupted. However, there exists a broad use case for applications employing stateless frameworks, such as the popular Representational State Transfer (REST). Authors of previously proposed MTDs [14] have developed MTDs with similar operational restrictions.

4 QUANTITATIVE ANALYSIS

Based on our attacker and defender model, we can determine the attacker's success rate. Individually, the MTDs might be analyzed in a manner similar to [27], where an attacker is assumed to win if they are in control for a certain period of time before a reconfiguration occurs. In a similar manner, an attacker in our attack model wins if can complete the attack sequence to find a vulnerable node and compromise it before being interrupted by a reconfiguration.

We can also compute the availability of the system with MTD(s) in operation. This represents the momentary loss of service caused by changing services or IP addresses and affects the attacker's success rate as well as benign users.

Algorithm 1 Concurrent MTD Implementation

```
s ← Service reconfiguration rate
i ← IP reconfiguration rate

function ServiceThread()
    while (currentTime − startTime > duration) do
        waitTime ← random(s)          ▷ Exponentially distributed
        sleep(waitTime)
        lock(mutex)
        reconfigureService()
        unlock(mutex)
    end while
end function

function IPThread()
    while (currentTime − startTime > duration) do
        waitTime ← random(i)          ▷ Exponentially distributed
        sleep(waitTime)
        lock(mutex)
        reconfigureIP()
        unlock(mutex)
    end while
end function

function main()
    init(mutex)                       ▷ Mutex lock for threads
    if s > 0 then
        threadCreate(serviceThread)
    end if
    if i > 0 then
        threadCreate(IPThread)
    end if                            ▷ Create threads
    if s > 0 then
        threadJoin(serviceThread)
    end if
    if i > 0 then
        threadJoin(IPThread)
    end if                            ▷ re-join threads
end function
```

4.1 Definitions and Assumptions

Because we are working with attacker and defender models based on real-world constraints, we define the parameters of those models and summarize them here:

- T_a: The average time required to successfully complete an attack from start to finish,
- μ_S: The average interarrival time for service reconfigurations
- μ_{IP}: The average interarrival time for IP reconfigurations
- t_S: The average time required to complete a service reconfiguration
- t_{IP}: The average time required to complete an IP reconfiguration
- s: The total number of possible configuration states
- s_v: The number of vulnerable configuration states

T_a also includes reconnaissance steps required to perform an attack if they are also disrupted by the reconfiguration. However, some MTDs might not disrupt certain reconnaissance actions. μ_S and μ_{IP} are the parameters for interarrival times for reconfiguration, which are exponentially distributed. t_S includes the time required to stop the previous service, wait for all processes to shut down, start processes for the new service, and verify they are running. Likewise, t_{IP} includes the time required to bring the interface down and back up again with a new IP address. Because of the additional steps and requirement to connect externally to obtain the new IP address, verify it and ensure that the change is propagated to users, t_{IP} can be an order of magnitude larger then t_S. s represents the total number of possible states the system can be in between reconfigurations, with s_v being the number of those states that are vulnerable to an exploit.

4.2 Availability

We first determine the availability A, as this also affects attacker success rate. For MTDs operating individually, we can calculate the availability as the expected uptime per reconfiguration cycle (μ_S or μ_{IP}) divided by the total expected uptime plus expected downtime per reconfiguration cycle (t_S or t_{IP}, as appropriate), leading to the availability due to service reconfigurations:

$$A_S = \frac{\mu_S}{t_S + \mu_S} \tag{1}$$

and availability due to IP reconfigurations as:

$$A_{IP} = \frac{\mu_{IP}}{t_{IP} + \mu_{IP}} \tag{2}$$

Because the MTDs utilize mutual exclusion, the downtime from one MTD must occur during the uptime from the other MTD. Therefore, we can compute overall availability as the product of the two availability values:

$$A = A_S \cdot A_{IP} \tag{3}$$

4.3 Attacker Success Rate

Next, we can compute the attacker's success rate based on the expected attack time and reconfiguration rates. Our defender model assumes that in order for an attack to be successful, it must be uninterrupted by a reconfiguration. Because of the memoryless property of the exponential distribution, as long as the system is not currently undergoing a reconfiguration, the expected time before the next configuration is equal to the respective value of μ, regardless of when the last reconfiguration occurred. Using the exponential distribution with $\lambda = \frac{1}{\mu}$, we can determine the base probability that the attack is successful by finding the probability that the random variable $\mathcal{X}$ which represents the time before the next reconfiguration is greater than the time required to execute the attack, or $P(\mathcal{X} > T_a)$. Solving using the probability distribution for the exponential function :

$$
\begin{aligned}
ps_S &= P(x > T_a) \\
&= 1 - P(x \leq T_a) \\
&= 1 - (1 - F_x(T_a)) \\
&= e^{-\lambda_S T_a}
\end{aligned} \tag{4}
$$

