Model-Driven Situational Awareness for Moving Target Defense

Ravi Jhawar and Sjouke Mauw
SnT, University of Luxembourg, Luxembourg
Email: {firstname.lastname}@uni.lu

Abstract: Moving Target Defense (MTD) presents dynamically changing attack surfaces and system configurations to attackers. This approach decreases the success probabilities of attacks and increases attacker's workload since she must continually re-assess, re-engineer and re-launch her attacks. Existing research has provided a number of MTD techniques but approaches for gaining situational awareness and deciding when/how to apply these techniques are not well studied. In this paper, we present a conceptual framework that closely integrates a set of models with the system and obtains up-to-date situational awareness following the OODA loop methodology. To realize the framework, as the first step, we propose a modelling approach that provides insights about the dynamics between potential attacks and defenses, impact of attacks and adaptations on the system, and the state of the system. Based on these models, we demonstrate techniques to quantitatively assess the effectiveness of MTD and show how to formulate decision-making problems.

Keywords: Adaptive cyber defense, Attack-Defense Trees, Dependency graphs, Security evaluation

1. Introduction

Security technologies that we use today are largely governed by cautious processes involving periodic penetration testing, security patch development and human-in-the-loop security events monitoring. The configuration of systems does not change for a long period and adaptations, if at all, happen only in deterministic ways to support maintenance and availability requirements. This provides an asymmetric advantage to the adversaries in terms of *time* – attackers can spend as much time as needed to perform reconnaissance of the target system and choose the best time to launch an attack. Furthermore, once an attack succeeds, attackers can maintain back doors and illegal privileges in the system for a long period since the system configuration remains static and identical.

To counter the advantages afforded to adversaries, a new class of adaptation techniques called moving target defense presents dynamically changing attack surfaces and system configurations to attackers. This approach complicates the adversary's mission in two ways. First, the attacker's workload increases significantly since she must continually re-assess, re-engineer and re-launch her attacks. Second, the success probabilities of her attacks decrease. MTD is currently considered as one of the game changing themes in cyber security [CITATION Exe \l 1033], particularly for protecting Internet services and critical infrastructures for both civilian and military applications.

A number of MTD techniques that increase the complexity of attacks by making the system less homogeneous, less static, and less deterministic have been proposed in the literature. For example, host-based MTD techniques that attempt to make seemingly random adaptations to the memory layout [CITATION VIy10 \l 1033] [CITATION CKi06 \l 1033], application's code [CITATION TRo10 \l 1033] and operating systems [CITATION MCh02 \l 1033][CITATION XJi07 \l 1033] have been proposed. Similarly, network-based MTD techniques that adapt network properties such as IP addresses, topology and protocols [CITATION SAn05 \l 1033][CITATION Eha11 \l 1033] have been developed. However, a majority of existing MTD techniques (e.g., IP address space randomization) were developed in isolation, to defeat specific attack steps (e.g., reconnaissance), by considering only certain aspects of the system configuration [CITATION HOk13 \l 1033]. Given its potential to be a game-changing technology, relatively recently, efforts have focused on studying MTD as a whole, instead of confining to individual techniques. In this direction, attempts have been made to define an MTD system using the concept of a configurable system [CITATION RZh14 \l 1033]. A few other works have focused on developing frameworks for quantitative evaluation of the effectiveness of MTD and/or to provide approaches for decision-making within an MTD system. We review such studies in detail and highlight research challenges that are constraining wide adoption of MTD in Section 2.

The contributions of this paper are twofold.

- A framework that outlines an approach to address the research challenges identified in Section 2. Our framework considers MTD in totality and represents its properties using a set of models. These models are used to measure the effectiveness of MTD and to formulate decision-making problems (Section 3).
- As the first step towards realizing our framework, we define the set of models that represent several properties of an MTD system: i) dynamics between potential attacks and defenses, ii) impact of attacks and adaptations on the system, and iii) state of the system and involved agents. We demonstrate how these models provide situational awareness and improve security assessment (Section 4).

