Investigating the Application of Moving Target Defenses to Network Security

Rui Zhuang*, Su Zhang*, Alex Bardas*, Scott A. DeLoach*, Xinming Ou*, Anoop Singhal†
*Kansas State University
{zrui, zhangs84, bardasag, sdeloach, xou}@ksu.edu
†National Institute of Standards and Technology
{psinghal}@nist.gov

***Abstract*—This paper presents a preliminary design for a moving-target defense (MTD) for computer networks to combat an attacker's asymmetric advantage. The MTD system reasons over a set of abstract models that capture the network's configuration and its operational and security goals to select adaptations that maintain the operational integrity of the network. The paper examines both a simple (purely random) MTD system as well as an intelligent MTD system that uses attack indicators to augment adaptation selection. A set of simulation-based experiments show that such an MTD system may in fact be able to reduce an attacker's success likelihood. These results are a preliminary step towards understanding and quantifying the impact of MTDs on computer networks.**

***Keywords*-moving target, adaptive security, network security**

I. Introduction

In cyber space, time is on the attackers' side; they have time to study our networks to determine potential vulnerabilities and choose the time of attack to cause maximal impact. Then, once attackers acquire a privilege, they can keep that privilege for a long time without being detected [1], [2]. The static nature of current networks makes it easy to attack and breach a system and to maintain illegal access privileges for extended periods of time. To combat this advantage, a promising new approach to network security has been suggested called the moving target defense (MTD) [3]. While there are many facets of MTD, for computer networks, one can broadly interpret MTD as the fact that the network constantly changes its configuration to increase the difficulty of intrusion as well as maintain illegally acquired privileges for long. By changing a system's configuration proactively, it can also avoid a number of key obstacles that have made the traditional approaches less effective, such as zero-day vulnerabilities and false positive from intrusion detection systems (IDS). While promising, little research has been done to show that MTDs can work effectively in realistic networked systems.

There are a number of challenges to make an MTD system work for computer networks. First and foremost, while a network's configuration can be made more dynamic through constant changing, legitimate users still need to locate and access needed services. The design of an MTD system must ensure that even with the possibility of locating services through compromised components, the overall attack surface is still significantly reduced.

Typical security hardening measures focus on reducing the size of the attack surface. However, alternative ways to increase security include 1) increasing the "exploration surface" by moving the attack surface and 2) adapting the system's attack surface by changing system components. The larger the exploration surface and the smaller the attack surface, the more difficult it will be for an attacker to successfully penetrate a system. Both the frequency of these adaptation and which aspects of the systems can be modified determine the effectiveness of the moving target defense. Ideally a set of objective analytical models and metrics, parameterized by the above factors can be used to predict the effectiveness of MTD systems to protect computer networks. Such metrics would be difficult to calculate without a detailed design of the MTD system, since they are highly dependent on the runtime adaptation approaches used.

This paper presents a preliminary design for a moving-target defense system for computer networks. After presenting the preliminary design in Section II, we show simulation results of three different sets of experiments that show the efficacy of our design in Section III. We then discuss these results in Section IV followed by related work in Section V and our final conclusions in Section VI.

II. Moving Target Defense System Overview

The high-level architecture of our proposed MTD system that adapts in a purely randomly fashion is shown inside the

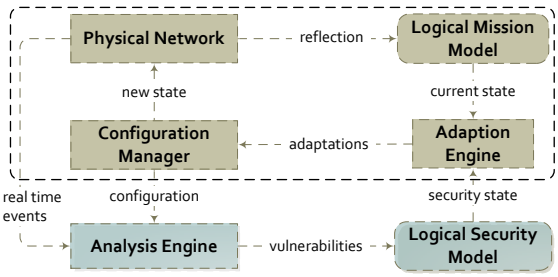

Figure 1. Moving Target Defense System Designs

dashed box in Figure 1. This system produces random adaptations that do not inhibit correct system operation. The key to making these random adaptations is that they are based on a *Logical Mission Model*, which captures an abstract view of the Physical Network's current configuration along with the functional requirements of the network. The driver is the *Adaption Engine*, which orders random adaptations to the network configuration at random intervals. These adaptations are implemented by the *Configuration Manager*, which controls the configuration of the *Physical Network*.

The architecture of an *intelligent* MTD system is the complete system shown in Figure 1. The basic operation of the random adaptation remains the same; however, we have added an *Analysis Engine* that takes real-time events from the Physical Network and the current configuration from the Configuration Manager to determine possible vulnerabilities and on-going attacks. The Adaptation Engine is extended to look at the network's current state along with its security state, as captured in the Logical Security Model. The *Logical Security Model* also consists of two runtime models: a goal model and a model of system vulnerabilities. The goal model captures the system's security goals while the vulnerability model is in the form of a novel *Conservative Attack Graph* (CAG), which captures both known and unknown system vulnerabilities and how an attacker might move through the system to gain specific privileges.

