Mission-oriented Moving Target Defense Based on Cryptographically Strong Network Dynamics

Justin Yackoski Jason Li
Intelligent Automation, Inc.
Rockville, MD USA
{jyackoski, jli}@i-a-i.com

Scott A. DeLoach Xinming Ou
Kansas State University
Manhattan, KS USA
{sdeloach, xou}@ksu.edu

ABSTRACT

This paper describes a computer network moving target defense (MTD) system that incorporates the benefits of both literal modifications of various network aspects along with semantic changes made to several fundamental aspects of the network. The result is a cryptographically strong MTD system that is transparent to legitimate users while appearing random and chaotic to potential attackers.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access—*Management of computing and information system*

Keywords

moving target defense, enterprise network security

1. INTRODUCTION

The static nature of current networks gives attackers time to study our networks, determine potential vulnerabilities and choose the time of attack. In addition, once attackers acquire a privilege, they can maintain that privilege for a long time without being detected. Further, detection-based security approaches are likely to remain imperfect. A promising approach to eliminating these advantages is called the *moving target defense* (MTD) [1], which, for computer networks, can be interpreted as changing various aspects of the network constantly to reduce/shift the attack surface available for exploitation by attackers. It is also possible to implement an MTD by constantly manipulating the *appearance* of the network, where dynamics are imposed at the network layer to secure attributes such as device location/identity, service availability, user authentication, and network topology.

In this paper we present a preliminary design of a moving-target defense system that incorporates both physical modifications of various network aspects with semantic adaptation at the network layer. The result is a cryptographically strong MTD system that is transparent to users yet appears random and chaotic to potential attackers.

CSIIRW '11 October 12-14, Oak Ridge, Tennessee, USA

Approved for Public Release; Distribution Unlimited: 88ABW-2012-4397, 10-Aug-2012

2. MTD POLICY MANAGEMENT

Our vision for MTD supports adapting multiple aspects of a network (e.g., IP addresses, ports, firewall settings, host-application assignments, application types/versions, protocols, etc.) simultaneously and continually, all while being transparent to legitimate users. An overview of our framework is shown in Figure 1, which combines the ability to adapt randomly over time with intelligent control to make the network configuration appear chaotic to a potential attacker. A more detailed discussion of our proposed design and results of experiments showing its potential effectiveness can be found in [4].

The general operation of the MTD system is driven by the Adaptation Engine, which orders adaptations to the current configuration. These adaptations are carried out by the Configuration Manager, which creates a set of configuration policies that are implemented on the Physical Network. The state of the network is reflected in a Logical Mission Model and a Logical Security Model, which are fed back into the Adaptation Engine.

While the *Adaptation Engine* orders what appears to be random adaptations to the network configuration at random intervals, in reality these adaptations can be chosen randomly or based on risk indicators such as vulnerability scanning results and IDS alerts. However, without great care, adapting the network on the fly would quickly yield the system inoperable since services could be assigned to inappropriate hosts or the communications paths required for the system to work appropriately could be interrupted. To enable real-time adaptations to work effectively, the underlying MTD system must have an understanding of the functional and security requirements of the system. In our system, this understanding is based on a *Logical Mission Model* and the *Logical Security Model*, which reflect the current network configuration and security state as well as the functional and security requirements of the network. With this information, the MTD system can make apparently random adaptations with an understanding of the requirements of the system and the current configuration.

A key enabler of our design is the abstract *Logical Mission Model*, which captures the network resources, the services used, and the dependencies between services that are required to achieve the overall mission of the network. A key element of the Logical Mission Model is the *Mission Goal Model*, which captures the overall mission of the network