2. State-of-the-art and research challenges

Figure 1 illustrates how MTD, in comparison with traditional system and adaptive hardening approaches, operates on a networked system and how it affects the attack surface. While traditional hardening aims at reducing the system's attack surface by removing unnecessary resources, MTD constantly shifts the attack surface and manifests its size, shape and frequency. This implies that the existing definitions of attack surface [CITATION PMa11 \t \l 1033] are not suitable for evaluating a *shifting attack surface* and the following remains an open research challenge.[1] **[RC1]** *There is a need to provide a definition of attack surface that can serve as a metric for characterizing MTD, formulate its semantics, and develop methods to measure it.*

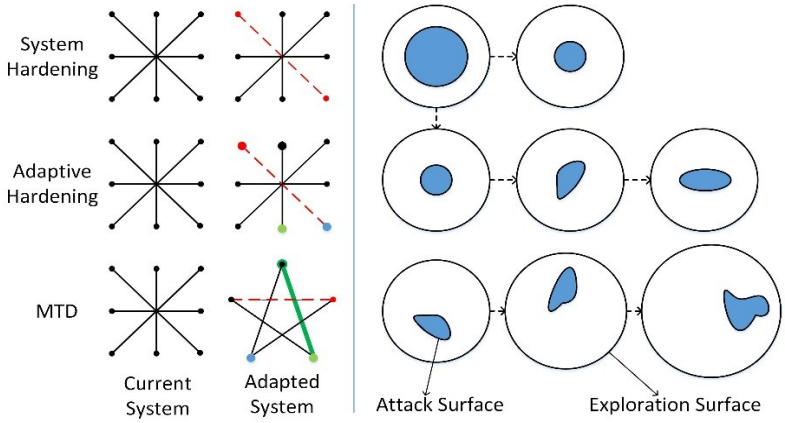

Figure 1 Impact of traditional hardening approaches and MTD on a networked system and attack surface

2.1 Measuring the effectiveness of MTD

We now review the two main approaches for measuring the effectiveness of MTD. The first approach to MTD evaluation consists in defining models to represent MTD techniques, and in providing methods to measure specific properties using those models [CITATION JXu14 \l 1033][CITATION Hon16 \l 1033]. However, using existing solutions, either the impact of both attacks and defenses on the system is not modelled or the state of the system cannot be derived. This situation gives rise to the following research challenge. **[RC2]** *There is a need to develop modelling schemes that can not only capture the dynamics between attacks and defenses but also the state of the system, involved agents, and interdependencies.*

The second approach to evaluate MTD consists in evaluating a set of system parameters using testbeds or simulations. Studies have focused on identifying the aspects that each MTD technique is intended to improve and measuring the technique's ability to realize that improvement e.g.,[CITATION PJD15 \l 1033]. In contrast, [CITATION KZa15 \l 1033] proposed a framework to evaluate MTD using testbeds, where the attacker's characteristics are mapped using cyber kill chains and metrics validated using simple mission models.

While the problem of assessing the effectiveness of MTD has been recognized as an important issue, existing solutions have considered only specific types of MTD techniques and have been validated using a single

1 Research challenges are enumerated for the sake of readability (RC1, RC2, RC3, RC4).

evaluation method. In this context, the following research challenge arises. **[RC3]** *A holistic framework with well-defined approaches to measure the impact of MTD on a system needs to be designed*.

2.2 Decision-making for MTD

A method to quantify the shift in attack surface and a stochastic game model to determine an optimal MTD strategy was introduced in [CITATION PMa13 \t \l 1033]. The goal of this work is to reduce the attack surface based on the trade-off between security and usability. However, the problems arising from the potential state space explosion and the instantiation of this model has not been studied yet. Albanese et al. define a model for evaluating and countering attacker's reconnaissance effort. The basic idea consists in controlling how an attacker perceives the attack surface of the target system. By doing so, attackers can be made to spend more time in identifying their target [CITATION MAl14 \l 1033]. Similarly, in [CITATION SJo15 \l 1033] a game model is used to evaluate and make decisions within an MTD system. In this case, the game's players compete with each other to control a shared resource.

The aspect of decision-making for MTD has been a central component of many recent frameworks. However, this aspect is still in a nascent stage and most frameworks are at a conceptual work-in-progress level. **[RC4]** *A research challenge consists in formulating the MTD decision-making problem and providing efficient methods to solve them*.

3. Overview of our framework

In this section, we present a conceptual framework that outlines the approach we adopt to address the research challenges (RC1—RC4) highlighted in the previous section. Our framework considers the MTD system in totality and models its properties using a set of formal security models. These models are kept up-to-date with the system and are used to measure the effectiveness of MTD techniques i.e., to perform quantitative security evaluation. Security models and evaluation results in turn allow us to formulate MTD decision-making problems and to answer questions such as: Which combination of MTD techniques satisfies given security goals? How often and when should one adapt the system? In the following, we discuss each component of our framework and summarize its operational details.

