Agent-Based Simulation in Support of Moving Target Cyber Defense Technology Development and Evaluation

Ben W. Priest, Era Vuksani, Neal Wagner, Brady Tello, Kevin M. Carter and William W. Streilein
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420, USA
{benjamin.priest, era, neal.wagner, brady.tello, kevin.carter, wws}@ll.mit.edu

ABSTRACT

Moving target (MT) technologies are a class of cyber security defensive techniques that seek to protect computer networks by making them less homogenous, less static, and less deterministic in order to increase the complexity required for a successful cyber attack. MT techniques are associated with performance costs, and thus their effectiveness and overhead must be evaluated before deployment in a live setting. However, testing the effectiveness and usage costs of a newly-developed MT technique on a live computer network is a costly process. This paper presents an agent-based approach for simulating an operational network and measuring the impact of security policies that incorporate MT technologies. The proposed agent-based simulation system (ABS) is intended to evaluate candidate MT techniques and provide important and cost-effective support for the overall MT technology development and testing process. We demonstrate the ABS model via a case study that evaluates a particular MT technology and discuss how this evaluation via simulation can be used to support the larger process of MT technology development and testing.

Author Keywords

Agent-based modeling; cyber security; moving target; security metrics; simulation

ACM Classification Keywords

I.6.3 SIMULATION AND MODELING: Applications; I.2.11 ARTIFICIAL INTELLIGENCE: Distributed Artificial Intelligence

INTRODUCTION

Practically all entities operating in both the public and private sectors, including banking, finance, public utilities, manufacturing, law enforcement, and medical services depend upon secure computer networks in order to continue to function properly. These agencies are under constant threat of cyber attack by malicious actors as evidenced by recently documented attacks against the International Monetary Fund, Lockheed Martin, Citibank, Google, and Sony [12].

Moving target (MT) techniques are a class of security technologies that seek to protect cyber systems by making them less homogenous, less static, and less deterministic in order to increase the complexity required for a successful cyber attack. However, these gains do not come for free, as MT techniques are associated with performance costs [25]. Thus, it is necessary to investigate the tradeoffs between security and performance of an MT technique before deploying it in a live setting.

This paper presents an agent-based approach for simulating an operational network and evaluating MT techniques for multiple relevant attack and defense scenarios with significantly reduced infrastructure, manpower, and time-costs.

Agent based models focus on the interaction of computational models of autonomous agents with each other and their environment. System behavior in such models is decentralized, where emergent phenomena are dependent solely upon low-level agent behaviors. Consequently, agent-based modeling and simulation is a preferred tool in the study of complex systems [5].

The proposed agent-based simulation (ABS) models a computer network with mission-critical and non-mission-critical users and traffic, cyber attacks sent by malicious actors, and an MT technique to be evaluated. The ABS model produces metrics that measure the effects of attacks and security policies on overall and mission-specific network operations and allows for security versus performance tradeoff analyses. We demonstrate the ABS model via a case study that evaluates a particular MT technology and discuss how this evaluation via simulation can be used to support the larger process of MT technology development and testing.

This paper is organized as follows. The first section provides a discussion of the related work in network security modeling and simulation. The second section discusses the benefits of the proposed simulation approach to the larger process of MT development and testing. The third and fourth sections proceed to describe the MT technology to be evaluated and the proposed ABS model, respectively. The fifth section describes the security and performance metrics generated by the

CNS'15, April 12-15, 2015, Alexandria, VA, USA

model and their interpretation. Finally, the last two sections demonstrate the model via a case study and provide an analysis and conclusions.

RELATED WORK

There exist numerous recent studies investigating various models appropriate for network intrusion detection. Some recent examples include [2, 3, 4, 7, 10, 11, 14, 13, 22, 24, 32, 31, 33, 34]. [4, 11, 13, 24]. While several of these studies employ simulation to explore the efficacy of proposed models for intrusion detection, the focus of these studies is on network situational awareness rather than network simulation.

[1] and [18] provide agent based simulation studies that are focused on investigative network modeling rather than network situational awareness. [1] employs agent-based simulation to analyze a modeled network with respect to a specific network security protcol in order to automatically determine protocol compliance. [18] provides an agent-based model that integrates the OMNet++ network simulation software package with network libraries INET Framework and ReaSE to investigate cooperative botnet attacks and corresponding defenses.

[8, 16, 19, 20, 35] provide non-agent simulation studies for investigative network modeling. [8, 16, 35] provide recent non-agent simulation studies for investigative network modeling. [8] applies a game theoretic model to simulate and assess network security in which a single attacker and defender are modeled. [16] combines discrete event simulation with metaheuristic optimization to simulate network attacks and optimize network defenses. [19] proffers a model built using OMNeT++ to simulate distributed denial-of-service attacks on wireless networks. [20] employs a Markov model to simulate worm attacks with simulation splitting techniques for efficient simulation of rare catastrophic network states. [35] provides an epidemic model to simulate malware propagation over network devices.