A. Resource Mapping System

A significant problem with a "moving" system is how to ensure that components in the system can locate other components they depend upon for functionality, given constant adaptations. We propose a *Resource Mapping System* (RMS) to solve this problem. An RMS is ideally implemented as a system communication enforcement component and knows the location of all the other components that this component depends upon. The RMS interacts with the Configuration Manager, which pushes the up-to-date location information of the various resources to the corresponding RMS's.

For clarity, we use the terms *roles* to refer to network services such as web servers, db servers, etc., and *resources* to refer to the physical system components such as a host with a concrete network address. Our initial design of the RMS's use in MTDs is shown in Figure 2. We assume each mission-critical role is executed on a unique virtual machine (VM), which has a dedicated RMS component to support communications with other roles.

The limitation of the RMS becomes evident if attackers compromise a critical role or VM. In this case, roles with which the compromised role initiates communications can be easily located and attacked since the compromised role's RMS knows their location. However, the attacker *must* follow the exact communication pattern defined by the Logical Mission Model; communication outside the predefined paths can be easily detected. In addition, adaptation

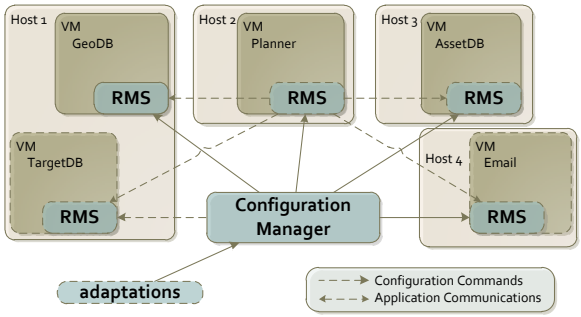

Figure 2. RMS System

can come to the rescue as, eventually, the VM of the compromised role will be refreshed and the attacker will lose any gained privilege.

B. Adaptation Engine

In traditional adaptive systems, the adaptation algorithm would attempt to provide optimal or near optimal configurations [4]. The objective of this component is to produce *effective* configurations that are significantly different in some aspect while limiting the costs of adaptation, or essentially maximizing the entropy of the configurations. Effective configurations must be functionally correct and consistent, as well as having an tolerable impact on network performance; the physical network and logical mission models are designed to allow the adaptation engine to predict these impacts based on the capabilities of the resources assigned to the system roles.

While the use of intelligent adaptations allows the MTD to react to suspected intrusions instead of simply adapting randomly, using intelligent adaptations in conjunction with purely random adaptations allows the MTD to effectively mitigate unpredicted attacks as well as mask the actions of the intelligent control system. While the use of random adaptations may not keep an attacker from learning all aspects of how the MTD system responds, we predict that it will make the learning process more difficult and time consuming. Additionally, by incorporating responses to suspected intrusions into adaptations, the system can react to suspected intrusions much sooner than a normal intrusion response system since even responses to false positives will leave the system in an operational state with no more overhead expended than for a random adaptation.

Since the Adaptation Engine is the main decision making apparatus for the MTD, it must be able to control the various modifiable aspects of the system such as the assignment of roles to resources, IP addresses and ports, firewall settings, applications (types, versions, etc.), VM types, and protocols between roles.

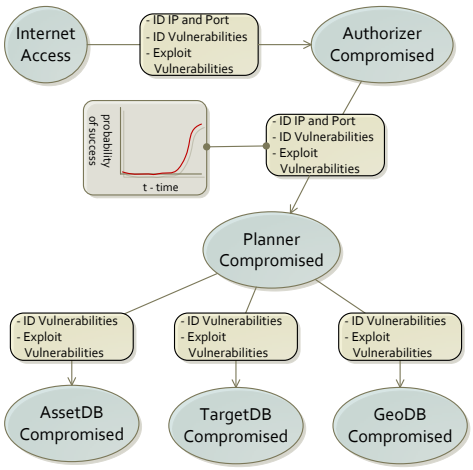

Figure 3. Conservative Attack Graph

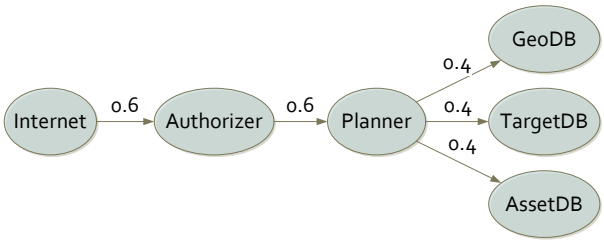

Figure 4. Simplified Conservative Attack Graph for Simulation

C. Analysis Engine

The purpose of the Analysis Engine is to infer the most critical vulnerabilities and most likely attack activities so the Adaptation Engine can make intelligent adaptation choices. The key output of the Analysis Engine is the CAG that captures known and unknown vulnerabilities and indicates paths the attacker might take in attacking the system.