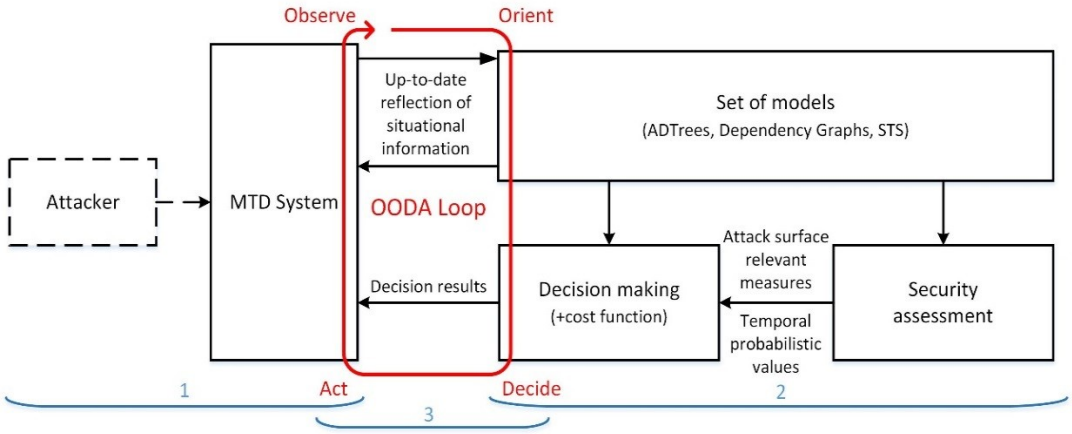

Figure 2 Overview of our framework and interaction between various components

MTD System and attacker model:[2] In general, our approach applies to any system that can implement MTD techniques. However, in this study, we consider a typical enterprise network that comprises servers and workstations and supports classic business applications such as emails, web services and databases as our system of interest. On the one hand, solutions developed for such general-purpose enterprise networks can be customized easily for other networked systems, and on the other hand, enterprise networks are increasingly hosting critical applications and are becoming part of large-scale attacks. For example, Stuxnet malware that targeted supervisory control and data acquisition (SCADA) systems, inflicting massive consequences on different levels including national scale disasters, first compromised the enterprise corporate network and used the conditions created in order to penetrate into the SCADA system [CITATION SKr12 \l 1033].

2 Components Attacker and MTD System – Part 1 – of Figure 2.

Enterprise networks typically comprise subnets delimited by firewalls, where each subnet supports a specific business unit, and the network is supported by security measures such as an intrusion detection system and access control. We consider a threat model where an attacker characterizes a standard kill chain (following steps: *reconnaissance*, *access*, *exploit development*, *attack launch and persistence*) [CITATION EHu11 \l 1033] [CITATION HOk13 \l 1033], whose goal is to breach security measures in the network. To defend against such attacks, we assume that the system administrator, in addition to the traditional defenses, has deployed the following MTD techniques that can together counter all stages of a kill chain.

- ***Network Address Space Randomization****:* This technique can be implemented in two modes. In the first mode, it improves the agility of the network gateway by frequently changing externally visible network addresses. This technique is effective against *reconnaissance* and *access* steps of the kill chain w.r.t. a remote adversary. In the second mode, it slows the effect of network scanning (as a consequence, the effect of IP address hitlist-based worms) by assigning short address leases to the hosts in the network and by changing the addresses when the lease expires.
- ***Dynamic runtime environment*:** To counter the *exploit development* and the *attack launch* steps of the kill chain, this technique prevents injection attacks on generic hosts and database servers as follows.
    - ***Address space layout permutation*:** This technique prevents the programs running on the hosts from code injection by randomizing individual programs independently.
    - ***Randomized SQL*:** This technique prevents buffer overflow attacks based on SQL injection by randomizing the query language so that any injected code is not executed. This technique is particularly useful when the query depends partially on untrusted inputs.
- ***Lightweight portable security:*** This technique uses a bootable OS that resides on a read-only medium to enable rapid recovery of hosts by ensuring that the OS boots into a clean and known good state. Such recovery removes existing infections and helps mitigate *persistence* threats on a system.
- ***Dynamic policies:*** To mitigate exploitation of trust conditions, this technique allows applications to change security policies in response to attempted intrusions, while maximizing the utility of machines in the network. Security policies can be changed over the entire network or on individual machines.

The techniques listed above are only examples of how MTDs can protect the system against a class of attackers. Our approach remains applicable even if a subset of the above techniques is considered or if new defenses introduced.