MT defense is a broad area of research that covers cyber defenses across many computing domains. MT defenses protect cyber systems by introducing diversity and dynamism, increasing the complexity of the attack surface and thereby thwarting attacks. MT defenses can be categorized into five categories: dynamic runtime environments, dynamic software, dynamic data, dynamic platforms, and dynamic networks [15].

In this paper we demonstrate the ABS model via a case study that evaluates a new MT technology that utilizes a dynamic software technique. Dynamic software is an MT defense that dynamically changes application code to thwart attacks that rely on the details of a specific binary [26]. Automated software diversity and multi-compilers are commonly used dynamic software techniques that introduce diversity in compiled binaries at compilation time to combat software monoculture. Two examples of such techniques are provided by [23] and [29]. In [23] a multi-compiler technique is used to distribute diverse binaries across a network to reduce the number of machines that a single attack can compromise while in [29] buffer overflows are mitigated by maintaining diverse binaries for the same application, with each binary casting a vote with respect to an incoming request.

CYBER MODELING AND SIMULATION

Comprehensive testing and evaluation of a new or developing MT technology requires examination of numerous cyber scenarios including scenarios with varying network environments, attacker threat levels, and defensive system configurations. Generally, there are four approaches to the evaluation of a candidate cyber defense technology: 1) analysis, 2) modeling and simulation, 3) emulation, and 4) live testing [21, 27, 28]. Each of these approaches has its strengths and weakness.

Live testing is important because it allows for evaluation of a defensive technology within the operational environment. However, executing comprehensive tests for relevant scenarios of attack and defense on a live computer network is, in general, cost prohibitive [9]. DARPA and others use large-scale network emulation experiments executed on cyber ranges to test new cyber defense technologies. These cyber ranges typically involve hundreds or thousands of machines and dozens of operators, and are more scalable than live tests [30]. However, network emulation is still quite costly and, in practical application, this means that a large proportion of relevant cyber scenarios are left unexamined before a technology is deployed. Analysis is important in the early stages of cyber defense technology development as it is necessary to formalize the model of the threat that the defensive technology is intended to mitigate. However, mathematical analysis of a complete cyber system with network environment, users, mission operations, attackers, and defenders is intractable. Modeling and simulation provides for tractable evaluation of a technology at the network scale with relatively low cost. Thus, a technology can be evaluated for a larger proportion of relevant attack and defense scenarios than it can be evaluated for via the other approaches. Ideally, all four above approaches should be applied for comprehensive evaluation of a candidate MT technology.

Modeling and simulation can be used at two stages in the evaluation of a new MT technology. Early in the technology development cycle, simulation can be used iteratively to test technology design decisions, try new designs, or make refinements to existing designs. Later, simulation can be used to cost-effectively investigate numerous relevant attack and defense scenarios to verify a MT technology's performance and define the smaller subset of scenarios that should be tested via emulation on a cyber range.

This work introduces a modeling and simulation method that may be used to evaluate MT technologies across a broad array of scenarios at a low cost. In practical application, such an approach will allow a wider examination of relevant scenarios. We focus on a particular developing technology in our analysis, which is described below.

MT MANAGEMENT TECHNIQUE

In this paper, we focus on a particular dynamic software MT (MCP) technique that uses a multi-compiler to dynamically

introduce software diversity to thwart attacks. MCP is a management system for maintaining software diversity that determines when the managed machine should generate a new binary for a critical application and swap it out for the current one. MCP is similar to multi-compiler techniques described in the literature, such as in [23] and [29]. However, it differs from the aforementioned studies by dynamically swapping between multiple different binaries for its managed application on a single device.

Because it is possible to detect an exfiltration of a known binary by searching outgoing communications for matching byte sequences, MCP intercepts outgoing communications and searches them for binary leaks. If one is found, MCP instructs the host to swap to a new binary. The specific costs of running MCP will depend upon how it is implemented and how it is configured for deployment. For this paper, we assume a particular implementation of MCP that is associated with a constant cost both for searching an outgoing communication and for swapping binaries once a leak is detected.

However, for even the most efficient search algorithm, searching all outgoing communications will also introduce significant overhead on the system. Therefore, searching every outgoing communication may not be advisable. This paper examines how to analyze the parameter space and develop plans for deploying MCP *before* investing in the process of implementing it at the network scale.

NETWORK MODEL

We use an agent-based model utilizing an abstract representation of device agents on a notional network. This model focuses only on internal device details that are relevant to MCP and attacks associated with such devices. Internal agents represent the network to be modeled, while the external agents represent the internet. There are three types of agents: *users*, *attackers*, and *defenders*. We will refer to the settings defining the users and attackers as the *scenario*. Additionally, we will refer to the settings defining the defenders as the *security policy* or simply *policy*. The purpose of the model is to evaluate a policy based upon the impact on the network for a given simulated scenario.