Now that we have the base probability that no reconfiguration occurred, we have to adjust the probability based on the probability $\frac{s_v}{s}$ that it was already in a vulnerable state, where s_v is the number of vulnerable states and s is the total number of states. In our case of service reconfiguration with three different services, $\frac{s_v}{s} = \frac{1}{3}$. In the case of IP reconfiguration, each state is considered to be equally vulnerable. If different states share common weaknesses, it would be reflected in s_v.

Using this methodology to determine attacker success rate also assumes the service is already running at the time the attack begins. We have already established that there is an impact to availability by using MTDs, which we adjust for by multiplying the attacker's chance of success by the availability. This further reduces the attacker's success rate and gives us an unintended benefit from an otherwise undesirable side effect of MTDs. Our probability of attacker success for each MTD are thus:

$$
ps_S = e^{-\lambda_S T_a} \cdot \frac{s_v}{s} \cdot A_S \tag{5}
$$

$$
ps_{IP} = e^{-\lambda_{IP} T_a} \cdot A_{IP} \tag{6}
$$

When both MTDs are operating at the same time, we can use a similar method to determine the attacker's success rate. If no reconfigurations are currently taking place at the time the attack starts, the attacker's probability of success is equal to the probability that both random variables $\mathcal{X}_s$ and $\mathcal{X}_{IP}$ are greater than the attack time T_a, adjusted as earlier for number of vulnerable states and overall availability, or:

$$
\begin{aligned}
ps &= e^{-\lambda_S T_a} \cdot e^{-\lambda_{IP} T_a} \cdot \frac{s_v}{s} \cdot A_s \cdot A_{IP} \\
&= e^{-T_a(\lambda_S + \lambda_{IP})} \cdot \frac{s_v}{s} \cdot A
\end{aligned} \tag{7}
$$

If necessary, this method could also be further generalized for three or more MTDs.

5 EXPERIMENTAL FRAMEWORK

In this section, we detail the experimental framework used to demonstrate the analysis shown in Section 4. This includes a description of the testbed environment with specific hardware and software configurations.

5.1 Experimental Environment

Experimental testing was performed using the Center for Secure Information Systems (CSIS) testbed environment located at George Mason University (GMU). This testbed environment is managed using XenServer 7 to quickly deploy virtual machines (VMs) and associated networking services, such as DHCP, to allow the VMs to communicate.

For this paper, a target VM was developed with an OS image adapted from the freely-available Metasploitable and provided by Rapid7. This image is commonly used in security testing. The base VM was modified, enabling various web services for the simulated attacker to compromise.

The attacker is represented by a VM on the network which is adapted from the Rolling release version of Kali Linux, a Linux distribution developed specifically for security testing. Our attacker VM was deployed with Nmap 7.50 and version 4.14.28-dev of Metasploit, a widely available security penetration testing tool. Attacks are launched using the php_cgi_arg_injection Metasploit exploit. The exploit targets unpatched versions of the Apache 2.2.8 HTTP Server, which was originally released in 2008 [5]. The exploit php_cgi_arg_injection specifically targets CVE-2012-1823 [1], an argument injection vulnerability. The vulnerability was exploited in the wild in June of 2013.

For each set of configuration parameters, 500 independent trials were conducted, consisting of a complete attack from the Metasploit node. Another process emulated a legitimate user, consistently trying to connect to the web server to perform small stateless transactions to measure effect on availability.

6 EXPERIMENTAL RESULTS

Results from the experiments performed are presented in this section. In addition to measuring attacker's success rate and availability, we also measured the time T_a required from start to finish of the attack where the target must undergo no reconfigurations; the average time t_s required to reconfigure a service; and the average time t_{IP} required to reconfigure an IP address. These values are shown in Table 1 and are used to along with the methods in Section 4 to predict the attacker's success rate and availability and compare them to our collected results.