Models, security assessment, and decision-making:[3] Existing solutions have primarily focused on developing MTD techniques and on assessing and decision-making either using simulation methods or ad hoc models. In contrast, our approach consists in defining a set of well-integrated formal security models that not only represent the dynamics between potential attacks and defenses and their impact on the system, but also the state of the system and involved agents. This set of models provides basic situational awareness of the system. We discuss our modeling approach in detail in Section 4.

We use this set of models to assess the effectiveness of MTD. The *security assessment* component of our framework, for instance, complements Attack—Defense Trees with dependency graphs to measure the impact of each attack on the system and with stochastic techniques to measure temporal-probabilistic metrics. We note that assessing key temporal aspects is important because one of the goals of MTD is to eliminate attacker's asymmetric advantage of time. As part of the future work, we will provide new formalism of attack surface based metrics (e.g., shift in attack surface) and develop a model-driven approach to measure these metrics.

Finally, before discussing about MTD decision-making, we need to take into account the notion of overhead costs. MTD techniques introduce additional costs while defending against attacks. The term cost denotes administrative costs, service downtimes or performance costs. For instance, MTD that changes virtual machine images to implement platform diversity is useful against exploit development and persistence but likely disrupts running services, introducing downtime. In this direction, our goal is to formulate the cost function that measures the usefulness of MTD against introduced overhead costs.

Using the cost function, the effectiveness measures and the set of models, our framework formulates MTD decision-making problem as a i) multi-objective constrained optimization problem and ii) game-theoretic

---

3[5] Second block of Figure 2.

problem. An example of the first category of a decision-making problem is as follows. Given a set of available MTD techniques and the state of the system, identify the course of actions that minimizes the attack surface of a mission, while satisfying the functional and security requirements provided as constraints. This approach is effective if the attack surface remains constant during an attack and if the system remains static. However, since these conditions may not be satisfied, one must assume continuous stream of attacks that seek to disrupt system operations and act against defender's goals. Therefore, our framework extends the work in [CITATION RJh161 \t \l 1033] and models the decision-making problem as a game between the attacker and the defender with competing objectives. The cost function provides the pay-off values to each player and the equilibrium provides the decision results. Solvers such as GTE [CITATION Ege15 \l 1033] are used to obtain game's equilibrium and model different types of knowledge states that each player possesses.

Operational details: Figure 2 illustrates how the components of our framework interact with the MTD system. Given that the system is highly dynamic, we adopt the OODA loop methodology as follows:

- ***Observe*** – collate information from the system (e.g., by means of continuous monitoring, IDS and honeypots) about system incidents, ongoing attacks and adaptations.
- ***Orient*** – arrange the collated information by updating the set of models, say, using incremental generation methods. This step provides models that are in sync with the system. We note that this step is essential since the system is expected to experience frequent adaptations.
- ***Decide*** – quantitatively measure the security of the new configuration and the cost values. Apply decision-making algorithms in concurrence with the updated models and measures.
- ***Act*** – Perform what-if analysis to check if the system will be in a valid state if the decision results (e.g., an adaptation solution) are enforced. If true, implement appropriate actions (e.g., change in configuration) on to the system.

The loop reverts to the observe step after act and continues similarly thereafter.

The framework presented in this section outlines our high-level approach to address the research challenges in this domain. The most critical aspect of our approach consists in building a set of models that provide basic situational awareness. These models then serve as the basis for security assessment and decision-making. In the rest of this paper, we therefore focus on our modeling methodology and briefly discuss its relationship with security assessment. Formulation of the MTD decision-making problem and in depth semantics of MTD effectiveness measurement is out of the scope of this paper due to space restriction.

4. Modelling MTD-based security scenarios

In this section, we describe our methodology for modeling an MTD system. We start by representing attacker's behavior and potential defender's options.

4.1 Attack—Defense Trees

Attack—Defense Trees (ADTrees) provide a formal yet intuitive approach to systematically represent potential attacks and countermeasures in a system. ADTrees improve the widely used attack trees formalism, by including not only the actions of an attacker, but also possible counteractions of a defender. The root node in an ADTree represents the attacker's (or defender's) goal and the children of a given node represents its refinement into sub-goals. Each node can have one child of the opposite type, representing the node's counteraction, which can be refined and countered again. The leaves of an ADTree represent the basic actions of an agent, which need not be refined any further [CITATION Kor14 \l 1033].

