# Model-driven, Moving-Target Defense for Enterprise Network Security

Scott A. DeLoach, Xinming Ou, Rui Zhuang, and Su Zhang

Department of Computing and Information Sciences
Kansas State University
234 Nichols Hall, Manhattan, KS USA 66506
{sdeloach,xou,rui,szhang}@ksu.edu
http://cis.ksu.edu/

**Abstract.** This chapter presents the design and initial simulation results for a prototype moving-target defense (MTD) system, whose goal is to significantly increase the difficulty of attacks on enterprise networks. Most networks are static, which gives attacker's a great advantage. Services are run on well-known ports at fixed, easily identifiable IP addresses. The goal of an MTD system is to eliminate the static nature of networks by continuously adapting their configuration over time in ways that seems random or chaotic to attackers, thus negating their advantage. The novelty of our approach lies in the use of runtime models that explicitly capture a network's operational and security goals, the functionality required to achieve those goals, and the configuration of the system. The MTD system reasons over these models to determine how to make changes to the system that are invisible to users but appear chaotic to an attacker. Our system uses these runtime models to analyze both known and unknown vulnerabilities to ensure that adaptations occur often enough and in the right ways to protect the system against external attacks.

**Keywords:** Runtime models, moving target defense, adaptive systems, network security.

## 1 Introduction

In cyber space, attackers have time to study our networks to determine potential vulnerabilities and choose the time of attack to gain the maximum benefit. Additionally, once an attacker acquires a privilege, that privilege can be maintained for a long time without being detected [4]. 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) [20]. 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 reduce/shift the attack surface area available for exploitation by attackers. An MTD system will make attacking a system more difficult

N. Bencomo et al. (Eds.): Models@run.time, LNCS 8378, pp. 137–161, 2014.

because the attacker will spend more time scanning the network for potential vulnerabilities and will not be able to maintain illegally acquired privileges for long. While promising, little research has been done to show that MTDs can work effectively in realistic networked systems.

Current approaches to network defense rely on reacting to attacker's efforts to penetrate the system. Similarly, current adaptive systems react to a variety of stimuli (e.g., system failure or new tasks) to trigger their adaptations. Thus, to be effective, MTD research must push beyond the existing state-of-the-art in both network security and adaptive systems in order to allow the system to adapt proactively without negatively affecting system functionality. Our vision is that an MTD system should be able to reason about its current configuration and make changes that are invisible to a valid user but appear chaotic to an attacker. In order to reason about its current configuration, runtime models that reflect the current configuration are needed that capture the modifiable aspects of the system and their relationship to the overall goals of the system. The *modifiable aspects* of the system are parts of the configuration that may be changed such as IP addresses, ports, firewall settings, host assignments, protocols, routing, virtual machines used, and software application type, versions, etc.

The research challenges in MTD systems are significant. First, we must find a way to model both the requirements, design, and current configuration (implementation) of the system in such a way as to allow automated reasoning. Second, we must provide a mechanism that supports automated reconfiguration of the system to include reassigning host addresses and returning the services to known good configurations. Third, we provide a mechanism that allows services to find the services they depend on in the midst of wide-spread system reconfiguration. Fourth, we must provide an adaptation mechanism (algorithm) that can adapt multiple aspects of the network's configuration in a way the mitigates the effect of attacks against critical network resources. And finally, we must integrate intrusion detection and risk assessment methodologies so that the system adaptation can respond to attack and risk indicators in a way that continues to appear random and chaotic to the attacker. This paper seeks to describe our initial approach at modeling the network requirements and design and demonstrate that an MTD based approach has potential for significantly increasing the difficulty of attacks on enterprise networks.

## 2 Moving Target Defense System

The high-level architecture of a simple MTD system that adapts randomly is shown in Figure 1 within the dashed box. Here, an *Adaptation Engine* orders (what appears to be) random adaptations to the network configuration at random intervals. These adaptations are carried out by a *Configuration Manager* that controls the configuration of the *Physical Network*. The key to these apparently random adaptations is that they are based on a *Logical Mission Model*, which is a runtime model that captures the Physical Network's current configuration as well as the functional requirements of the network. Since purely random

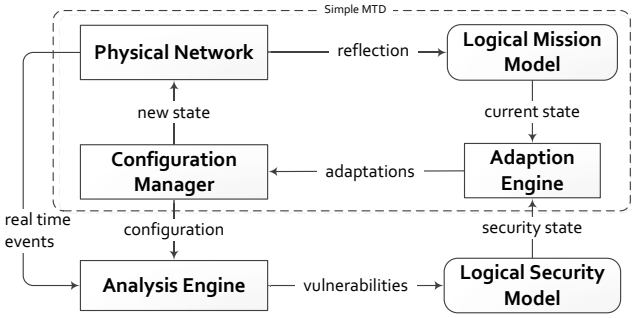


**Fig. 1.** Moving Target Defense System

adaptations would quickly yield the system inoperable, the adaptations must be made with an understanding of the requirements of the system in light of the current configuration. Specifically, the Logical Mission Model includes two runtime models: an organization model and a goal model. The *organizational model* captures the current configuration including the required functionalities in the system and the physical hardware capabilities. The *goal model* captures the system level requirements and the importance of the various requirements. We use the Organizational Model for Adaptive Complex Systems (OMACS) developed previously [10] for the organizational model and a new goal model, the Value-based Goal Model (VGM) [11], which is designed to capture requirements of long-lived, service-based systems.

While a simple MTD system holds promise, our ultimate vision for MTD systems uses apparently random changes in conjunction with intelligent control, where adaptations can occur randomly or based on risk indicators such as vulnerability scanning results and alerts from intrusion detection systems. The intelligent MTD system architecture extends the simple MTD architecture in Figure 1 by adding 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 uses the VGM like the Logical Mission Model; however, instead of capturing required functionalities, the Logical Security Model's VGM is used to capture the security goals of the system. The system vulnerability model is captured 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. If there are security issues that need to be addressed, the Adaption Engine uses these two models to determine an appropriate set of adaptations and sends them to the Configuration Manager (along with "random" adaptations) to implement.

### 2.1 Nomenclature

The terminology differences between enterprise networks, network security, and the abstract models we use in our research can be confusing. Therefore, for the remainder of the paper, we try to consistently use the following terms. We use the term *role* to refer to network services (e.g., web server, e-mail server, db servers) and the term *resource* to refer to the physical components of the system (e.g., computer hosts, firewalls, servers). The term resources is also equated to agents when referring to the OMACS model discussed in Section 4.2.