Variable	Observed Value (seconds)
T_a	28.01
t_S	0.635
t_{IP}	9.59

Table 1: Average Attack and Reconfiguration Times

6.1 Service Reconfiguration

As seen in Figure 4, adding service reconfigurations greatly decreased the attacker's chance of success. In the case where service is static, we assume a worst-case scenario where the only service is the vulnerable Apache service. Therefore, the largest decrease in attacker success rate came from the diversity introduced by the initial introduction of the MTD. However, as we increase the reconfiguration rate, the service randomization MTD was able to prevent more than the expected 33.3% of the attacks directed against it, and these values match with our predictions.

Service reconfiguration also reduced our availability slightly, as seen in Figure 5. We observe a decrease in availability of up to 3.3% compared to the static case at the highest reconfiguration rate.

6.2 IP Reconfiguration

As we expected, Figure 6 shows adding IP randomization also decreased the attacker's success rate, although not to the extent that

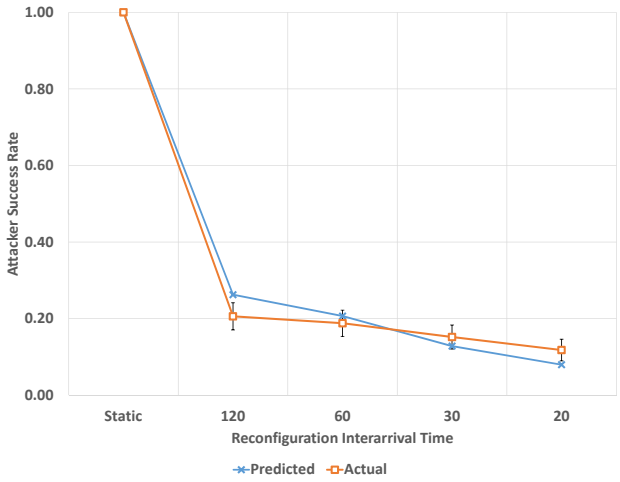

Figure 4: Probability of Attacker Success for Varying Service Reconfiguration Interarrival Rates

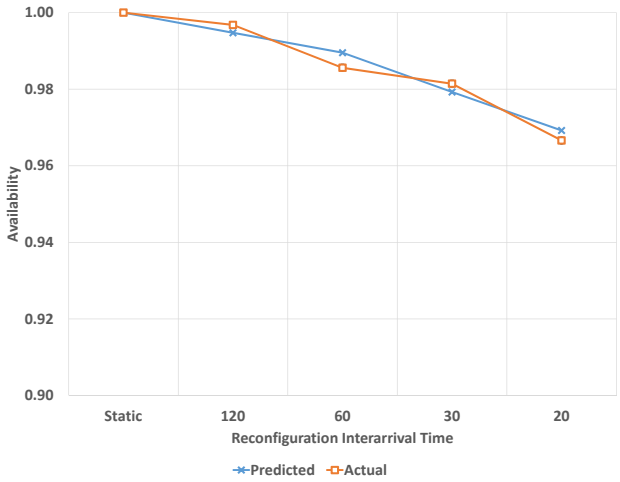

Figure 5: Availability for Varying Service Reconfiguration Interarrival Rates

the diversity-based service reconfigurations did, and the actual values follow the downward trend predicted.

Figure 7 shows the impact to availability when using IP reconfigurations. We observe a much larger decrease in availability when using the IP randomization scheme compared to service reconfiguration. This is due to the fact that as implemented, changing the IP address requires sending an external request for an IP address to the MTD controller and receiving a reply back. The connection to the monitor must also be rebuilt, which required a total of 9.59 seconds on average. This reduction in availability of 25% or more could mean that this method with higher reconfiguration rates may not be acceptable to users.