Example 1: The Stuxnet attack can be distinguished into two main phases: i) infiltration and propagation into the corporate enterprise network and ii) compromising SCADA systems and industrial sabotage. The ADTree in Figure 3 shows how an attacker can achieve the first phase of Stuxnet. For the sake of simplicity, the second phase of the attack is not shown in this example. The triangle on top of the ADTree's root node indicates that the part of the tree (above) modelling attack on SCADA system has been abstracted. To compromise the enterprise network, the attacker must first gain privileges on the hosts in the network, then inject its payload into other legitimate processes (attack launch), and finally propagate the malware in the network in order to infect as many workstations as possible (this maximizes her chances to transit later to the control network). To gain privileges on the hosts, the attacker must perform reconnaissance of the network to identify potential vulnerabilities, develop specific exploits (e.g., several Windows and 0-day exploits are known to be used), and

establish trust conditions on her target hosts. She can then use these conditions to launch her attack by either performing code or control injection on the hosts and database servers. Finally, the attacker can propagate the malware either via LAN or by sending email attachments. The worm can use network shares or printers to propagate via LAN.

If the system administrator deploys the MTD techniques discussed in Section 3, the attacks on the enterprise network can be countered as shown in the ADTree (Figure 3).

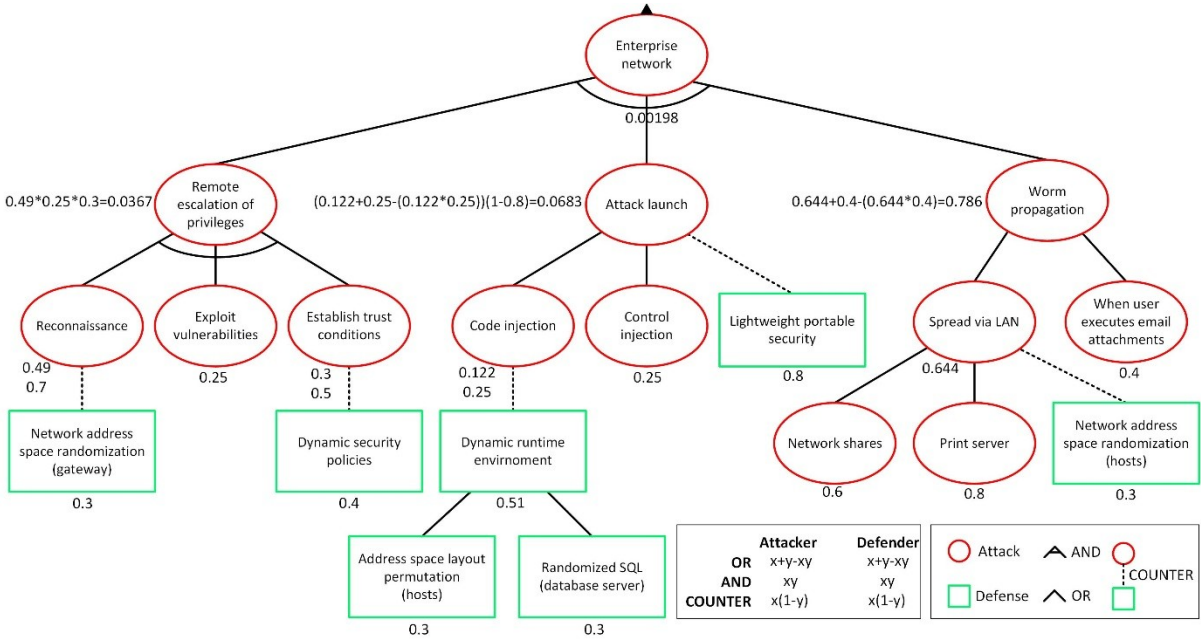

Figure 3 Attack--Defense Tree decorated with the likelihood attribute

ADTrees can be used to quantify security scenarios with respect to given parameters, called attributes. For example, an attribute could be the likelihood of satisfying the attacker's (or defender's) goal. The bottom-up algorithm is the most widely used approach for calculating attribute values. In this approach, the user first assigns attribute values to the leaf nodes and functions corresponding to each refinement type. Attribute values are then propagated up to the root node. As an example, we show the probability with which an attacker can compromise the enterprise network in Figure 3 (see the numbers associated to each node and functions at the bottom of the figure). For instance, let the probability with which the countermeasure *network address space randomization* (*gateway*) prevents a scanning attack be 0.3; then, an attacker performing *reconnaissance* with probability 0.7 will be successful with likelihood 0.49.

Therefore, we can infer that ADTrees are well suited to reason about the refinement of security goals into easily understandable actions, to capture the dynamics between potential attacks and defenses, and for simple attribute evaluation. However, to improve situational awareness, in addition to attack—defense information, we also need to gain insights on the impact of an attack (or a defense) on the system. We achieve this using the notion of dependency graphs.