### 2.2 Resource Mapping System

One problem with a "moving" system is ensuring that system resources can locate each other after adaptations occur. Thus, to make these changes invisible to the system itself, a *Resource Mapping System* (RMS) is required, which also serves as a hardened system core that the attacker must penetrate to exploit the system. Current networks are so complex that even their system administrators have no clear understanding of the service dependencies [6,3,15]. In such complex systems, attacks can follow many patterns making their identification and prevention difficult. This is evidenced by research that shows an exponential increase in the number of attack paths in even moderate-sized networks [21,17]. The RMS interacts with the Configuration Manager, which pushes the current configuration to the RMS components. All communication between system roles must go through the RMS so that communications can be maintained even as the location of the roles change.

As shown in Figure 2, each role is assigned to a single virtual machine (VM), which has a dedicated RMS component that handles all communication with other roles. Each dedicated RMS component only knows the locations of the roles it needs to communicate with as defined by the role's communication requirements in the Role model (See Section 5). All communications between mission critical roles are controlled by RMS even as their locations change dynamically. In some sense the RMS functions like an end-host firewall with highly restrictive policies for critical roles; critical roles can be isolated on VMs with only the minimal ports open. Compared with traditional firewalls, the RMS provides flexibility for non-critical roles, while increasing protection for critical roles.

In modern virtual environments, isolating individual services on separate VMs provides the ability to tailor the VM's operating system to specific services and thus limit potential vulnerabilities. We believe that tailoring a VM's security environment while controlling communications via the RMS will provide a highly tailored security environment that will make successful attacks more difficult. However, due to its knowledge of the entire system configuration, the Configuration Manager is the key vulnerability of our design. We currently assume the Configuration Manager runs on a trusted host and significant resources are used to ensure its safety.

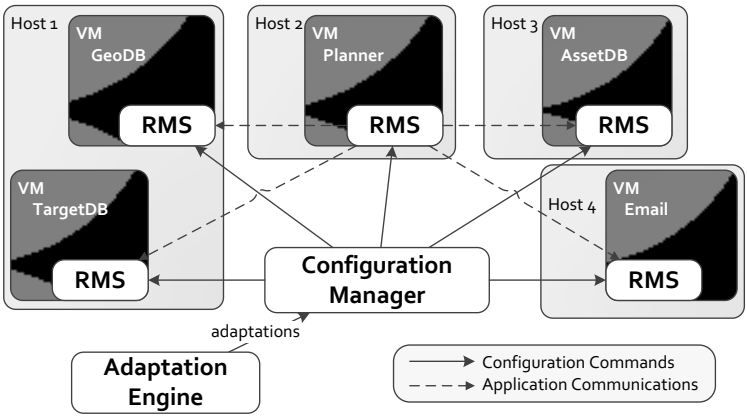


**Fig. 2.** RMS System

### 2.3 Adaptation Engine

The assignment of roles to resources allows the system to adapt autonomously while ensuring all mission goals are still supported. In traditional adaptive system, the adaptation algorithm would attempt to provide optimal or near optimal configurations [4]. However, since the goal of MTDs is to produce non-predictable configurations, we must consider alternative approaches. Instead of seeking optimal configurations (in terms of system performance), which tends to produce the same configuration over and over, we must develop algorithms that find near optimal configurations that are significantly different in some aspect.

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. The assignment of roles to resources is similar to our existing reassignment algorithms [10] and is based on ensuring the resources have the appropriate capabilities to play the role. Since we use unique VMs for each role, we can assign any available IP address on the network to a new VM. If the role's communication is supported by the RMS, the port number can also be randomly selected. Firewall settings can be updated based on the knowledge of which VMs actually need to communicate. Specific application types (e.g., Apache, Oracle, Hiawatha, etc.), versions, or VM types can also be specified by the Adaptation Engine. Finally, we can consider adapting the protocols for various critical roles that communicate via the VM. Such protocol changes could be minor while still allowing the RMS to easily detect compromised VMs or physical resources. A further discussion of how the models are used in the adaptation process is given in 4.5 after the models are presented.

### 2.4 Analysis Engine

The Analysis Engine is based on Dr. Ou's existing work on MulVAL [22,21] and SnIPS [23,28]. 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 Analysis Engine outputs a CAG that is derived from the Role model (see Section 5) dependencies and incorporates real-time evidence to infer the network's security state.

Traditional enterprise security risk assessment uses vulnerability scans and firewall configurations to identify potential attack paths into the system. This has a number of disadvantages, such as the inability to mitigate risk due to unknown threats (e.g., zero-day vulnerabilities). Intrusion detection system (IDS) and Security Information and Event Management System (SIEM) are typically deployed for the purpose of situational awareness and forensics. The analysis engine in our envisioned system will take input from these traditional sources but map them to the unique conservative attack graph (CAG) model due to the dynamic nature of the adaptation. The unique advantage moving-target brings to security analysis is that the usable attack surface is greatly reduced due to shifting, and the false-positive challenge in intrusion detection can be mitigated by proactively adapting the system even with less than certain attack indicators.

The CAG model is also used in our simulation study of the effectiveness of the system against both known and unknown attacks. In our model we assume each host could contain exploitable vulnerabilities and for this reason there is no distinction between known and unknown vulnerabilities in our simulation. When a CAG is used in deployed systems, however, such distinctions will matter and we intend to build models to capture the impact of both known and unknown vulnerabilities in the moving-target system as part of our future work.

## 3 Example System

To demonstrate that MTD systems can be effective for network security defense, we simulated an MTD system using a simple military mission planning system that allows authorized users to access a mission planner. We provide an overview of that system here and use it to illustrate our proposed runtime models in Section 4. The mission planning system, shown in Figure 3, supports users located both inside and outside the local network. The system allows users to access three different databases in order to construct a specific mission. The databases that the planner accesses are an *asset database* that includes the types and numbers of assets available to carry out planning, a *target database* that includes the intelligence on targets of interest, and a *geographical database* that includes maps and geographical information about the areas required for planning appropriate ingress, target attack, and egress routes.

In this system, the likely targets of interest are not the Authorizer or Planner systems, but the data behind them. Specifically, the TargetDB and AssetDB have the most potentially important data, thus, they would likely be the targets of an attack. In our simulation, we assume the TargetDB is the main focus.