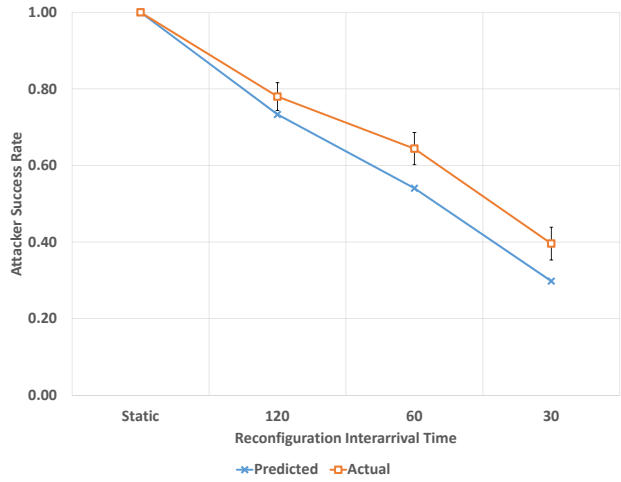

Figure 6: Probability of Attack Success for Varying IP Reconfiguration Interarrival Rates

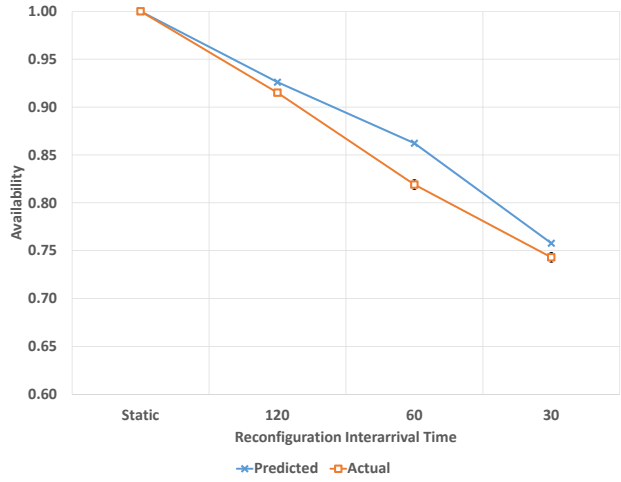

Figure 7: Availability for Varying IP Reconfiguration Interarrival Rates

6.3 Combined Effects

The effects on attacker success rate for each combination of settings for the two MTDs is shown in Table 2. As service and IP reconfiguration rates increase, the attacker's success rate tends to decrease monotonically.

Using the measurements obtained for T_a, t_S, and t_{IP} during the experiments, we compared the results to the values predicted by our analysis in Section 4, as seen in Table 3. The results are similar to the ones observed, with some larger errors present with lower interarrival times, but generally predict the behavior of our pair of MTDs.

Likewise, we measured service availability for each combination of settings and obtained similar results. As the interarrival time for IP reconfigurations decreases, the availability decreases.

		IP Reconfiguration			
		Static	120	60	30
Service	Static	1.000 ± 0.000	0.780 ± 0.036	0.644 ± 0.042	0.396 ± 0.043
	120	0.206 ± 0.035	0.102 ± 0.027	0.074 ± 0.023	0.054 ± 0.020
	60	0.188 ± 0.034	0.086 ± 0.025	0.078 ± 0.024	0.046 ± 0.018
	30	0.152 ± 0.031	0.086 ± 0.025	0.056 ± 0.020	0.044 ± 0.018
	20	0.118 ± 0.028	0.034 ± 0.016	0.034 ± 0.016	0.022 ± 0.013

Table 2: Attacker's Success Rate

		IP Reconfiguration			
		Static	120	60	30
Service	Static	1.000	0.733	0.541	0.298
	120	0.263	0.193	0.142	0.078
	60	0.207	0.152	0.112	0.062
	30	0.128	0.094	0.069	0.038
	20	0.080	0.058	0.043	0.024

Table 3: Attacker's Success Rate (Predicted Values)

However, as the service reconfiguration interarrival time decreases and IP reconfiguration is held at a some constant rate, we observe that the availability actually *increases* at some point. For example, when IP reconfiguration interarrival time = 60 sec, as service reconfiguration interarrival time goes from 60 seconds to 30 seconds, availability increases from 0.782 to 0.832. This may because the IP reconfigurations take so much longer relative to service reconfigurations and the two are mutually exclusive. This means that lengthy IP reconfigurations are delayed somewhat compared to service reconfigurations, resulting in the system behaving more similarly to that of service reconfiguration and that the two MTDs are still not wholly independent from one another.