To analyze the effect of an MTD on computer networks, we propose to use a *conservative attack graph* (CAG). Assuming unknown vulnerabilities in CAG actually reduces the size of the state model and makes it easier to apply stochastic analysis. Modeling an attacker both gaining and losing knowledge and privileges in CAG invalidates the typical monotonicity assumption [5] of most attack-graph work and requires a state-machine model, rather than traditional dependency attack graphs [6], [7], [8]. Previous state-enumeration attack graphs [9], [10] have encountered scalability challenges when applied to large networks [8]; however, these issues should be minimized due to the smaller size of a CAG.

As an example, Figure 3 shows the CAG for the mission planning system used in this paper. The topology of the conservative attack graph is partially derived from the logical mission model (not shown), where dependencies between roles are explicitly captured. In normal operation, valid users log in from the internet through the Authorizer node and interact with the Planner node. The Planner node interacts with the user by using data from the three database nodes, GeoDB, TargetDB, and AssetDB. The only legitimate access paths in the system are (1) from the Internet to the Authorizer, (2) from the Authorizer to the Planner and (3) from the Planner to the three database servers (AssetDB, TargetDB, and GeoDB). The conservative attack graph captures these logical access paths.

The conservative attack graph can also be viewed as a state-transition system. Each arrow is annotated with a label describing the activities involved to move from one state to the next. The effort involved in the activities can be measured in various ways. For example, one can ascribe a success-likelihood to time diagram to indicate how much time it will take the attacker to reach a certain success likelihood for a specific action.

III. Experiments

To determine if our approach has merit, we developed three high-level simulations to reflect the MTD approach discussed above. The first simulation, which we call the RMS-only Simulation, was developed to evaluate the effectiveness of our MTD approach using an existing network simulator called NeSSi2, an open-source, discrete-event based network security simulator with support for complex application-level scenarios based on a simulated TCP/IP protocol stack [11]. In this simulation, we assumed the attacker had full knowledge of the logical system configuration and only attacked through the systems RMS system. In the second and third simulations (which we term *broad attack simulations*), while the attacker still has full knowledge of the logical system configuration, the attacker also attempts attacks between nodes not directly connected via the RMS system. For these broad attack simulations, we developed a unique event-driven simulator based on the experience of building our first NeSSi2-based simulator. In the first two simulations, we assumed only a basic MTD system that adapted randomly at a specified time interval. However, in the last simulation, we upgraded the MTD to an intelligent MTD system that could detect when attacks were attempted outside the RMS system.

An overview of the simulated network is shown in the simplified CAG of Figure 4. The edges in the graph (with the exception of the Internet to Authorizer edge) show the valid paths supported by the RMS. In normal operation, valid users log in through the Authorizer node and interact with the Planner node. The Planner node interacts with the user by using data from the three database nodes, GeoDB, TargetDB, and AssetDB. We assume the attacker is located at the Internet node and wishes to attack the TargetDB. We made several assumptions to simplify our simulations.

1) Adaptations are applied at a specified time interval and are random in nature (which is extended in the third

simulation to include intelligent adaptation).
2) Adaptations are limited to VM refreshing.
3) All VMs assigned to play a given role have the same configuration except for its ID and IP address.
4) Once a node is compromised, the attacker can immediately use the RMS to attack the next node.
5) Attack is restricted to the VMs playing the five roles.
6) The attacker knows immediately when a VM it has compromised has been refreshed.

While these assumptions make the simulation easier, they are also tilted toward the attacker since we do not use advanced variability techniques (software versions, operating systems, etc.), which would make compromises more difficult, and we assume the attacker knows the system design and can immediately compromise the RMS.