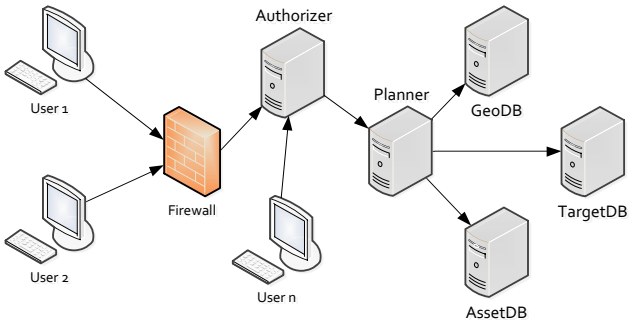


**Fig. 3.** Mission Planning System

## 4 Runtime Models

The key to our MTD approach lies in novel runtime models based on human organizations that explicitly capture a system's goals, the functionality required to achieve those goals, and the logical and physical configuration of the system. These runtime models allow the system to reason about its current state and make changes that are invisible to the user but appear chaotic and significantly increase the difficulty for an attacker. Key runtime models include:

- a Value-based Goal model (VGM) that captures the system's mission and security requirements
- an OMACS model that captures the physical resources in the system, their capabilities, the software functions available carry out system goals, and the current assignment of functions to physical resources
- a Conservative Attack Graph (CAG) that captures both known and unknown vulnerabilities based on the current system configuration

### 4.1 Value Based Goal Model (VGM)

It has recently been recognized by the adaptive systems community that the key to highly efficient and effective adaptive systems is explicitly modeling the requirements or objectives of the system [5,25], a position we have espoused for several years [9,10,12]. Specifically, we capture the system objectives as goals, which allows the system to adapt while still ensuring it can support its overall goals. As there are trade offs during adaptation, understanding which goals are the most important is critical to ensuring the system adapts appropriately. Thus we capture the system's mission and security goals in a novel Value-based Goal model (VGM), which allows us to determine the effect of attacks and adaptations on system functionality and security [18].

Formally, a VGM is a tree whose nodes are value-based goals rooted at goal $g_0$, as shown in Figure 4 where $g_0$ is the *Mission Goal*. Typically, $g_0$ represents the overall operational goal of the network or the overall goal of system security.

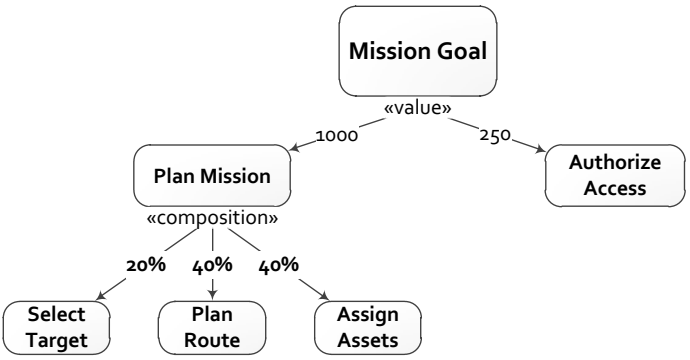


**Fig. 4.** Value-Based Goal Model

Goals are typically defined as some desired state of the world, and this is true of value-based goals as well. However, value-based goals are not achievement goals whose state must be attained by the system, but instead are maintenance goals whose state must be preserved by the system [8]. Thus instead of achieving goals, the objective of a value-based system is to maintain the maximal value of a set of goals expressed in a unique decomposition and value aggregation approach.

The key to determining the overall value of a VGM tree is to know which goals are currently maintained. Thus, we define a set called the *maintained set* that captures the current set of goals maintained by the system. The maintained set is computed by first determining the leaf goals in the maintained set and then computing the parent goals that are in the maintained set. The value of a VGM is based on the current set of goals in the maintained set. Thus, the current value of the VGM as well as future values of the VGM with different maintained sets can be computed. The current value of any goal that is not maintained is zero.

The root goal, $g_0$, of a value-based goal model represents the overall value of the system. Goal $g_0$ is always a *value goal*, which is decomposed into a set of sub-goals, each of which are assigned a maximum value. The current value of the $g_0$ is simply the sum of the values of its children, which can range from 0 to their maximum values. The root value goal is decomposed into one or more of the following types of goals: Composition, AND, OR, or Leaf.

If a goal is a *Composition goal*, all of its sub-goals contribute a percentage to its value. Thus, each sub-goal of a Composition goal has an associated contribution value and the contributions of all sub-goals of a Composition goal must equal 1.0. The current value of a Composition goal is the sum of the sub-goal contributions that are currently maintained.

An *AND goal* denotes the case when all sub-goals must be maintained in order for the parent goal to be maintained and contribute its maximum value. In some cases, the current values of an AND goal's sub-goals may be maintained, but not at the maximum value. Thus, we define the current value of an AND goal to be the minimum value of all its sub-goals (if one is not maintained its value is 0).

The maximum values of each sub-goal of an AND goal is the maximum value of the AND goal since the failure to maintain any one of the sub-goals reduces the value of the parent to zero.

An *OR goal* is similar to an AND goal as its value is based on a Boolean operator, in this case logical OR. Thus, if any sub-goal of an OR goal is maintained, then the OR goal itself is maintained. However, unlike the AND goal, each sub-goal has an individual contribution value associated with it, stated in terms of a percentage (0 to 100) of the OR goal's maximum value. The notion of an OR goal is that there may be multiple ways to maintain a specific goal, although some may be better than others.

*Leaf goals* have no sub-goals and contribute to the overall value of the goal tree based on their parent's type. Actually, only Leaf goals are actively maintained by the system. As the system maintains (or fails to maintain) Leaf goals, the overall value is aggregated based on parent goal types until a final value for the system is computed. Using the description above, it should be noted that the value of a system is not simply the value of all its Leaf goals and thus care should be taken when using the values of Leaf goals independently of their parent goals. In many cases, the value of a Leaf goal (that are sub-goals of AND/OR goals directly or indirectly) can only be computed in light of a specific configuration.

In an MTD, a VGM captures the relative importance of the system goals in case trade-offs must be made. As shown in Figure 4, our example system's objectives are decomposed into two main goals: allowing external users access to the system (Authorizer) and allowing users to plan missions (Plan Mission). The Plan Mission goal is decomposed into a set of subgoals, where each subgoal is weighted to express its contribution to its parent goal. Thus, based on the system's VGM, if there are not enough resources to achieve all goals, the *Plan Mission* goal is more important than the *Authorize Access* goal and thus the system should try to support the *Plan Mission* goal.