We separate the scenario into three components, the benign (S_B), mission (S_M), and attacker (S_A) scenarios. We assume a constant network latency, t_l, as a part of the scenario definition. Thus, the whole scenario can be described with the tuple

$$S = \langle S_B, S_M, S_A, t_l \rangle.$$

An experiment is described completely by a scenario-policy pair.

Agents operate in a discrete timing environment where actions take a certain number of time units to complete. Parameters for the scenario and policy that measure time are expressed in these time units.

Benign Scenario

Users are benign agents that model a particular machine that is connected to the internet and exist in both the internal and external networks. A user's primary function is to send, receive, and reply to communications, generating traffic.

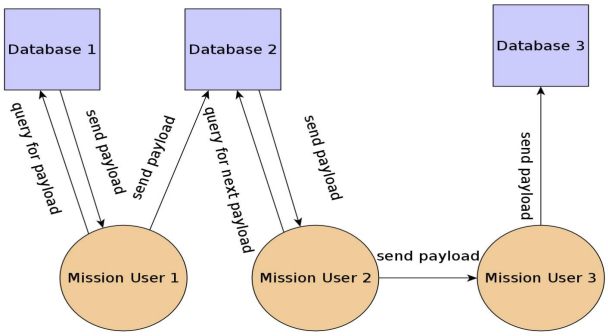

Figure 1. Mission layout

S_B will involve N_I internal users and N_E external users. These users issue new communications probabilistically following a Poisson process [17] with inter-arrival times drawn from an exponential distribution with a fixed parameter λ_u. Internal users respond to incoming benign communications with probability P_R. The benign scenario is described by the tuple

$$S_B = \langle N_I, N_E, \lambda_u, P_R \rangle.$$

Mission Scenario

In addition to background communication traffic, some internal users execute specific, time-sensitive *missions*. Any delay to the mission's completion is undesired. A mission team involves three internal users and three database servers that exist on different home network devices. We assume that each internal user can assume at most one mission role. We implemented the specific mission depicted in Figure 1, where the mission agents pass a payload from Database 1 to Database 3.

The network includes N_M mission teams, meaning that there are $6 * N_M$ mission devices. Mission users require a fixed amount of uninterrupted time, t_M, to operate on the payload before passing it to the next step. Normal operations, such as sending and receiving benign traffic can occur during this time, but suffering or mitigating a compromise will delay the mission.

The mission scenario is described by the tuple

$$S_M = \langle N_M, t_M \rangle.$$

Attacker Scenario

We assume that the attacker does not have previous knowledge regarding the home network, which leads to information-seeking campaigns. By sending initial exploit communications to a device, the attacker can gain insight into the software binary that exist on a home user's device. The attacker can then craft an attack against that binary. When an attack campaign succeeds, the target device is compromised, as outlined in Figure 2. We assume that these attacks are

intended to disrupt the mission, and thus are classed as availability attacks. Attacks that affect the confidentiality or the integrity of missions is an area of future work.

The attacker initiates new attack campaigns as a Poisson process with inter-arrival times sampled from an exponential distribution with parameter λ_a. Each campaign targets a randomly selected internal user. The attacker scenario is described by the tuple

$$S_A = \langle \lambda_a \rangle.$$

Defender Policy

A defender is a model of the security policy applied to an internal user. The defender has three capabilities, each of which has an associated time cost. Cleansing the user's device when it is compromised takes t_E time units. Inspecting outgoing communications for binary information leaks takes t_I. Swapping the user device's software binary when it detects a leak in an outgoing communication takes t_S. The latter two capabilities model MCP.

The MCP defense model is depicted in Figure 2. MCP intercepts outgoing communications and must make a decision based on its policy configuration to either inspect the communication for a leak or let it pass through uninspected. Recall that one purpose of this examination is to determine whether it is worthwhile to inspect all communications. If MCP decides to inspect and consequently detects a leak, it then generates a new version of the binary that replaces the user's current version. Attacks crafted against the old binary will now fail.

Defenders inspect an outgoing communication with probability P_I. Depending upon the implementation of the inspection capability, we assume that the defender inspection detects binary information leaked in an outgoing communication with probability P_D and has a probability of falsely detecting a binary leak in a benign communication, P_{FP}.