A. RMS-only Attack Simulation

The three main components of the RMS-only testbed include the Defense component, the Attack component and the Ground Truth component. The *Defense component* contains the Configuration Manager, three physical resources (hosts) and five active VMs. These five VMs can be assigned to any hosts to play any of the five roles: Authorizer Planner, TargetDB, AssetDB, or GeoDB. The Configuration Manager is the core of the Defense component and combines the functionality of the Configuration Manager and the Adaptation Engine from Figure 1. At each adaptation time interval, the Configuration Manager selects an adaptation by creating a *new* task, $t_{new} = \{role, host, vmid, ip\}$, by (1) randomly picking a role, (2) randomly picking a host, (3) generating a new unique VM ID, and (4) randomly picking an unassigned IP address. The Configuration Manager finds the associated old task, $t_{old} = \{role, host', vmid', ip'\}$, within its set of existing tasks, T, by matching role names. It then informs the old task's current host, $host'$, to shut down the $vmid'$ VM and tells the new host, $host$, to start up a new VM at address ip to play the $role$. Finally, the Configuration Manager updates the Ground Truth component with the current configuration.

The *Attack component* simulates the attacker and uses the CAG shown in Figure 4 to allow it to know exactly where to attack to achieve its goal, the TargetDB. Since the only available attack path is to penetrate from Internet to the Authorizer, the Authorizer to the Planner, and then from Planner to TargetDB. The edge values in the CAG denote the attacker's probability of attack success between nodes assuming both nodes remain static. As shown, the attacker has a 40% chance of compromising the TargetDB if (1) it has already compromised the Planner and (2) the Configuration Manager does not adapt either the Planner or the TargetDB during the time step. In a real system, these probabilities would be based on the current probability of unknown and known vulnerabilities of the roles.

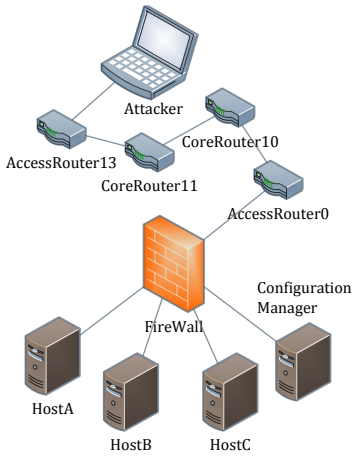

Figure 5. Network Topology

Each simulated attack has several steps. First the current CAG is retrieved from the Ground Truth component. Next, after waiting Δt time intervals (which simulates the time required to launch an attack), an updated version of the CAG is retrieved and used to determine whether the attack has succeeded or not. To determine attack success, we first generate a random value and check to see if it's less than the CAG edge value for the current attack. If it does, the simulation determines if the VMs on either the attacker's current node or the attacked node have been refreshed; if either of them has been refreshed, the attack fails. If the attacker's current node was the VM that was refreshed, the attacker is pushed back to its previous node. If neither were refreshed, the attack succeeds.

The *Ground Truth component* maintains the current CAG. The Ground Truth component receives adaptation information from Configuration Manager and updates the CAG as required. It also supplies the current CAG to the Attack component when requested. The Attack component, Defense component, and Ground Truth component are implemented as NeSSi2 components along with the three host resources: hostA, hostB, and hostC. These six components are loaded onto the corresponding nodes as shown in Figure 5.

1) RMS-only Attack Simulation Results: We conducted two different experiments (denoted 1a and 1b) to see how the frequency of system adaptation would impact attack success. Within each experiment, we included a control scenario where no adaptation occurred. Attacks were launched from the Internet towards the TargetDB. Each attack consisted of *single step attacks* from the Internet to the Authorizer, the Authorizer to the Planner, and from the Planner to the TargetDB. Once the TargetDB was compromised, the attack was counted as a successful. If a single step attack failed, the attacker remained at its the current VM and retried the attack until successful or the MTD system refreshed the VM. In each experiment, we performed 1000 single step attacks

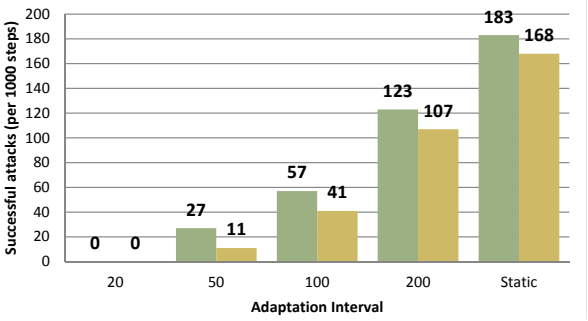

Figure 6. Attack Success Against TargetDB (experiment 1a are shown by green while experiment 1b are shown by gold bars)