### 4.2 Organization Model for Adaptive Complex Systems (OMACS)

The Organization Model for Adaptive Computational Systems (OMACS) [10] is a model that defines the knowledge required to allow a team of agents to reorganize in response to agent failure or changing team goals. While adapting to failure and changing goals can be a benefit in a network-based system, our objective in using OMACS as the basis for our MTD system is to use this knowledge to ensure the adaptations carried out in a defensive effort do not inhibit the system's ability to achieve its goals. As shown in Figure 5, the key entities in OMACS include a set of goals, roles, agents, and capabilities. For our MTD system, the *goals* represent the functional requirements of the system, *roles* represent services (such as the applications, web servers and database servers), *agents* represent physical resources such as computer hosts, and *capabilities* represent agent attributes such as memory, bandwidth, and installed software. This information is used to compute the assignments (configurations) that tell agents the roles they are assigned to play in order to achieve system goals.

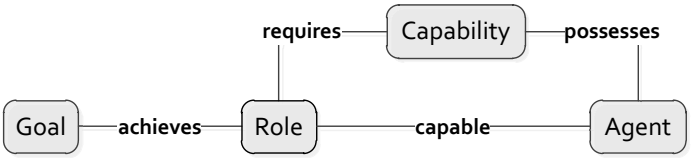


**Fig. 5.** Key OMACS Entities

Organizations are generally formed with some specific objective or goal in mind. In OMACS, the overall goal of an organization is represented by a set of goals that the organization is trying to achieve. The relationship between the various goals is not handled directly by OMACS but is entrusted to the VGM. More specifically, OMACS goals are the leaf goals in a VGM model. Goals are achieved by agents playing specific roles within the organization.

Every OMACS organization has a set of agents, which in a computer network are the physical computer hosts available for use. Agents possess capabilities that are required to play roles for the network.

In general, OMACS roles denote a set of responsibilities or the expected behaviors. In our approach, we use roles to describe services such as the applications, web servers and database servers required to support (and thus maintain) various system goals as shown in Figure 6. Each role has two types of characteristics that are critical to effective system adaptation: requirements and attributes. The specific requirements and attributes of each role are used by the Adaptation Engine to select the appropriate agent to carry out those roles. Each role has a set of required capabilities such as processing power, memory amount, bandwidth, and installed software. In a minor extension to OMACS, MTD roles also contain a set of attributes that give the RMS and the assigned physical resource (and its VM) precise directions on how to setup and run that role. To support the assignment process, OMACS defines the *achieves* function, which takes as input a goal and a role, and returns a value that reflects how well the given role achieves the given goal type.

Roles are defined in a Role model as shown in Figure 6. The Role model not only captures the requirements and attributes of each role, but also defines the communications that must be allowed between roles, which is critical to the definition of the CAG and the operation of the RMS system. In our example system, each mission leaf goal from Figure 4 is supported by a role, namely the Planner, AssetDB, TargetDB, GeoDB, and Authorizer roles. The relationship between goals and roles is formally captured in the *achieves* relation. When the system is running, these roles are assigned to physical resources such hosts or VMs while their communications are supported by the RMS.

Before a role is assigned to an agent (i.e., before a service is deployed on a host), the agent must meet the requirements for that role. Capabilities are essential in determining the roles that each agent is capable of playing. Capabilities are used to represent a wide variety of abilities. In a computer network, capabilities are used to model the hardware and software capabilities of a network

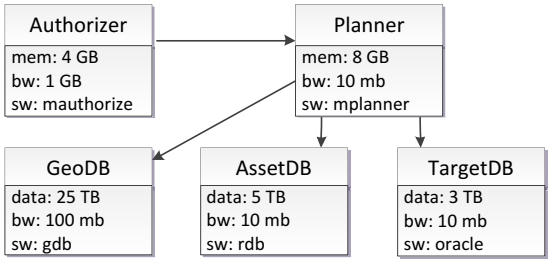


**Fig. 6.** Role Model

resource such as processing power, memory amount, bandwidth, and installed software. If an agent has all the required capabilities to play a role, the role capability function (rcf) is used to compute how well the agent can play that role, thus allowing designers to indicate the importance of specific capabilities to each role.

To determine the best overall set of assignments (configurations) for a specific set of goals, OMACS defines an organization assignment function (oaf). The oaf determines the effectiveness of a given set of assignments and assigns it a value. In normal systems, the optimal oaf value is selected when a reorganization (in our case an adaptation) is required. However, in an MTD system, we must use non-optimal configurations since our goal is to produce a constantly changing attack surface. A complete definition of OMACS can be found in [10].

### 4.3 Physical Resource Model

The capabilities of OMACS agents are taken from the Physical Resource model, which captures the configuration of available resources (e.g., computers, firewalls, servers, etc.) that support the operational system. Each resource in the physical resource model has a particular set of capabilities that can be used to play roles in the operational system. A role's physical capability requirements and the assigned role attributes are compared with resource capabilities to determine if the role can be assigned to that resource. In addition, role communication requirements are used to modify firewall (RMS) configuration as needed. With the advent of virtual machines (VMs), virtual resources can be created when and where (logically) required. The use of VMs supports the movement of roles between physical resources. In addition, the use of VMs allows a single role to be executed on a single VM and thus makes the security configuration of the VMs simpler and more secure.

As a side benefit, the ability to adapt by changing the mapping of goals to roles to physical resources also allows the MTD to respond effectively to changes and failures in the physical configuration of the network. If a role is running on a resource that fails, the MTD must identify that a goal is no longer being supported and assign that role to a new resource.

### 4.4 Conservative Attack Graphs

An integral effect of MTDs is that an attacker must continually regain the knowledge and privileges obtained through prior attacks. This effect invalidates the typical monotonicity assumption found in most attack-graph work where an attacker cannot lose a privilege after gaining it. In an MTD, it becomes important to model losing privileges due to constant changes in the system configuration. The frequency and type of adaptations affect how far an attacker can move forward in a system. Modeling such dynamism requires a state-machine model, rather than the commonly used dependency attack graph. Previous state-enumeration attack graphs were not scalable to large attack graphs. However, we do not need to apply a fine-grained attack-graph models to analyze the effect of MTDs since MTDs do not attempt to counter known vulnerabilities, but use dynamism to counter *assumed* vulnerabilities at every node. Thus, we use a conservative attack graph (CAG), which assumes the existence of unknown vulnerabilities without enumerating all possible vulnerabilities. This assumption actually makes the model smaller and lends itself to stochastic analysis through a state-machine model.