In a real system, P_D, P_{FP}, and t_I will not be independent: for any given algorithm looking for binary leaks, increasing P_D and decreasing P_{FP} will increase computational overhead (t_I). t_S on the other hand is the cost to create and swap the existing binary, which is independent of the search algorithm used for inspection. Also, t_E will be much greater than t_S, since the time to perform a binary swap will in general be much less than the time needed to restore a compromised machine to a safe state. t_I, t_S, and t_E must be set relative to t_l and t_M using data-driven methods that utilize ground truth data. We describe a policy, P, as a tuple

$$P = \langle P_I, P_D, P_{FP}, t_I, t_S, t_E \rangle.$$

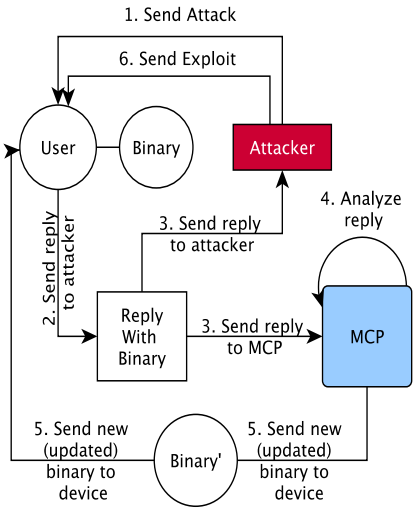

Figure 2. MCP layout

SIMULATION METRICS

Over the course of a simulation, users assume a number of different states, depending on their interactions with the network. We monitor these states over the course of the experiment to define metrics of the effectiveness of the security policy. We compare the effectiveness of a given set of defensive policies defending the same scenario by examining the tradeoffs between these metrics.

Throughput index

Each of the K communications in a simulation has an ideal lifespan of t_l, the communication latency of the network. However, if the instance of MCP defending the sender inspects a particular communication k, it will experience additional latency $\ell_I^{(k)}$. If k's sender or receiver is compromised during k's lifetime, it experiences latency due to attacker exploit $\ell_E^{(k)}$. If the defender of either host swaps binaries, then k experiences overhead due to the policy, $\ell_P^{(k)}$. We can normalize a throughput index $I_T(k)$ for a communication k by taking the complement of the total latency $(\ell_I^{(k)} + \ell_P^{(k)} + \ell_E^{(k)})$ with respect to the total lifespan of the communication $t^{(k)}$ and dividing by $t^{(k)}$. $I_T(k)$ is the ratio of uptime over the total lifetime of a communication k. This gives us an index where a value near 0 means that the communication is heavily delayed, a loss for the policy, while a value near 1 means that the communication is near ideal, a win for the policy.

$$I_T(k) = \frac{t^{(k)} - (\ell_I^{(k)} + \ell_P^{(k)} + \ell_E^{(k)})}{t^{(k)}}.$$

Performance indices

In a particular simulation run, the mission actors attempt to execute M missions. A particular mission, i, may or may not experience some amount of overhead over the course of the simulation. Recall from above that a mission i relies on six separate users, each of whom controls the mission payload for

some period of time. If any of these users are compromised while handling the payload, then i experiences overhead due to attacker exploit $d_E^{(i)}$. If any of these users swap binaries while handling the payload, then i experiences overhead due to the policy $d_P^{(i)}$. If MCP inspects any communications containing the payload, i experiences overhead due to inspection $d_I^{(i)}$.

We can normalize a performance metric $I_P(i)$ for a mission i in the same manner as above by taking the complement of the total downtime $(d_I^{(i)} + d_P^{(i)} + d_E^{(i)})$ with respect to the total runtime of the mission, $t^{(i)}$. We will also consider metrics formed by taking the complement of only the downtime due to the attacker $I_{P_A}(i)$, and only the downtime due to the defender $I_{P_D}(i)$.

$$I_P(i) = \frac{t^{(i)} - (d_I^{(i)} + d_P^{(i)} + d_E^{(i)})}{t^{(i)}}.$$

$$I_{P_A}(i) = \frac{t^{(i)} - d_E^{(i)}}{t^{(i)}}.$$

$$I_{P_D}(i) = \frac{t^{(i)} - (d_I^{(i)} + d_P^{(i)})}{t^{(i)}}.$$

EXPERIMENTS

In the previous sections, we described an agent-based modeling environment for evaluating security policies and defined metrics for evaluating the success of a security policy with respect to both attack mitigation and system overhead. In this section, we compare the results of experiments using different policy configurations and attacker scenarios.

We held most of the model parameters constant. The specific settings for the policy and scenario are given in the sections describing them. The only variable parameters are the probability of inspection, P_I, and the attack rate, λ_a. We will compare the different policy settings using P_I against one another in the different attacker scenarios specified by λ_a.

We matched up each value of P_I and λ_a. We simulated 1000 missions from start to finish for each policy-scenario pair. For each simulation, we collected the metrics described above. We use these metrics to compare the different policy settings against one another in different scenarios.

Policy Definition

We chose to consider a model of MCP implemented with a Bloom filter as its inspection algorithm. A Bloom filter is a probabilistic data structure that can be used to test whether an element is part of a set [6]. MCP uses a Bloom filter to decide whether a specific byte string is *definitely not* contained in a sample communication. This type of search does not allow the possibility of a false negative but does allow false positives. A particular configuration for a Bloom filter specifies the filter's false positive rate, carrying with it an associated search cost. Bloom filters configured for lower false positive rates have higher associated search costs relative to those configured for higher false positive rates.