		IP Reconfiguration			
		Static	120	60	30
Service	Static	1.000 ± 0.000	0.916 ± 0.003	0.819 ± 0.005	0.743 ± 0.005
	120	0.997 ± 0.000	0.909 ± 0.002	0.838 ± 0.003	0.692 ± 0.003
	60	0.986 ± 0.001	0.793 ± 0.003	0.782 ± 0.003	0.647 ± 0.003
	30	0.981 ± 0.001	0.692 ± 0.003	0.832 ± 0.003	0.711 ± 0.003
	20	0.967 ± 0.001	0.914 ± 0.002	0.794 ± 0.003	0.677 ± 0.003

Table 4: Availability

		IP Reconfiguration			
		Static	120	60	30
Service	Static	1.000	0.926	0.862	0.758
	120	0.995	0.921	0.858	0.754
	60	0.990	0.916	0.853	0.750
	30	0.979	0.907	0.844	0.742
	20	0.969	0.898	0.836	0.734

Table 5: Availability (Predicted Values)

6.4 MTD Protection Against Multiple Targets

The original set of experiments focused on a protecting a single target. However, most enterprises have multiple nodes that might require protection. This also matches more closely with a real-world example, as attackers probe a network and obtain a list of possible targets before attempting to probe further. With our virtual environment, we can create multiple instances of MTD-protected nodes and analyze those results.

We repeated the experiments using a total of six nodes in our virtual environment. Each node had identical MTD settings but ran independently. The attack script did a target scan against the entire network, followed by a deeper scan to determine service and an attack on vulnerable target. Scans and attacks were performed sequentially on each node found during the initial scan to model an attacker not opening connections to multiple targets at the same time to maintain a stealthy presence on the network, with a total of 100 trials performed for each combination of settings due to the increased number of targets.

We can observe the effect of service reconfiguration in Figure 8 which contains a series of histograms showing the number of times that a certain number of attacks were successful for multiple service reconfiguration settings. For these values, all IP addresses remained static.

For example, with an service reconfiguration interarrival time of 120 seconds, we observe a large number of trials where two or more out of six attacks were successful. However, as the interarrival time between service reconfigurations decreases, the histograms' distributions shift to the left. When the average interarrival time between reconfigurations is 20 seconds, the majority of trials resulted in zero or one out of the six available VMs compromised.

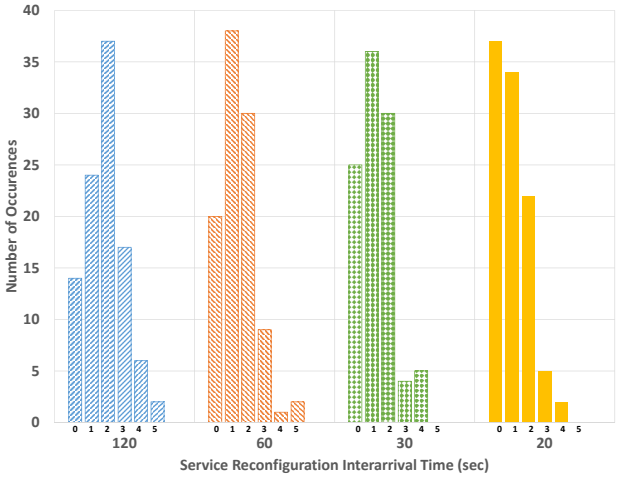

Figure 8: Histograms Showing Frequency of Numbers of Successful Attacks for Varying Service Reconfiguration Interarrival rates

With multiple nodes, we must also re-examine our criteria for success as the attacker or defender. We assume that an attack is considered a success if any nodes are able to be compromised. If attacks against each target are independent, the overall chance of attacker success $\hat{ps}$ would be 1, minus the probability that the individual attack with success rate ps on each of n different nodes all failed, or $\hat{ps} = 1 - (1 - ps)^n$. This constitutes an upper bound on attacker's success rate.