Figure 7 shows the CAG for our example. The topology of the CAG is partially derived from the dependencies specified in the Role model. As shown in Figure 6, the Authorizer role initiates interactions (depicted by the arrows between roles) with the Planner role, which initiates interactions with the TargetDB, GeoDB, and AssetDB roles. Because the RMS system limits communication between system roles, we can assume that the only paths between roles are those allowed by the RMS. Thus, 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 (TargetDB, GeoDB, and AssetDB). Thus, our CAG captures these logical access paths.

The RMS components on the VMs implement the network communication policy derived from the Role model to adhere to the logical paths. If an RMS component is compromised, the attacker would potentially be able to bypass this control and try to access roles not exposed to the VM. However, in such situations the compromised RMS would not know the location of those roles and thus the attacker would have to correctly guess the IP and address (among other aspects), which is a low-probability event. Thus, if an RMS is compromised, the only realistic attack path is along the paths of the CAG.

### 4.5 Model-Driven Adaptation

This section shows how the Adaptation Engine uses the models to make adaptations to the system configuration. All adaptations are initiated by a triggering mechanism. In an MTD system, the trigger could be a timer (for random adaptations), a goal modification (addition, deletion, or changing of various goal values), or a change in the current state of the system (either software/hardware failure or identification of a potential intrusion). The end goal of an adaptation is to produce a configuration that ensures that system goals are achieved at the

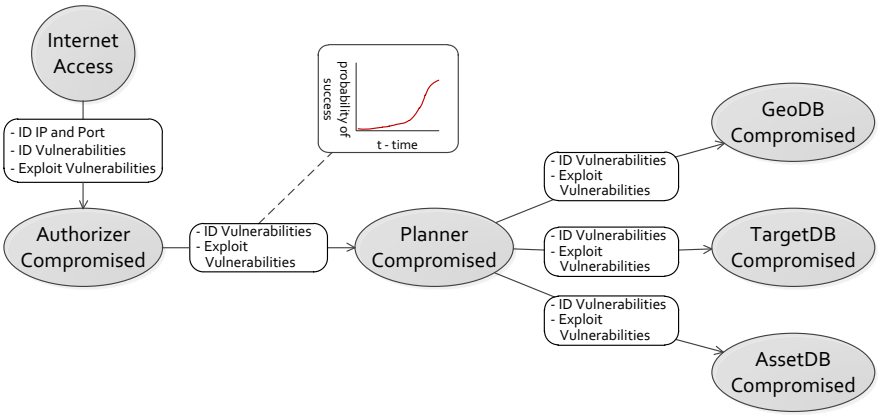


**Fig. 7.** Conservative Attack Graph

highest possible value. Here we assume all goals are achievable by the available roles and agents; however, if they are not, the least valuable goals can be dropped until a valid configuration is obtained.

For the initial configuration, a role (a service) is selected that achieves each goal and is assigned to an agent (a host) that provides the capabilities required to carry out that role. This configuration is given to the Configuration Manager, who makes the physical assignments and provides the appropriate knowledge to the RMS components.

If a failure occurs, goals that are no longer being achieved due to the failure are reassigned to new role-agent pairs. If a potential intrusion is detected, the goals and roles of the agents that are involved in the potential intrusion (source or destination) are reassigned to new role-agent pairs. When goals are added, new assignments must be made while when goals are deleted, old assignments may be removed. When a random adaptation is triggered, the Adaptation Engine selects a specific goal-role-agent assignment in the system to modify along with a specific modifiable aspect and a new assignment for the goal is generated. In all cases, the changes determined by the Adaptation Engine are passed on to the Configuration Manager who makes the appropriate changes in the physical system.

The key to random adaptation is ensuring that adaptations are as unpredictable as possible within a reasonable cost. Ideally, the probability of adapting a particular aspect and agent would be represented as a uniform probability distribution across the entire domain of the configuration space, thus maximizing the entropy of the system [27]. However, a system with maximum entropy would likely degrade system performance to the point where the system performance would be unusable. Therefore, we plan to investigate approaches that allow for a trade off between system entropy and performance/cost.

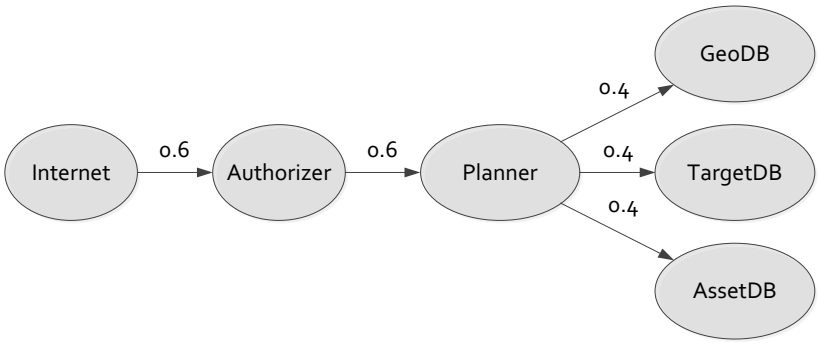


**Fig. 8.** Simplified Conservative Attack Graph for Simulation

## 5 Simulation Results

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 [26]. NeSSi2 is an open-source, discrete-event based network security simulator with extensive support for constructing complex application-level scenarios based on a simulated TCP/IP protocol stack [26]. In this simulation, we assumed the user had full knowledge of the logical system configuration and only attacked through the 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. 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.

The overview of the simulated network is shown in Figure 8. The edges in the graph (with the exception of the Internet to Authorizer edge) show the valid paths supported by the RMS. We assume the attacker is located at the Internet node and wishes to attack the TargetDB. In the first simulation, we assume the attacker can only attack along the valid RMS paths. However, in the second and third simulations, we assume that attacker attempts to attack through valid RMS paths as well as directly between hosts (e.g., Authorizer to TargetDB).

To simplify our simulations, we made several assumptions.

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, which also includes changing the VM's IP address.

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 in the attack path.
5. The attacker knows the basic system architecture as defined by the Logical Mission Model and thus the attack is restricted to the VMs playing those five roles.
6. The attacker knows immediately when a resource 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.