4.2 Dependency graphs

A dependency graph is a directed acyclic graph that captures how system components depend on each other. The nodes of the dependency graph correspond to the network entities and an edge between two nodes (n_1, n_2) denotes the relationship n_1 *depends on* n_2. For instance, the dependency graph in Figure 4 (right side only) illustrates that workstation WS3 depends on WS2, which further depends on three servers. The application server, web server, email server and print server depend on the database server. We note that the dependencies at any abstraction level could be defined (e.g., instead of network entities, nodes in the graph could represent business units) and, as discussed in [CITATION MAl11 \l 1033], information related to the nature of dependencies (e.g., redundancy, graceful degradation, strict dependency) can also be represented.

We now demonstrate how the information provided by the dependency graph improves situational awareness. Consider the enterprise network described in Section 3; based on the ADTree in Figure 3, the system administrator may try to prevent worm propagation via the print server since the attacker is most likely to perform this action (with probability 0.8). However, considering the dependency graph in Figure 4, it appears that this choice may not be optimal because only workstation WS1 depends on the print server and consequently the damage is marginal. On the other hand, worm propagation via Email server or LAN, although less likely (0.4 and 0.6 respectively), could have a much higher impact (e.g., since WS2 and WS3 depends on Email server).

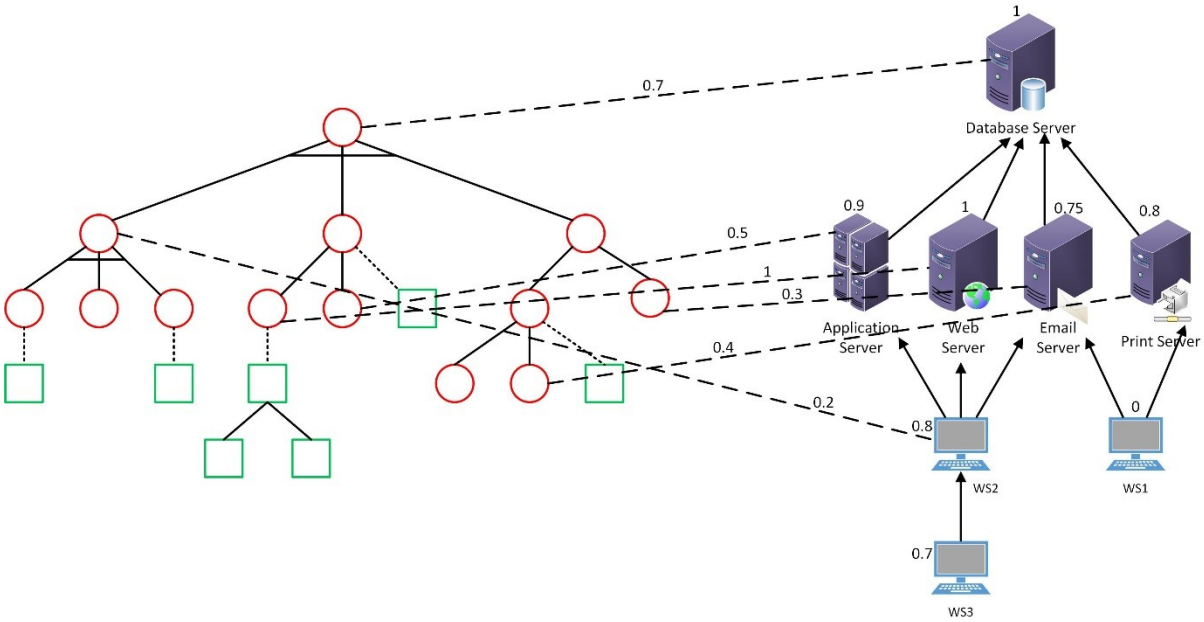

Figure 4 Attack--Defense Tree and the dependency graph

Our modeling approach consists in combining ADTrees and dependency graphs, and denoting how the execution of an attack or a defense (nodes in the ADTree) might *impact* the performance and utility of one or more system components (nodes in the dependency graph), and how this may further affect the components that depend on the directly affected components.