However, even more attacks may fail over time because the attacks are scripted to be performed in a sequential manner. By the time a likely target is probed further and exploited, the higher the likelihood of it being reconfigured already. This is illustrated in Figure 9.

ps for the single target case is the likelihood a reconfiguration takes place within time period T_a; However, due to the sequential

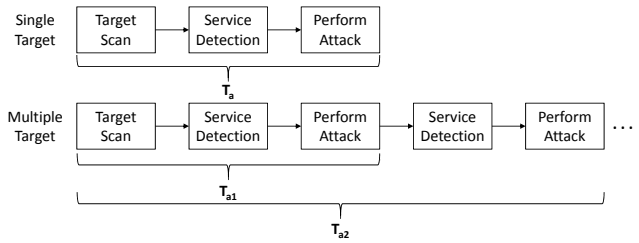

Figure 9: Timing Windows for Attacks on Multiple Targets

nature of the attacker probing and attempting attacks, the individual attack success probability ps_i against node i will vary due to the total time elapsed. Using the example in Figure 9 and Equation 7 we get:

$$ps_i = e^{-T_{ai}(\lambda_S+\lambda_{IP})} \cdot \frac{s_v}{s} \cdot A \tag{8}$$

As T_{ai} increases on each subsequent vulnerable node found, ps_i approaches zero, as the node must remain unchanged for a longer period of time for an attack to be successful. The overall success rate $\hat{ps}$ is then:

$$\hat{ps} = 1 - \left(\prod_{i=1}^{n} (1 - ps_i) \right) \tag{9}$$

which converges to a lower value than $1 - (1 - ps)^n$ as $ps_i \to 0$.

The complete observed results showing attacker's success rate are shown in Table 6.

		IP Reconfiguration			
		Static	120	60	30
Service	Static	1.00 ± 0.000	0.88 ± 0.064	0.61 ± 0.096	0.29 ± 0.093
	120	0.86 ± 0.068	0.64 ± 0.094	0.55 ± 0.098	0.22 ± 0.081
	60	0.80 ± 0.078	0.55 ± 0.098	0.43 ± 0.097	0.14 ± 0.068
	30	0.84 ± 0.072	0.55 ± 0.098	0.49 ± 0.098	0.15 ± 0.070
	20	0.63 ± 0.095	0.41 ± 0.096	0.29 ± 0.089	0.22 ± 0.081

Table 6: Attacker's Success Rate (Multiple Targets)

6.5 Computing Utility

The analytic model presented in Sections 4 allows one to predict the attacker's success rate and availability. We can then use these results to answer questions such as "given the user's objectives for security and availability, what combination of MTDs and settings maximize overall utility?"

We can solve this by assigning utility values to the attacker's likelihood of success and availability using the following sigmoid functions:

$$U_P(ps) = \frac{e^{\sigma(-ps+\beta_P)}}{1 + e^{\sigma(-ps+\beta_P)}} \tag{10}$$

$$U_A(A) = \frac{1}{1 + e^{\sigma(-A+\beta_A)}} \tag{11}$$

where A is the availability, β_A is the availability objective, ps is the attacker's probability of success, β_P is the attacker's success probability objective, and σ is a steepness parameter for the sigmoid. Two different forms of the sigmoid function are used because a solution with optimal utility seeks to minimize the attacker's chance of success and maximize availability.

Based on the utility values derived security and availability, we can now compute a global utility function U_G as:

$$U_G = w_P \cdot U_P(ps) + w_A \cdot U_A(A) \tag{12}$$

where w_P and w_A are weight factors chosen such that $w_P + w_A = 1$. Different values of w_P and w_A influence the optimal choice of MTDs and reconfiguration rates. Table 7 shows utility values of all settings for values of $\beta_P = 0.2$, $\beta_A = 0.99$, $\sigma = 10$, $w_P = 0.25$, and $w_A = 0.75$. These values correspond to both the defender security risk appetite and tolerance for service disruption. For example, a defender which favors availability would also favor more time between reconfigurations to limit service disruption (and promote higher availability) as opposed to preventing attacks. These weight factors are assumed to be known to - or other otherwise can be determined by - defenders via means outside the scope of this work. The values chosen represent this high availability use case, reflecting many service providers, which are contractually-obligated to maintain service uptime at levels exceeding 99%.

		IP Reconfiguration			
		Static	120	60	30
Service	Static	0.394	0.241	0.118	0.089
	120	0.509	0.413	0.329	0.239
	60	0.500	0.290	0.293	0.236
	30	0.513	0.344	0.330	0.250
	20	0.505	0.344	0.303	0.245

Table 7: Utility Values

Out of the settings evaluated, the optimal utility occurs with a service reconfiguration interarrival time is an average of 30 seconds and IP address reconfigurations is not employed at all. In our implementation, IP reconfiguration offered relatively little security benefit and relatively large loss of availability compared to the service reconfiguration technique. However, our results align with the conclusions of other researchers regarding network randomization [12], which also demonstrate relatively minor security benefits. Other pairings of MTDs might offer a more balanced result if their performance profiles were more comparable. In summary, a decision-maker presented with similar results may initiate efforts to implement service reconfiguration and also conclude that IP reconfiguration is not worth pursuing altogether; avoiding costly, but ineffective investments.