### 5.1 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 as shown in Figure 9. The *Defense component* contains the Configuration Manager, three physical resources (hosts) and five active VMs, which can be assigned to play host to any of the five roles: Authorizer Planner, TargetDB, AssetDB, or GeoDB. At a preset time interval, $\Delta t$, the Configuration Manager selects an adaption by randomly picking an existing role and refreshing its VM which includes modifying its address. Then, the Configuration Manager notifies the affected hosts of the changes and updates the Ground Truth component with new current configuration.

The *Attack component* simulates the attacker and uses the CAG shown in Figure 8 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 the Internet to the Authorizer, from the Authorizer to the Planner, and then from Planner to the 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 and would likely increase over time. The success of attacks is determined randomly based on the probabilities in the CAG as well as whether or not the nodes in the attack have been refreshed during the attack.

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

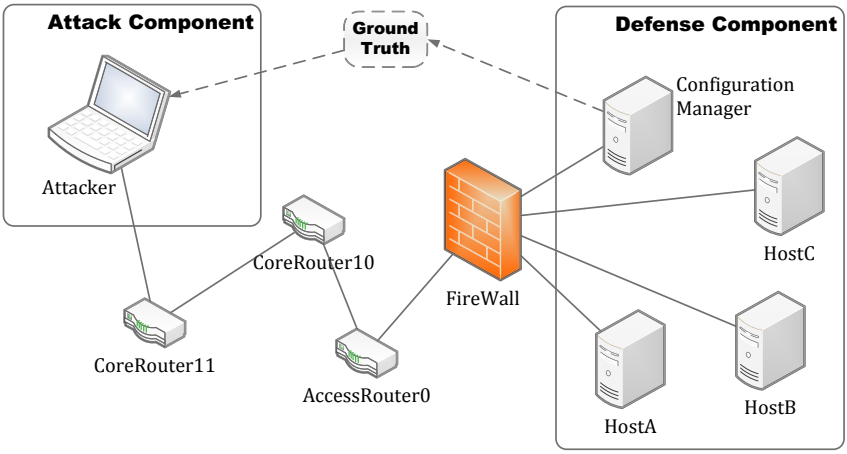


**Fig. 9.** Network Topology

the three host resources: hostA, hostB, and hostC. These six components are loaded onto the corresponding nodes as shown in Figure 9. The hosts do not actually perform their assigned role responsibilities, but merely exist to give the attacker something to attack. The results of our initial experiments are presented in the next section.

**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 the current VM and retried the attack until successful or until the MTD system refreshed the VM. In each experiment, we performed 1000 single step attacks with a fixed $\Delta t$ 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 $\infty$) between each adaptation. Note that an $\infty$ adaptation interval corresponds to a completely static system.

In the experiment 1a, we assumed that in order to stop a single step attack from succeeding, 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 an single step attack occurring from the Planner to the TargetDB, it could be stopped if either 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. Figure 10 shows the ability of the MTD to deter a successful attack from the

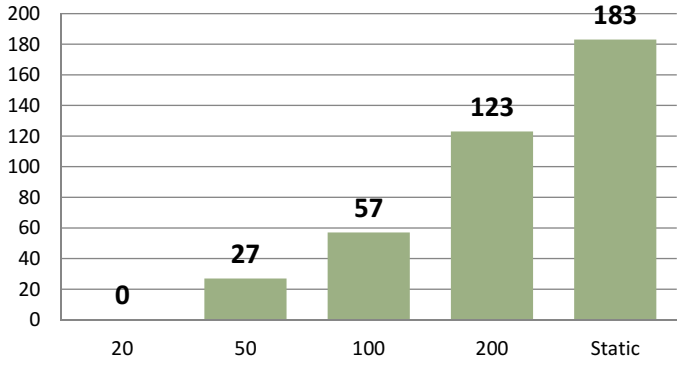


**Fig. 10.** Attack Success Against TargetDB (assuming only refresh of current node or attacked node inhibits attack)

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 decrease. 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 10 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. Figure 11 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 11 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is obvious.

### 5.2 Broad Attack Simulation System

In the broad attack simulation, the attacker is again attempting to compromise the TargetDB. Since the attacker knows the details of the system configuration, it can use the RMS to its advantage; however, the attacker also attacks outside the RMS to stress the MTD defenses. For these simulations, we assume a sophisticated attacker who automatically attacks each available node in the network

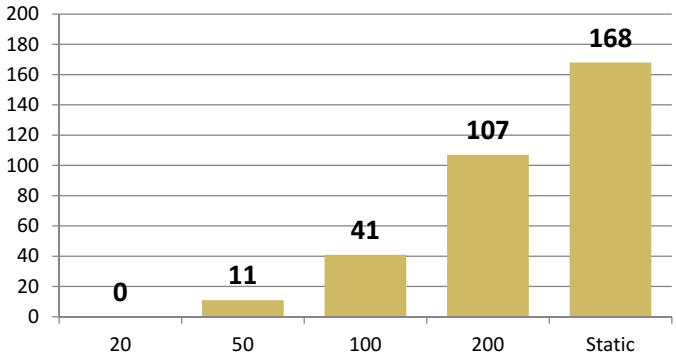


**Fig. 11.** Attack Success Against Target DB (assuming refresh of any node in path inhibits attack)

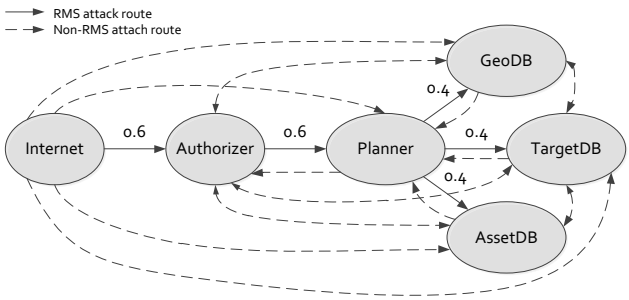


**Fig. 12.** Attack Success Probabilities in Broad Attack Simulation (dashed lines have a probability of p/65,536 where p is the probability of successfully attacking the role through the RMS)

from its current location using the RMS or by attempting to guess the address and port of an available node. Therefore, the attacker is not limited to the RMS routes and thus the attack routes form a completely bidirectionally connected graph (except for the Internet node, which only has arrows to nodes into the network) as shown in Figure 12. However, since the RMS will *not* respond to standard network requests for mapping information, this eliminates the ability for the attacker 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 12. 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. 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 (once 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 and 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).

**Broad Attack Simulation Results.** We conducted 1000 runs (as opposed to 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. Since the broad attack simulation allowed the attacker to keep attempting to attack network nodes as long as the attacker had access to a compromised network node, 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 $\Delta t$ between each attack of 100 time intervals. For each experiment, we ran the 1000 runs using 5 different adaptation intervals (20, 50, 100, 200 and $\infty$).