Example 2: Similarly to [CITATION MAl11 \l 1033] where attack graphs and dependency graphs are combined for scalable analysis of attack scenarios, assume that the system administrator associates a number with each network entity in the dependency graph to denote its performance/utility value. For simplicity, this number could be normalized in $[0,1]$ where 1 associated to the web server indicates that it is functioning at its optimal capacity and 0 associated to WS1 indicates that this workstation is completely unusable. Similarly, associate a number with each edge connecting an attack node in the ADTree and a network entity in the dependency graph. This number represents the percentage reduction in the utility of the network entity due to this attack. For instance, Figure 4 illustrates that the code injection attack makes the web server unusable. Since WS2 and WS3 depend strictly on the web server, their utility also becomes 0. Similarly, the damage caused by the control injection attack on the application server is 0.5*0.9=0.45 and its *total impact* is 0.45 + 0.8(WS2) + 0.7(WS3) = 1.95.

4.3 State transition system

The combination of ADTrees and dependency graphs provides insights about the interaction between attack and defenses and their impact on the system in a systematic manner. Finally, we need models that allow us to represent the state of the system and how it reacts to various attacks and defenses in order to obtain complete situational awareness. Such models are useful, for instance, to identify whether an adaptation moves the system to an invalid state or not. To achieve this, we introduce the notion of labeled state transition model (STS) in which a state characterizes the configuration of the system, and the transitions denote the actions that move the system from one state to another. State transitions can be enriched with system events, security

policy conditions, or probability distributions. The formalism of an STS with probability distributions, providing an equivalent representation of an ADTree, is given in [CITATION RJh16 \t \l 1033].

Example 3: For the ADTree in Figure 3, an abstract and representational STS in Figure 5 shows how the system transits from one state to another depending on the attacker's actions and MTD techniques. For instance, the MTD lightweight portable security, by reverting the OS to a known clean state, negates the exploits that the attacker had developed and removes any trust conditions generated. This moves the system from State 5 back to State 2. We note that MTD techniques remove the monotonicity assumption normally attributed to attacker's behavior in traditional security modeling, and is captured using STS. Similarly, by code or control injection, an attacker can successfully inject payload of the malware (i.e., complete the attack launch step and reach State 7) and move to the state (8, 9 or 10) where she can start propagating the worm within the network.

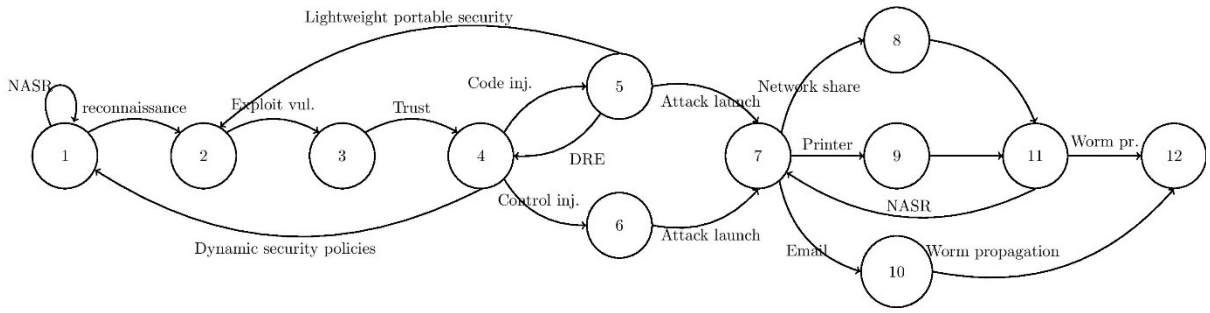

Figure 5 State transition model

In our framework, ADTree, dependency graph and STS provide situational awareness in an MTD system. These models not only capture the dynamics between potential attacks and defenses, and their impact on the system, but also the state of the system and involved agents. These models serve as our basis for assessing the effectiveness of MTD and decision-making.

5. Conclusions and Future work

MTD has the potential of complicating adversary's missions and of protecting Internet services and critical infrastructures. We provided an overview of the state-of-the-art and highlighted the research challenges that are constraining wide adoption of MTD. We then proposed a framework that outlined our model-driven approach to security evaluation and decision-making. Finally, we presented our modeling approach using ADTrees, dependency graphs and STS, and demonstrated how it provides situational awareness and how it can help us in realizing our framework. This paper is only the first step towards achieving our research goals in this area. As part of our future work, in addition to the points mentioned in the paper, we will develop the language, syntax and semantics of our models, and develop approaches to automatically generate these models.

Acknowledgements

The research leading to these results has received funding from the Fonds National de la Recherche Luxembourg under grant C13/IS/5809105.

6. References

Albanese, M., Battista, E., Jajodia, S., & Casola, V. (2014). Manipulating the attacker's view of a system's attack surface. *IEEE CNS*, (pp. 472-480). San Francisco, USA.