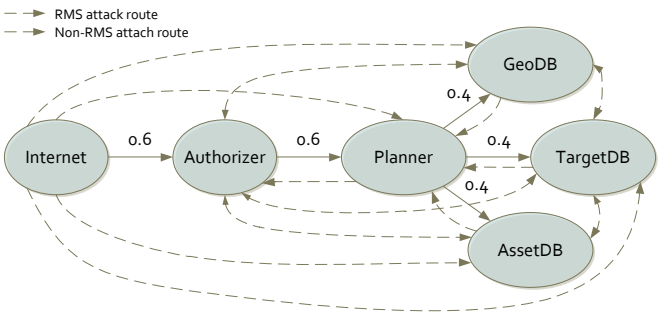

Figure 7. Attack Success Probabilities in Broad Attack Simulation (dashed lines have a probability of p/65,536 where p is the probability of successfully attacking the role through the RMS)

with a fixed Δt between each single step attack of 100 time intervals. We ran the 1000 single step attacks against an MTD system using 5 different time intervals (20, 50, 100, 200 and ∞) between each adaptation. Note that an ∞ adaptation interval corresponds to a static system.

In the experiment 1a, we assumed that in order to stop a single step attack from succeeding, either a randomly generated probability value is greater than the single step edge associated probability in CAG, or the MTD must refresh either the node under attack or the node from which the attack was launched during the attack (100 time intervals). Therefore, if there was a single step attack occurring from the Planner to the TargetDB, it could be stopped if either a randomly generated probability value is greater than 0.4, or the Planner, or TargetDB roles were refreshed by the MTD system during the attack. However, the attacker would remain on the network unless the actual VM it was residing on was refreshed. The green bars in Figure 6 shows the ability of the MTD to deter a successful attack from the Internet through the Authorizer and the Planner to the TargetDB. When the configuration is static, the number of successful attacks (of each round of 1000 single step attacks) is 183. Essentially, since no refreshing was going on, this is maximum number of successful attacks given the probabilities of single step attack success. Once the MTD system is activated, the number of successful attacks decreases. With an adaptation interval of 200, the number of successful attacks is reduced to 123, while an interval of 100 reduces it to 57, and an interval of 20 eliminates all successful attacks against the TargetDB. Figure 6 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is clearly visible.

In the experiment 1b, we assumed that in order to stop an attack from succeeding, the MTD could refresh any node on the path to the node being attacked during the attack (100 time intervals). Thus in this version, if there was an single step attack occurring from the Planner to the TargetDB, it could be stopped if either the Authorizer, Planner, or TargetDB roles were refreshed during the attack. The gold bars in Figure 6 shows the ability of the MTD to deter a completed attack from the Internet through the Authorizer and the Planner to the TargetDB. When the configuration is static, the number of completed attacks (out of 1000) is 168, while an adaptation interval of 200 reduces that number to 107, 100 reduces it to 41, and an adaptation interval of 20 again eliminates all successful attacks against the TargetDB. Again, Figure 6 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is obvious.

B. Broad Attack Simulation System

In the broad attack simulation, the TargetDB is again the attacker's goal. However, we assume a more aggressive attacker who automatically attacks each available node in the network from each compromised VM using either the RMS or by guessing an address and port of an available node. Thus, the attacker is not limited to the RMS routes and the attack routes form a completely bidirectionally connected graph (except for the Internet node) as shown in Figure 7. However, since we assume that the RMS is designed to *not* respond to standard requests for mapping information, this eliminates the attacker's ability to automatically map the address space.

The probabilities associated with each attack depend on the node from which the attack originates and the node being attacked. All attacks along the RMS maintain their probabilities as shown in Figure 7. However, the dashed lines, which denote attacks outside the RMS, have a much lower probability due to the fact that the attacker must guess the appropriate port for the attack to even have a chance to succeed. Therefore, each dashed line has an attack success probability of $p/65,536$ where p is the probability of successfully attacking that node through the RMS and 65536 is the port number space. Thus, all attacks against the TargetDB from any node but the Planner would have a $0.4/65,536$ probability of success. While this might seem like a very low probability, we believe that it is actually the upper bound for such an attack. Since the VMs addresses are being modified over time, the attacker will also have to

guess the VM address. However, since it is hard to determine the specific range over which the addresses be assigned, we assume the attacker can guess that in some way (again giving the benefit to the attacker as opposed to the MTD system).

The simulation starts with the attacker at the Internet node. From the Internet node, the attacker attempts to attack each node in the network. The success of each attack is determined based on the probability of success of the attack and whether either the node being attacked or the node from which the attack originated was refreshed during the attack. If any of the attacks were successful, the newly compromised nodes are used to mount new attacks. Again, we assume we try to attack all uncompromised nodes from each newly compromised node. This process continues until the TargetDB becomes compromised, or the attacker has no compromised nodes in the network (other than the Internet).