Figure 13 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

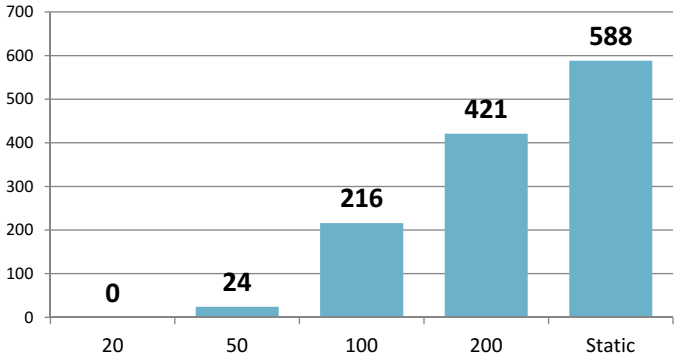


**Fig. 13.** Attack Success Against TargetDB for Broad Attack Simulation Against Simple MTD

interval of 20 totally eliminated the ability of the attacker to compromise the TargetDB. Once again, Figure 13 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is clearly visible.

### 5.3 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 $\Delta t$ as used in the previous experiments.

**Intelligent MTD Simulation Results.** The result of the intelligent MTD simulation is shown in Figure 14; 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

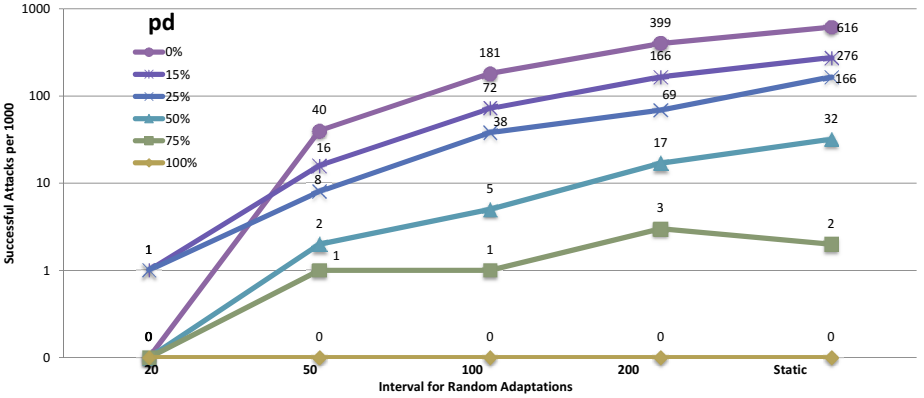


**Fig. 14.** Attack Success Against TargetDB for Broad Attack Simulation Against Intelligent MTD

intelligent MTD system. The RMS minimizes the attack surface to such a degree that attacks outside the RMS are easily detected and significantly decrease the attackers likelihood of success.

When compared with the attack success rate of the simple MTD system (which is represented by the line in Figure 14 with $p_d = 0\%$), we see that the intelligent MTD system performs significantly better since the simple MTD system. We do see a slight anomaly in the data since at an adaptation interval of 20, when the $p_d$ is both 15% and 25%, we see 1 successful attack while there are 0 successful attacks with a $p_d$ of 0%. Although the probability of success is extremely low, the attacker can succeed. We believe that with more runs (than 1000) the data would have normalized. Overall, while not conclusive, this experiment clearly shows the need for further investigation into the costs and benefits of intelligent MTD systems.

### 5.4 Discussion

The design of our MTD is based on knowing the current situation, which is captured in a set of runtime models. These runtime models allow the system to reason over the current state of the system and produce adaptations to confuse and rebuff potential attackers. The system design shows how a set of runtime models can be combined to model and reason over multiple aspects of a complex problem. Specifically, this system includes models for the system configuration, the system objectives (operational and security), and entities external to the system (the attackers).

The simulation presented here is our first, and one of the first anywhere, simulation of an MTD architecture for enterprise network security. As such, the simulation implemented only a simple MTD system and did not use the full power of its runtime models. However, the results demonstrate the potential effectiveness of MTDs for enterprise computer networks. Therefore, we plan to

continue to simulate more complex systems (in terms of nodes and interconnections), increase the sophisticated of the attackers, and integrate in the full power of an intelligent MTD system based on runtime models. In addition, we are currently building a real world MTD test bed using existing visualization technologies.

This system also demonstrates the applicability of the OMACS model to new and novel applications. The OMACS model was originally envisioned as a metamodel to capture multiagent systems. However, it quickly became clear that by looking at existing research on human organizations, the model could be made much more general. Since its inception, OMACS has been applied to multiagent systems, cooperative robotics, human-robot teams, sensor networks, and power distribution systems. In fact, the claim could be made that OMACS can be applied to any domain in which distributed agents (natural or artificial) need to coordinate their actions to achieve shared objectives and adapt to the current state of the environment or their problem solution.

## 6 Related Work

### 6.1 State of Practice

Network configuration currently tends to be static and routine assumptions are made about the location of services in terms of fixed URL's or IP addresses. Such static configuration is largely due to the use of legacy system components. The benefit of static configurations appear to be ease of management and programming. However, it has been observed that the static network configurations (i.e, service dependencies) have actually made it harder to manage systems, especially when changes must be made [3,15,6]. The state of the practice in computer network defense relies upon firewalls in both network and application layers, intrusion detection and prevention systems, and anti-malware products that provide defense in depth. Unfortunately, once a method is found to circumvent these mechanisms, the attacker can keep attained privileges until discovered. In addition, the attacker can generally use the same methods to circumvent other similar defenses. Here, the lack of dynamism is an important contributor to the ease with which an attacker can launch a successful cyber attack.

### 6.2 Moving Target Defenses

Most of the prior work on MTDs in a network context has been related to 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 [16,2] 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 [16], they required all trusted computers be shielded by special processes and displayed had several application interoperability issues [19]. More recently, a

network address space randomization scheme to thwart hit list worms [1], which configured DHCP servers to expire the leases of hosts at various intervals to support address randomization. In [7], 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. In each of these cases, only the network addresses were changed or made to look like they changed and only served to confuse attackers without the ability to automatically remove them from the system once they compromised a resource.