Albanese, M., Jajodia, S., Pugliese, A., & Subrahmanian, V. S. (2011). Scalable Analysis of Attack Scenarios. *ESORICS* (pp. 416–433). Leuven, Belgium: Springer.

Al-Shaer, E. (2011). Toward Network Configuration Randomization for MTD. In *MTD: Creating Asymmetric Uncertainty for Cyber Threats* (pp. 153-159).

Antonatos, S., Akritidis, P., Markatos, E. P., & Anagnostakis, K. G. (2005). Defending against hitlist worms using network address space randomization. *ACM Workshop on Rapid Malcode*, (pp. 30–40). Fairfax, VA, USA.

Chew, M., & Song, D. (2002). *Mitigating buffer overflows by operating system randomization.* CMUCS-02-197.

Donovan, P. J., McLamb, J. W., Okhravi, H., Riordan, J., & Wright, C. V. (2015). Quantitative evaluation of moving target technology. *HST* (pp. 1-7). IEEE.

Egesdal, M., Gomez-Jordana, A., Pelissier, C., Prause, M., Savani, R., & Stengel, B. (2015). *Game Theory Explorer*. Retrieved from http://gte.csc.liv.ac.uk/gte/builder/

Executive Office of the President, NST Council, USA. (2011). Retrieved from Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program: https://www.whitehouse.gov/

Hong, J. B., & Kim, D. S. (2016). Assessing the effectiveness of moving target defenses using security models. *IEEE Trans. on Dep. and Sec. Comp*, 163-177.

Hutchins, E., Cloppert, M., & Amin, R. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *ICCWS*. Washington, DC, USA.

Iyer, V., Kanitkar, A., Dasgupta, P., & Srinivasan, R. (2010). Preventing overflow attacks by memory randomization. *ISSRE* (pp. 339–347). IEEE.

Jhawar, R., Mauw, S., & Lounis, K. (2016). A Stochastic Framework for Quantitative Analysis of Attack–Defense Trees. *STM*. Heraklion, Greece: Springer.

Jhawar, R., Mauw, S., & Zakiuddin, I. (2016). Automated Cyber Defense Responses using Attack-Defense Trees and Game Theory. *ECCWS*, (pp. 163-172). Munich, Germany.

Jiang, X., Wangz, H. J., Xu, D., & Wang, Y. (2007). Randsys: Thwarting code injection attacks with system service interface randomization. *SRDS*. IEEE.

Jones, S., Outkin, A., Gearhart, J., Hobbs, J., Siirola, J., Phillips, C., . . . Mulder, S. (2015). *Evaluating Moving Target Defense with PLADD*. Sandia National Laboratories.

Kil, C., Jun, J., Bookholt, C., Xu, J., & Ning, P. (2006). Address space layout permutation: Towards finegrained randomization of commodity software. *ACSAC* (pp. 339–348). IEEE.

Kordy, B., Mauw, S., Radomirovic, S., & Schweitzer, P. (2014). Attack-defense trees. *Journal of Logic and Computation*, 55--87.

Kriaa, S., Bouissou, M., & Pietre-Cambacedes, L. (2012). Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments. *CRiSIS*, (pp. 1-8).

Manadhata, P. (2013). Game Theoretic Approaches to Attack Surface Shifting. In *MTD II: Application of Game Theory and Adversarial Modeling* (pp. 1-13). Advances in Inf. Sec., Springer.

Manadhata, P., & Wing, J. (2011). An attack surface metric. *IEEE Trans. on Software Engg.*, 37(3):371-386.

Okhravi, H., Rabe, M. A., Mayberry, T. J., Leonard, W. G., Hobson, T. R., Bigelow, D., & Streilein, W. W. (2013). *Survey of cyber moving target techniques*. MIT Lincoln Lab.

Roeder, T., & Schneider, F. (2010). Proactive obfuscation. *ACM Trans. Comp. Sys.*, 28(2):1–54.

Xu, J., Guo, P., Zhao, M., Erbacher, R. F., Zhu, M., & Liu, P. (2014). Comparing different moving target defense techniques. *ACM Workshop on MTD*, (pp. 97–107). Scottsdale, USA.

Zaffarano, K., Taylor, J., & Hamilton, S. (2015). A quantitative framework for moving target defense effectiveness evaluation. *ACM Workshop on MTD*, (pp. 3-10). Denver, USA.

Zhuang, R., DeLoach, S. A., & Ou, X. (2014). Towards a theory of moving target defense. *ACM Workshop on MTD*, (pp. 31-40). Scottsdale, USA.