1) Broad Attack Simulation Results: We conducted 1000 runs (as opposed to 1000 single step attacks used in the RMS only experiments) of the broad attack simulation against various frequencies of MTD adaptation to determine its impact against attack success. Each run consisted of a sequence of attacks starting with the initial attack from the Internet to the Authorizer node and continuing until either (1) the attacker did not have access to a compromised node in the network or (2) the attacker successfully compromised the TargetDB. As with the previous experiments, we included a *static* control scenario where no adaptation occurred. In each experiment, we again assumed a fixed time interval $\Delta t = 100$ between each single step attack, and we ran the 1000 runs using 5 different adaptation intervals (20, 50, 100, 200 and ∞).

Figure 8 shows the ability of the MTD to deter an attack from the Internet through the network to the TargetDB. When the configuration is static, the number of completed attacks (out of 1000) is 588, which is close the expected 60% rate given that the probability of compromising the Authorizer node from the Internet is 0.6. This is due to the fact that if the attacker compromised the Authorizer node on the first attack, with a static network, the attacker will remain on the Authorizer node attacking various network nodes until the TargetDB is eventually compromised. We also noted that no attacks outside the RMS actually succeeded, which was expected given the extremely low probability of success. When we introduced our random adaptations, we found that an adaptation interval of 200 reduced the number of successful attacks against the TargetDB to 421, an adaptation interval of 100 reduced that number to 57, an adaptation interval of 50 allowed only 24 successful attacks, and an adaptation interval of 20 totally eliminated the ability of the attacker to compromise the TargetDB. Once again, Figure 8 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is clearly visible.

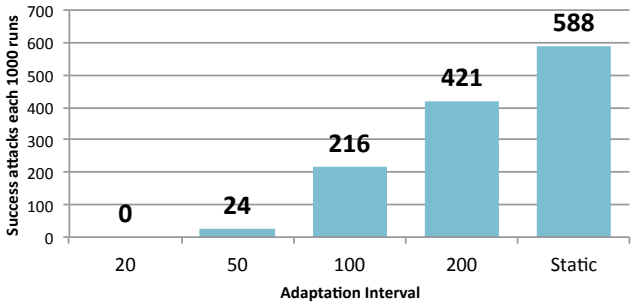

Figure 8. Attack Success Against TargetDB for Broad Attack Simulation Against Simple MTD

C. Intelligent MTD Simulation System

To help determine the effect of an intelligent MTD system, we again used our broad attack simulation where the attacker attempts to compromise the TargetDB. In fact, the experimental setup was the same as for the broad attack simulation presented above with one exception. To simulate an intelligent MTD system, we assumed that whenever the attacker attempted an attack outside the RMS, that such an attack could trigger an alert based on some probability of detection, p_d. Since the RMS is set up to allow only communication from known nodes on exactly one port, we believe the implementation of such detectors would be both practical and efficient. When detected, alerts would be sent directly to the Adaptation Engine, which would request that Configuration Manager immediately refresh the VM from which the detected attack originated. In addition, random adaptations continued to occur at the same predetermined intervals Δt as used in the previous experiments.

1) Intelligent MTD Simulation Results: The result of the intelligent MTD simulation is shown in Figure 9; note that the graph is logarithmic to show proper detail. Since the attacker indiscriminately attacks all nodes in the network without necessarily attempting to go through the RMS system, thus raising many alerts, the success rate of the attacker is reduced significantly. At a 100% probability of detection, the attacker is always immediately detected and removed from the system, thus the attack success rate is 0%. However, even with lower p_d values, the reduction in attack success is significant. Even in the static case, with a p_d of 50%, the number of successful attacks is reduced from 616 (61.2%) to 32 (3.2%). We believe this shows the power of using an RMS with an intelligent MTD system. The RMS minimizes the attack surface to such a degree that attacks outside the RMS are easily detected and significantly decreases the attacker's likelihood of success.

When compared to the attack success rate of the simple MTD (shown by the p_d = 0% line in Figure 9), the intelligent MTD performs significantly better. A slight data anomaly is evident at adaptation interval 20 when p_d is 15% and 25%; there is one successful attack while there are none