Scenario	λ_a	Attack Traffic Density	Severity
S_{A_1}	100	$\sim 0.024\%$	minor
S_{A_2}	25	$\sim 0.094\%$	moderate
S_{A_3}	17	$\sim 0.13\%$	severe
S_{A_4}	10	$\sim 0.24\%$	critical

Table 1. Attacker Scenarios

MCP utilizes a Bloom filter to search outgoing communications and detect if they include any leaked information pertaining to a known application binary. Thus, P_D is set to 1 for all experiments. We used a Bloom filter configured with a false positive rate, P_{FP}, of 0.1 to calibrate t_I, the time to inspect. We simulated a Bloom filter with this setting in order to set a value for t_I relative to the other time-sensitive parameters in the model. All time-oriented parameters are defined relative to the cost to inspect, since it is the central object of investigation in the model.

We performed similar algorithm simulations to infer that the relative time cost of swapping binaries, t_S, is about an order of magnitude greater than t_I. We chose to set the time required to recover from an exploit $t_E = 100 * t_S$.

We performed experiments with a policy of $P = \langle P_I, 1, 0.1, 5, 50, 5000\rangle$. P_I determines how often the defender operates. We varied P_I = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.

Scenario definition

We executed all experiments using the same benign and mission scenarios. We varied the attacker scenario to evaluate the same policy's effectiveness against different levels of threat. We set all of the time-sensitive parameters relative to the parameters defining the policy.

We used benign scenario $S_B = \langle 250, 100, 10, 0.5\rangle$, which specifies 250 internal and 100 external benign users. The Poisson process governing new communications uses an exponential distribution with a parameter of 10 time units. These users respond to 50% of the benign communications that they receive.

We used mission scenario $S_M = \langle 40, 5000\rangle$ which specifies 40 mission groups operating in parallel. Mission users require 5000 time units to perform their work before they can pass the payload on to the next step. If a mission user is interrupted by an attack or defensive mitigation, it picks up where it left off once it has recovered to a safe state. Thus, if a user is repeatedly compromised or is continuously mitigating attacks, the mission can be delayed indefinitely.

Table 1 summarizes the attacker scenarios that we considered. The table displays the rate of attack (λ_a) for parameterizing the attacker. It also shows the resulting percentage of total traffic due to the attacker and a qualitative assessment of the scenario's severity.

We considered each of the four scenarios in Table 1. In each scenario, the network latency $t_\ell = 10$ time units.

Results

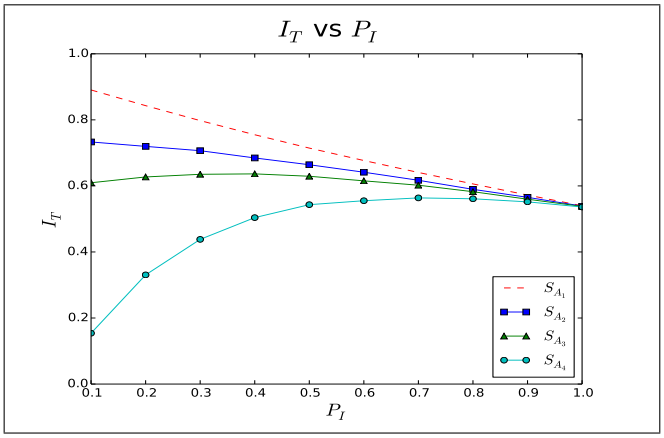

Figure 3. The mean throughput index $\overline{I_T}$ Versus P_I. As the probability of inspection increases, all scenarios converge to a stable value where all communications are inspected.

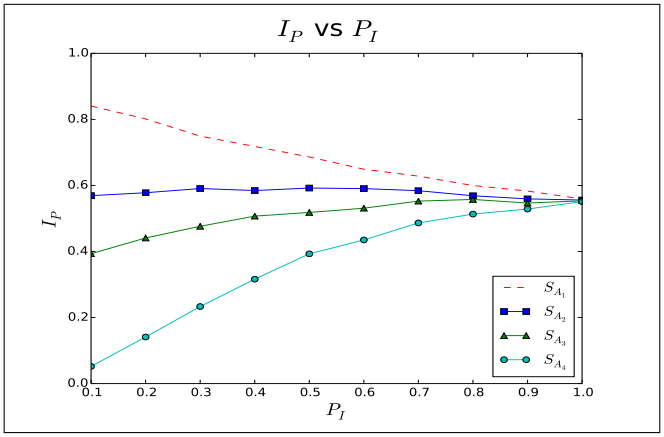

Figure 4. The mean performance index $\overline{I_P}$ Versus P_I.