7 CONCLUSIONS AND FUTURE WORK

Individually, MTDs have demonstrated effectiveness against a variety of threats and attack vectors. Collectively, they offer potential of a more secure future. As one step in achieving that potential, this paper continues prior work in evaluation and performance modeling of MTDs and provides further contributions to the field. However, more research must be performed to fully realize the potential of MTDs.

The analytic work and implementation might be improved with a higher level of fidelity. More work could be done to understand

specific interactions between the attacker and defender within the implementation to improve its accuracy. For example, edge cases and race conditions may exist between the attacker and defender that cause a reconfiguration to be unsuccessful in preventing an attack.

Likewise, while the MTDs utilized would ideally be fully independent, our implementation does contain interactions between them. It is expected that other practical implementations would not be fully independent. Further work could involve development of a metric to measure the level of dependence between two or more implemented MTDs. Such a metric would allow decision makers to avoid combinations of MTDs that are highly dependent upon each other and may have undesirable interactions.

Conversely, the analysis and defender model can also be further generalized to apply to more MTDs. As different MTD techniques affect attackers in different phases of the Cyber Kill Chain, a MTD that prevents an attack earlier in the process might be more effective overall in preventing attacks. A MTD that takes effect later in the kill chain might instead provide defense by delaying the attack, ensuring the service remains protected long enough to accomplish its mission, or simply reducing the number of attempts an attacker is able to make against the system.

For example, while our attack model features an attacker which simulates nearly the entire Cyber Kill Chain, we may take an additional step. Our attacker established command and control on its target, but progressed no further. Realistic attackers leverage their control over compromised hosts over time to accomplish a range of objectives. These actions on objectives, such as data exfiltration, is already the focus of research [29]. Furthermore, there are known refresh MTD techniques [6] which may may be relatively ineffective at preventing initial compromise, but return compromised hosts to a safe state, theoretically mitigating the long-term impact of these attacks. Such an extension would likely require the development of additional attacker success metrics. As the goal of this research is determining the concurrent effectiveness of MTDs, evaluating the effectiveness of these varied MTD techniques, in combination, against the entire Cyber Kill Chain is a promising direction for future research.

8 ACKNOWLEDGEMENTS

The work presented in this paper was partially supported by Army Research Office grant W911NF-13-1-0421.

The authors would also like to express their gratitude to Sridhar Venkatesan for his contributions and insight.

REFERENCES

[1] 2012. CVE-2014-0160. Available from MITRE, CVE-ID CVE-2012-1823.. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1823

[2] 2017. *2017 Cost of Data Breach Study.* Technical Report. Ponemon Institute.

[3] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2012. Random host mutation for moving target defense. In *International Conference on Security and Privacy in Communication Systems*. Springer, 310–327.

[4] Massimiliano Albanese, Alessandra De Benedictis, Sushil Jajodia, and Kun Sun. 2013. A moving target defense mechanism for manets based on identity virtualization. In *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 278–286.

[5] Apache Software Foundation. [n. d.]. Apache. https://archive.apache.org/dist/httpd/

[6] Anantha K Bangalore and Arun K Sood. 2009. Securing web servers using self cleansing intrusion tolerance (SCIT). In *Dependability, 2009. DEPEND'09. Second International Conference on*. IEEE, 60–65.

[7] Stephen W. Boyd and Angelos D. Keromytis. [n. d.]. SQLrand: Preventing SQL Injection Attacks. In *Proceedings of the 2nd Conference on Applied Cryptography and Network Security (ACNS) (Lecture Notes in Computer Science)*, Vol. 3089. Springer, Yellow Mountain, China, 292–302.

[8] Gui-lin Cai, Bao-sheng Wang, Wei Hu, and Tian-zuo Wang. 2016. Moving target defense: state of the art and characteristics. *Frontiers of Information Technology & Electronic Engineering* 17, 11 (01 Nov 2016), 1122–1153. https://doi.org/10.1631/FITEE.1601321

[9] Thomas E Carroll, Michael Crouse, Errin W Fulp, and Kenneth S Berenhaut. 2014. Analysis of network address shuffling as a moving target defense. In *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 701–706.