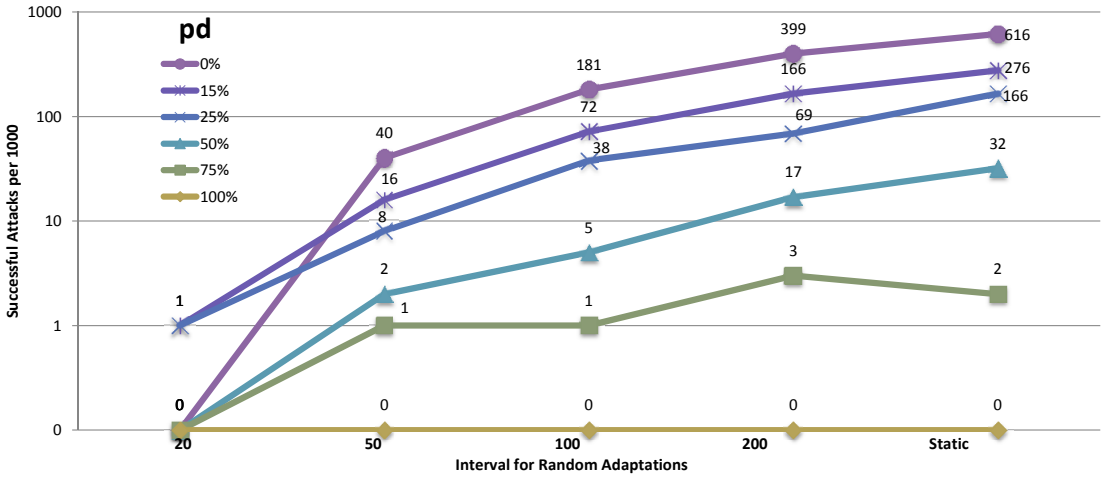

Figure 9. Attack Success Against TargetDB for Broad Attack Simulation Against Intelligent MTD

when $p_d = 0\%$. With more runs, we believe the data would normalized. This does show that while the probability is extremely low, the attacker can succeed. While not conclusive, this clearly shows the need for further research into the costs and benefits of intelligent MTD systems.

IV. Discussion and Future Work

This paper presents our initial results into understanding and quantifying the impact of MTD systems on computer networks. Our end goal is to develop objective analytical models to predict the effectiveness of MTD systems to protect networks, taking into account attacker models and the overhead of deploying the MTD system. The results presented are promising and warrant continued research.

To quantify the effect of MTD on the attack and exploration surfaces, we perform a simple analysis using only IP address and ports. In the static system where each VM instance performs a single function using a single port, the minimum attack surface size would be 5. With our MTD system, we add a host each for the Configuration Manager and Adaptation Engine and open an extra port on each VM for RMS and VM interactions. Thus, the attack surface increases to $(5 \times 2) + (2 \times 1) = 12$. If we assume our subnet has 256 addresses and each VM has 65,536 available ports, then the *maximum exploration surface size* is $256 \times 65,536 = 16,777,216$. The MTD's exploration surface size is equivalent to this maximum size. A static system's exploration area is much smaller as it uses standard ports and thus database services can be identified by probing around two dozen ports per VM; this reduces the *effective exploration surface size* to $256 \times 24 = 6,144$. Thus, the increase in the exploration surface for a MTD system would be 272,966%. Even with the increase in attack surface size, the MTD's vastly expanded exploration surface significantly increases an attacker's effort to find and attack the system.

A major issue that must be addressed is the cost of an MTD system. In our simulations, the biggest cost drivers were the cost of high quality detectors, in terms of p_d, and the random adaptation interval, Δt. There seems to be a trade off between these two costs as, even with a high Δt, the intelligent MTD was effective when $p_d > 50\%$. Other costs to be addressed include the overhead of actually adapting the system (e.g., refreshing VMs), automatic system configuration, and real time data collection of the system's operational and security states.

Another key issue is the design of the RMS system. We made several assumptions about how the RMS operates and what it will and will not allow. While no such system currently exists, there are possibilities such as the Self-shielding Dynamic Network Architecture [12], [13].

Finally, our results only show a *potential* for MTD to be effective in stopping attacks against computer networks. Obviously, simulation can only provide insights based on the assumptions made. Therefore, we are currently developing an MTD testbed using OpenStack [14] to show that our MTD design can actually work. Building an MTD system in a cloud infrastructure will be instrumental in demonstrating the effectiveness of MTDs for computer networks as well as validating analytical models.

Besides the security benefits discussed, using our MTD system in a cloud infrastructure may provide additional benefits. As the network expands, the system will know the exact configurations and service dependencies, which could significantly simplify life of the network administrator.

V. Related Work

Most work on network-based MTDs focuses on low-level techniques such as IP address shifting and network routing and topology control. In the late 90s, BBN developed approaches to active network defense [15] that gave the illusion that the addresses and port numbers used by the network's

computers changed dynamically. While these techniques significantly increased the attacker's effort by making it almost impossible to map the network [15], they required all trusted computers be shielded by special processes and had several application interoperability issues [16]. More recently, a network address space randomization scheme to thwart hit list worms [17], which configured DHCP servers to expire the leases of hosts at various intervals to support address randomization. In [18], an approach to dynamically changing network packet routes so that observable traffic patterns change was proposed to make network mapping more difficult and to make packet sniffing less effective.