We simulated 1000 missions for each experiment defined by a policy in Section and an attacker scenario in Section . We will define $\overline{I_*} = \sum_{i=1}^{N} \frac{I_*(i)}{N}$ as the mean of the index I_* for all of the N missions (or communications, if applicable) in a given experiment.

Figures 3 and 4 plot the Monte Carlo means $\overline{I_T}$ and $\overline{I_P}$, respectively, against P_I for each of the attacker scenarios. Notably, the curves for all of the scenarios converge as P_I increases.

This suggests that the throughput and performance overhead is roughly constant for P_I = 1, regardless of the rate of attack. At $P_I = 1$, all outgoing communications get inspected, incurring constant inspection overhead. As attacks are mitigated, there is no overhead from exploits other than the cost to swap binaries during disclosure campaigns. So, at P_1 all of the throughput and performance overhead comes from the defender. Table 1 indicates that even the most active attacker accounts for only a small portion of the total traffic in a simulation. Thus, the overhead from mitigating false positives dwarfs the overhead from mitigating true attacks.

This result suggests that it may be desirable to choose a lower P_{FP} so as to drive down the defender's overhead from false positives. However, as discussed in Section , driving down P_{FP} will drive up the cost of inspection t_I, resulting in an increased constant throughput overhead when $P_I = 1$. Finely tuning the relationship between P_{FP} and t_I and evaluating their effects in simulation will be explored in future work.

If all of the index curves in Figures 3 and 4 were to increase monotonically, it would suggest that $P_I = 1$ is the optimal setting regardless of the attacker scenario. However, curves in both figures invalidate this conclusion. In fact, increasing P_I results in monotonically decreasing $\overline{I_T}$ for S_{A_1} and S_{A_2} and decreasing $\overline{I_P}$ for S_{A_1}. This is because the low rate of incoming attacks results in less overhead than does the overhead induced by operating MCP. However, increasing P_I results in monotonically increasing $\overline{I_P}$ for S_{A_3} and S_{A_4}. Interestingly, for both of these scenarios the throughput gains from increasing P_I eventually level off and start to decrease, even though increasing P_I is always better for performance. For communication a k and a mission i, $I_T(k)$ is more sensitive to the increased P_I than $I_P(i)$, because the ideal lifespan of k is much less than that of i.

Interestingly, some of the curves in Figures 3 and 4 are not monotonic. As mentioned above, the $\overline{I_T}$ curves for S_{A_3} and S_{A_4} both improve initially, but eventually level off and decrease. Similarly, the $\overline{I_P}$ curves for S_{A_2} and S_{A_3} both reach a point in their arcs where increasing P_I actually degrades performance. Figure 5 provides a closer examination of this phenomenon. Figure 5 plots the Monte Carlo means $\overline{I_P}$, $\overline{I_{P_A}}$, and $\overline{I_{P_D}}$ against P_I for the moderate scenario S_{A_2}. As attacks are mitigated there is no overhead from exploits, so $\overline{I_{P_A}}$ approaches 1, which is ideal for the defender. However, the constant inspection overhead, successful mitigations, and false positives result in increasing defender overhead, encoded in the decreasing $\overline{I_{P_D}}$. In S_{A_2}, the aggregate performance index $\overline{I_P}$ levels out at $P_I = 0.3$ and starts to decrease after $P_I = 0.5$, where the $\overline{I_{P_A}}$ and $\overline{I_{P_D}}$ curves cross and the defender starts costing more in overhead than does the attacker. This tradeoff is not perfectly linear. The result is the non-monotonic $\overline{I_P}$ curve in the figure, which suggests that the optimal setting for P_I in this scenario is P_I, if the objective is to maximize mission availability.

While the other results are given in terms of Monte Carlo means, Figure 6 examines the distribution of $I_P(i)$ over all of the simulated missions i for the scenario S_{A_2}. Note that the plotted medians follow the same pattern observed above, gradually increasing until $P_I = 0.5$ and then gradually decreasing. Furthermore, the distributions exhibit lower variance as P_I increases. This is because random effects becomes less relevant as P_I increases. Even for missions that are relatively undefended, it is possible that due to random chance they may not be targeted during their execution. Additionally, missions with a less active defender are more likely than missions with more active defender to be compromised many times over the course of their execution. So, as P_I increases, the certain cost of defense increases while the uncertain cost