[10] Warren Connell, Massimiliano Albanese, and Sridhar Venkatesan. 2017. A Framework for Moving Target Defense Quantification. In *ICT Systems Security and Privacy Protection*, Sabrina De Capitani di Vimercati and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 124–138.

[11] Executive Office of the President, National Science and Technology Council. 2011. Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity Research and Development Program. http://www.whitehouse.gov/.

[12] Katheryn A Farris and George Cybenko. 2015. Quantification of moving target cyber defenses. In *SPIE Defense+ Security*. International Society for Optics and Photonics, 94560L–94560L.

[13] Jin Bum Hong and Dong Seong Kim. 2016. Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing* 1 (2016), 1–1.

[14] Yih Huang and Anup K Ghosh. 2011. Introducing diversity and uncertainty to create moving attack surfaces for web services. In *Moving target defense*. Springer, 131–151.

[15] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. Amin. 2010. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Lockheed Martin Corporation.

[16] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. 2012. OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking. In *Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)*. ACM, Helsinki, Finland, 127–132.

[17] Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and Xiaoyang Sean Wang (Eds.). 2011. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats* (1st ed.). Advances in Information Security, Vol. 54. Springer.

[18] Quan Jia, Kun Sun, and Angelos Stavrou. 2013. MOTAG: Moving Target Defense Against Internet Denial of Service Attacks. In *Proceedings of the 22nd International Conference on Computer Communications and Networks (ICCCN 2013)*.

[19] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou, and Walter Powell. 2014. Catch me if you can: a cloud-enabled DDoS defense. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)*. 264–275.

[20] Hoda Maleki, Saeed Valizadeh, William Koch, Azer Bestavros, and Marten van Dijk. 2016. Markov modeling of moving target defense games. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 81–92.

[21] Pratyusa K. Manadhata and Jeannette M. Wing. 2011. An Attack Surface Metric. *IEEE Transactions on Software Engineering* 37, 3 (May 2011), 371–386.

[22] Daniel A. Menascé. 2003. Security Performance. *IEEE Internet Computing* 7, 3 (May/June 2003), 84–87.

[23] Neda Nasiriani, Yuquan Shan, George Kesidis, Daniel Fleck, and Angelos Stavrou. 2017. Changing proxy-server identities as a proactive moving-target defense against reconnaissance for DDoS attacks. *arXiv preprint arXiv:1712.01102* (2017).

[24] Hamed Okhravi, Thomas Hobson, David Bigelow, and William Streilein. 2014. Finding focus in the blur of moving-target techniques. *Security & Privacy, IEEE* 12, 2 (2014), 16–26.

[25] Hamed Okhravi, MA Rabe, TJ Mayberry, WG Leonard, TR Hobson, D Bigelow, and WW Streilein. 2013. *Survey of cyber moving target techniques.* Technical Report. DTIC Document.

[26] Hamed Okhravi, James Riordan, and Kevin Carter. 2014. Quantitative evaluation of dynamic platform techniques as a defensive mechanism. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 405–425.

[27] Hamed Okhravi, James Riordan, and Kevin Carter. 2014. Quantitative Evaluation of Dynamic Platform Techniques as a Defensive Mechanism. In *Research in Attacks, Intrusions and Defenses*, Angelos Stavrou, Herbert Bos, and Georgios Portokalidis (Eds.). Springer International Publishing, Cham, 405–425.

[28] Joshua Taylor, Kara Zaffarano, Ben Koller, Charlie Bancroft, and Jason Syversen. 2016. Automated effectiveness evaluation of moving target defenses: Metrics for missions and attacks. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 129–134.

[29] Sridhar Venkatesan, Massimiliano Albanese, George Cybenko, and Sushil Jajodia. 2016. A moving target defense approach to disrupting stealthy botnets. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 37–46.

[30] Jun Xu, Pinyao Guo, Mingyi Zhao, Robert F Erbacher, Minghui Zhu, and Peng Liu. 2014. Comparing different moving target defense techniques. In *Proceedings of the First ACM Workshop on Moving Target Defense*. ACM, 97–107.