More recently, Al-Shaer has created a moving target approach named Mutable Networks (MUTE) [19], which also works by re-assigning host IP addresses randomly. However, instead of mapping messages between hosts via an RMS, MUTE synchronizes the address knowledge within the network using cryptographic functions and secret keys.

VI. Conclusions

This paper presented a preliminary moving-target defense system design. We conducted several experiments to study the effects of randomly adapting one aspect of the system (role to VM mapping) in reducing attacker's success likelihood. Our results showed, a reduction in attack success as the rate of adaptation increased. In addition, we conducted simulations that showed the effect of using knowledge about when and where to adapt based on detecting attacks outside the RMS. Even with less than perfect detectors, significant improvements in network security can be made. The results clearly show the potential for simple and intelligent MTD systems and are first steps toward a comprehensive evaluation and analysis framework for MTD systems.

Acknowledgment. This work was supported by the Air Force Office of Scientific Research (FA9550-12-1-0106) and U.S. National Science Foundation (1038366, 1018703).

References

[1] Mandiant Intelligence Center, "APT1: Exposing one of China's cyber espionage units," Mandiant, Tech. Rep., 2013.

[2] D. Barrett, "Hackers penetrate NASDAQ computers," http://online.wsj.com/article/, Feb. 2011.

[3] NITRD, "National Cyber Leap Year Summit 2009 co-chairs' report, networking and information technology research and development," National Office for the Federal Networking and Information Technology Research and Development Program, Tech. Rep., Sep. 2009.

[4] C. Zhong and S. A. DeLoach., "An investigation of reorganization algorithms." in *the International Conference on Artificial Intelligence (IC-AI'2006).*, June 2006.

[5] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis." in *Proceedings of 9th ACM Conference on Computer and Communications Security*, nov 2002.

[6] S. Jajodia, S. Noel, and B. O'Berry., "Topological analysis of network attack vulnerability." *Managing Cyber Threats: Issues, Approaches and Challanges*, 2003.

[7] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R.K.Cunningham., "Evaluating and strengthening enterprise network security using attack graphs." MIT Lincoln Laboratory, Tech. Rep., 2005.

[8] X. Ou, W. F. Boyer, and M. A. McQueen., "A scalable approach to attack graph generation." in *13th ACM Conference on Computer and Communications Security*, Oct 2006.

[9] C. Phillips and L. P. Swiler, "Graph-based system for network-vulnerability analysis." in *NSPW 98: Proceedings of the 1998 Workshop on New Security Paradigms*, 1998.

[10] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs." in *the IEEE Symposium on Security and Privacy*, may 2002.

[11] S. Schmidt, R. Bye, J. Chinnow, K. Bsufka, A. Camtepe, and S. Albayrak, "Application-level simulation for network security," *SIMULATION*, vol. 86, pp. 311–330, 2010.

[12] J. Yackoski, P. Xie, H. Bullen, J. Li, and K. Sun, "A self-shielding dynamic network architecture," *Proceedings of IEEE MILCOM*, pp. 1381–1386, 2011.

[13] J. Yackoski, J. Li, S. A. DeLoach, X. Ou, and A. Singhal, "Mission-oriented moving target defense based on cryptographically strong network dynamics," in *CSIIRW '12: Eight Annual Workshop on Cyber Security and Information Intelligence Research*. New York, USA: ACM, 2013.

[14] Openstack, "Openstack: The folsom release," http://www.openstack.org/software/, july 1, 2013.

[15] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering,," *Proceedings of the DARPA Information Survivability Conference & Exposition*, pp. 176–185, 2001.

[16] J. Michalski, C. Price, E. Stanton, E. L. Chua, K. Seah, W. Y. Heng, and T. C. Pheng, "Final report for the network security mechanisms utilizing network address translation LDRD project," Sandia National Laboratories, Tech. Rep. Technical Report SAND2002-3613, November 2002.

[17] S. Antonatos, P. Akritidis, E. Markatos, and K. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Comput. Netw.*, vol. 51, no. 12, pp. 3471–3490, Aug. 2007.

[18] M. D. Compton, "Improving the quality of service and security of military networks with a network tasking order process," Ph.D. dissertation, Air Force Institute of Technology, 2009.

[19] E. Al-Shaer, "Toward network configuration randomization for moving target defense," in *Moving Target Defense*, ser. Advances in Information Security, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds. Springer New York, 2011, vol. 54, pp. 153–159.