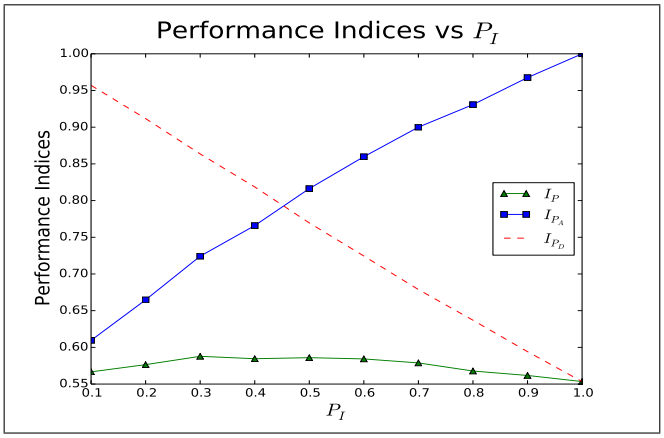

Figure 5. The mean performance indices $\overline{I_P}$, $\overline{I_{P_A}}$ and $\overline{I_{P_D}}$ plotted against P_I. Results shown use the moderate attacker scenario, S_{A_2}.

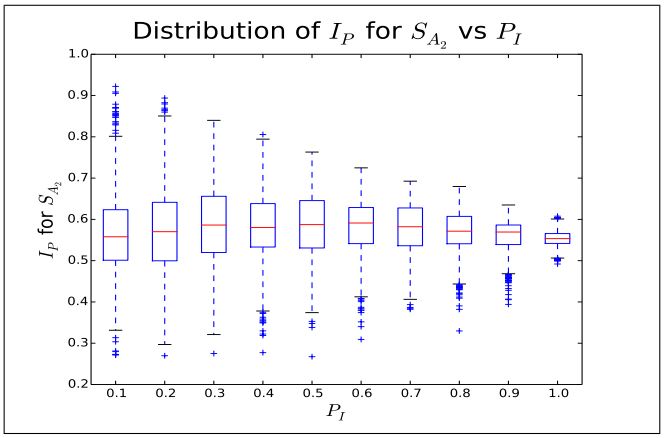

Figure 6. The distribution of $I_P(i)$ over all simulated missions i using attacker scenario S_{A_2}.

of attacks decreases, leading to the convergence of the box and whisker plots that we observe in the figure.

CONCLUSIONS

This paper presents an agent-based simulation model for evaluating MT techniques at the network scale. It is intended to augment the analysis and prototyping stages of the development of new MTD (Moving Target Defense) systems. The model captures interactions between attackers, MTD systems, and network operations and measures their effects on the security posture, performance, and throughput of the network.

The model is demonstrated via a case study in which an MT technology, MCP, is evaluated in a representative network environment for multiple policy configurations and attack scenarios. Experimental results show the relative tradeoffs between security and performance and throughput for the various policies investigated for different attack scenarios. These results demonstrate that policy decisions cannot be made without considering the scenario in which they will be deployed, as an effective policy for one scenario may not work for another. This case study illustrates the model's ability to provide decision support for answering analysis and early deployment questions. Moreover, it does so without incurring the large expense of deploying multiple candidate MT policies on a live network.

This paper only considers metrics surrounding the completion times of missions and of communications. These metrics only consider the availability aspect of information security. Thus, for some scenarios it is possible that the apparent optimal decision is actually not to use defenses at all. This is of course not necessarily practical. Many real-world scenarios place importance on the confidentiality and integrity of the mission. Planned future work for the system includes the development of attacker models that attempt to do more than just disrupt the mission's availability, as well as the development of metrics that capture the integrity and confidentiality states of missions.

Some additional planned future work for the system includes developing adaptive models for both the attacker and the defender. Other interesting areas of potential work include the consideration of stealthy attacker models as well as that of different implementations of the search algorithm for the purposes of comparison.

ACKNOWLEDGEMENTS

The authors would like to thank Hamed Okhravi and Michael Winterrose for their helpful insights.

This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

REFERENCES

1. Akbarzadeh, M., and Azgomi, M. Modeling and analysis of agent-based specifications of security protocols using csans and pdetool. In *2009 International Conference on Innovations in Information Technology (IIT)* (December 2009).
2. Ali, F., and Ismail, W. Network security threat assessment model based on fuzzy algorithm. In *IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (June 2011).
3. Anjana, V., and Bhuvaneswaran, R. Agent based cross layer intrusion detection system for manet. In *4th International Conference on Advances in Network Security and Applications, CNSA* (July 2011).
4. Arokia, J., and Hunmuganathan, K. Multi-agent-based anomaly intrusion detection. *Information Security Journal 20*, 4-5 (2011), 185–193.
5. Bar-Yam, Y. *Dynamics of complex systems*, vol. 213. Addison-Wesley Reading, MA, 1997.
6. Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13*, 7 (1970), 422–426.
7. Bonhomme, C., Feltus, C., and Khadraoui, D. A multi-agent based decision mechanism for incident

reaction in telecommunication network. In *ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2010* (May 2010).