In a different approach to MTD, Roeder and Schneider [24] propose to use *proactive obfuscation* to create application replicas with identical functionality but dissimilar vulnerabilities that react differently to identical attacks. The authors showed that with sufficient entropy in the executables, the approach effectively thwarted known attacks without greatly increasing costs. We anticipate that proactive obfuscation could be employed in our MTD approach to increase the difficulty of initial compromise as well as the ease with which attackers could reacquire resources after being removed from the system.

## 7 Conclusions

In this paper we presented a preliminary design of an MTD system that uses runtime models of a network's requirements, design, and implementation to allow it to adapt its configuration to increase the difficulty of attacks on the network. We conducted several simulation-based experiments to study the effects of randomly adapting the system 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 adding intelligence to the decision of when and where to adapt in the form of detectors that could detect when an attack occurred outside the normal RMS system. Our results showed that even with less than perfect detectors, significant improvements in network security can be made. These results clearly demonstrate the potential for both simple and intelligent MTD systems and are preliminary steps toward developing a comprehensive evaluation and analysis framework for MTD systems.

**Acknowledgments.** This work was supported by the Air Force Office of Scientific Research under award no. FA9550-12-1-0106, and U.S. National Science Foundation under award no. 0954138 and 1018703. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above agencies.

## References

1. Antonatos, S., Akritidis, P., Markatos, E.P., Anagnostakis, K.G.: Defending against hitlist worms using network address space randomization. Computer Networks: The International Journal of Computer and Telecommunications Networking 51, 3471–3490 (2007)

2. Atighetchi, M., Pal, P., Webber, F., Jones, C.: Adaptive Use of Network-Centric Mechanisms in Cyber-Defense. In: Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2003), pp. 183–192. IEEE Computer Society, Washington, DC (2003)
3. Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D.A., Zhang, M.: Towards highly reliable enterprise network services via inference of multi-level dependencies. In: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2007), pp. 13–24. ACM, New York (2007)
4. Barrett, D.: Hackers Penetrate Nasdaq Computers. Wall Street Journal, `http://online.wsj.com/article/SB10001424052748704709304576124502351634690.html` (February 5, 2011)
5. Bencomo, N., Whittle, J., Sawyer, P., Finkelstein, A., Letier, E.: Requirements reflection: Requirements as runtime entities. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010), vol. 2, pp. 199–202. ACM, New York (2010)
6. Chen, X., Zhang, M., Mao, Z.M., Bahl, V.: Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI 2008), pp. 117–130. USENIX Association, Berkeley (2008)
7. Compton, M.D., Hopkinson, K.M., Peterson, G.L., Moore, J.T.: Network Obfuscation Through Polymorphic Routing and Topology Control. IEEE Transactions on Dependable and Secure Computing (2012) (in preparation)
8. Dardenne, D., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20, 3–50 (1993)
9. DeLoach, S.A., Miller, M.: A Goal Model for Adaptive Complex Systems. International Journal of Computational Intelligence: Theory and Practice 5, 83–92 (2010)
10. DeLoach, S.A., Oyenan, W., Matson, E.T.: A Capabilities-Based Model for Artificial Organizations. Journal of Autonomous Agents and Multiagent Systems 16, 13–56 (2008)
11. DeLoach, S.A., Ou, X.: A Value Based Goal Model. Multiagent and Cooperative Robotics Laboratory Technical Report No. MACR-TR-2011-01. Kansas State University (2011)
12. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent Systems Engineering. The Intl. Journal of Software Engineering and Knowledge Engineering 11, 231–258 (2001)
13. Grimaila, M.R., Fortson, L.W., Sutton, J.L.: Design Considerations for a Cyber Incident Mission Impact Assessment (CIMIA) Process. In: Proceedings of the 2009 International Conference on Security and Management, SAM 2009 (2009)
14. Hellesen, D., Grimaila, M.R.: Information Asset Value Quantification. In: Proceedings of the 2010 International Conference on Information Warfare and Security (ICIW 2010), pp. 138–147 (2010)
15. Joukov, N., Pfitzmann, B., Ramasamy, H.V., Devarakonda, M.V.: Application-storage discovery. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference (SYSTOR). ACM, New York (2010)
16. Kewley, D.L., Bouchard, J.F.: DARPA Information Assurance Program dynamic defense experiment summary. Systems, Man and Cybernetics, Part A: Systems and Humans 31, 331–336 (2001)
17. Lippmann, K.W., Ingols, C., Piwowarski, S.K., Kratkiewicz, K.J., Artz, M., Cunningham, R.K.: Evaluating and strengthening enterprise network security using attack graphs. Technical Report. MIT Lincoln Laboratory (2005)

18. McQueen, M., McQueen, T., Boyer, W., Chaffin, M.: Empirical estimates and observations of 0day vulnerabilities. In: 42nd Hawaii International Conference on System Sciences, pp. 1–12 (2009)
19. Michalski, J., Price, C., Stanton, E., Chua, E.L., Seah, K., Heng, W.Y., Pheng, T.C.: Final Report for the Network Security Mechanisms Utilizing Network Address Translation LDRD Project. Technical Report SAND2002-3613. Sandia National Laboratories (2002)
20. National Cyber Leap Year Summit 2009, Co-Chairs' Report. (September 16, 2009)
21. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: 13th ACM Conference on Computer and Communications Security, pp. 336–345. ACM, New York (2006)
22. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: A logic-based network security analyzer. In: 14th USENIX Security Symposium, Baltimore, Maryland, U.S.A (August 2005)
23. Ou, X., Rajagopalan, S.R., Sakthivelmurugan, S.: An empirical approach to modeling uncertainty in intrusion analysis. In: Annual Computer Security Applications Conference, pp. 494–503 (December 2009)
24. Roeder, T., Schneider, F.B.: Proactive obfuscation. ACM Trans. Comput. Syst. 28, 4:1–4:54 (2010)
25. Sawyer, P., Bencomo, N., Whittle, J., Letier, E., Finkelstein, A.: Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In: Proceedings of 18th IEEE International Requirements Engineering Conference, pp. 95–103. IEEE Press, New York (2010)
26. Schmidt, S., Bye, R., Chinnow, J., Bsufka, K., Camtepe, A., Albayrak, S.: Application-level Simulation for Network Security. SIMULATION 86, 311–330 (2010)
27. Shannon, C.E.: A Mathematical Theory of Communication. Bell Syst. Technical Journal 27(3), 379–423 (1948)
28. Sundaramurthy, S.C., Zomlot, L., Ou, X.: Practical IDS alert correlation in the face of dynamic threats. In: International Conference on Security and Management (2011)