8. Boyun, Z., et al. Network security situation assessment based on stochastic game model. In *Advanced Intelligent Computing. 7th International Conference, IOC 2011* (August 2011).
9. Breslau, L., et al. Advances in network simulation. *Computer 33*, 5 (2000), 59–67.
10. Caiming, L., et al. A distributed surveillance model for network security inspired by immunology. In *Artificial Intelligence and Computational Intelligence. Third International Conference (AICI 2011)* (2011).
11. Caiming, L., Yan, Z., and Run, C. Research on dynamic model for network security based on artificial immunity. *International Journal of Knowledge and Language Processing 2*, 3 (2011), 21–35.
12. Gjelten, T. For recent cyber attacks, motivations vary. National Public Radio, June 2011.
13. Jaisankar, N., and Kannan, A. A hybrid intelligent agent based intrusion detection system. *Journal of Computational Information Systems 7*, 8 (2011), 2608–2615.
14. Jaisankar, N., Saravanan, R., and Durai Swamy, K. An agent based security framework for protecting routing layer operations in manet. In *1st International Conference on Networks and Communications, NetCoM 2009* (December 2009).
15. Jajodia, S., Ghosh, A. K., Subrahmanian, V., Swarup, V., Wang, C., and Wang, X. S. *Moving Target Defense I & II*. Springer, 2013.
16. Kiesling, E., et al. Simulation-based optimization of information security controls: An adversary-centric approach. In *Proceedings of the 2013 Winter Simulation Conference* (December 2013).
17. Kingman, C., and Frank, J. *Poisson Processes*. Claredon Press, 1992.
18. Kotenko, I., Konovalov, A., and Shorov, A. Agent-based simulation of cooperative defence against botnets. *Concurrency and Computation: Practice and Experience 24*, 6 (2012), 573–588.
19. Malekzadeh, M., et al. Validating reliability of OMNeT++ in wireless networks dos attacks: Simulation vs. testbed. *International Journal of Network Security 13*, 1 (2011), 13–21.
20. Masi, D., et al. Simulating network cyber attacks using splitting techniques. In *2011 Winter Simulation Conference (WSC 2011)* (December 2011).
21. Nicholson, A., et al. SCADA security in the light of cyber-warfare. *Computers and Security 31* (2012), 418–436.
22. Nishanth, N., and Venkataram, P. Mobile agent based tcp attacker identification in manet using the traffic history (maith). In *IEEE 13th International Conference on Communication Technology (ICCT)* (September 2011).
23. O'Donnell, A. J., and Sethu, H. On achieving software diversity for improved network security using distributed coloring algorithms. In *Proceedings of the 11th ACM conference on Computer and communications security*, ACM (2004), 121–131.
24. Ohoussou, A., et al. Autonomous agent based intrusion detection in virtual computing environment. In *IEEE International Conference on Wireless Communications, Networking and Information Security, WCNIS 2010* (June 2010).
25. Okhravi, H., Comella, A., Robinson, E., and Haines, J. Creating a cyber moving target for critical infrastructure applications using platform diversity. *International Journal of Critical Infrastructure Protection 5*, 1 (2012), 30–39.
26. Okhravi, H., et al. Survey of cyber moving targets. Tech. Rep. 1166, MIT Lincoln Laboratory, September 2013.
27. Permann, M. R., and Rohde, K. Cyber assessement methods for SCADA security. In *15th Annual Joint ISA POWID/EPRI Controls and Instrumentation Conference* (Nashville, TN, June 2005).
28. Ralston, P., Graham, J., and Hieb, J. Cyber security risk assessment for SCADA and DCS networks. *ISA Transactions 46*, 4 (2007), 583–594.
29. Roeder, T., and Schneider, F. B. Proactive obfuscation. *ACM Transactions on Computer Systems (TOCS) 28*, 2 (2010), 4.
30. Rossey, L. M., Cunningham, R. K., Fried, D. J., Rabek, J. C., Lippmann, R. P., Haines, J. W., and Zissman, M. A. Lariat: Lincoln adaptable real-time information assurance testbed. In *Aerospace Conference Proceedings, 2002. IEEE*, vol. 6, IEEE (2002), 6–2671.
31. Sa, M., and Rath, A. A simple agent based model for detecting abnormal event patterns in distributed wireless sensor networks. In *International Conference on Communication, Computing and Security, ICCCS 2011* (February 2011).
32. Schafer, J., and Drozd, M. Detecting network attacks using behavioural models. In *6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011* (September 2011).
33. Sen, J. An agent-based intrusion detection system for local area networks. *International Journal of Communication Networks and Information Security 2*, 2 (2010), 128–140.
34. Stafrace, S., and Antonopoulos, N. Military tactics in agent-based sinkhole attack detection for wireless ad hoc networks. *Computer Communications 33*, 5 (2010), 619–638.
35. Toutonji, O., Yoo, S., and Park, M. Stability analysis of VEISV propagation modeling for network worm attack. *Applied Mathematical Modelling 36*, 6 (2012), 2751–2761.