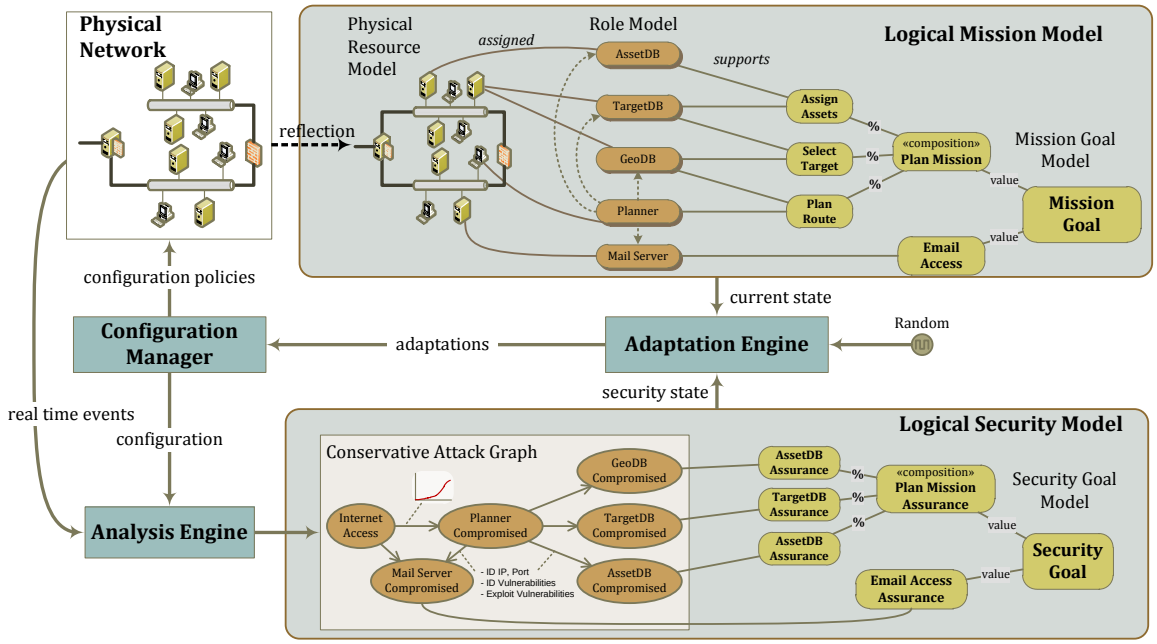

Figure 1: Design scheme for a network MTD system

and decomposes that mission into subgoals achievable by various parts of the system. The rest of the Logical Mission Model is based on the Organizational Model for Adaptive Computational Systems (OMACS) [2], which allows intelligent reasoning algorithms to assign *agents* (hosts) to play *roles* (provide services) in an organization (system) in order to achieve specific *goals*. The logical design of services is captured in the *Role Model*, which shows how various roles interact to achieve the mission goals. The Adaptation Engine determines an acceptable assignment of roles to physical resources based on the security state, role requirements and computational capabilities required.

The *Analysis Engine* takes real-time events from the Physical Network and the current configuration from the Configuration Manager to determine possible vulnerabilities and on-going attacks, which enables intelligent control. The use of intelligent control techniques allows the MTD to react to suspected intrusions as well as adapting randomly, which allows the MTD to mitigate unpredicted attacks and mask intelligent control system actions. By incorporating reactions into adaptations, the system can react to suspected intrusions much sooner than a normal intrusion response system since even responses to false positives will leave the system in an operational state with no more overhead expended than for a random adaptation.

The Analysis Engine updates the *Logical Security Model*, which captures potential attacks and known vulnerabilities via a *Conservative Attack Graph* (CAG) and their effect on the security goals of the system as defined in the *Security Goal Model*. The CAG is a novel abstraction that captures the ability of the attacker to gain and lose privileges as adaptations will affect how far an attacker can move forward in a system. Unlike traditional attack graphs, a CAG *assumes* there is an attack path between any two assets as long as the attacker is able to identify the target asset from the source asset. Thus, we do not need a fine-grained attack-graph, but instead propose a *conservative* attack graph as shown in Figure 1, which assumes the existence of unknown vulnerabilities without enumerating them. The topology of the conservative attack graph is partially derived from the Role Model and assumes attack paths are constrained by the Self-shielding Dynamic Network Architecture (SDNA) explained below. A CAG can be viewed as a state-transition system, where each arrow is annotated with a label describing the activities involved to move from one state to the next. The effort involved in these activities can be measured in various ways such as a success-likelihood versus time graph. From each state, there is also a probability for the attacker to be forced to "move back" to one of the prior states along the path, due to the MTD mechanisms.

Adaptations are implemented by a Configuration Manager who controls the configuration of the Physical Network. Besides informing physical hosts how to adapt their current configuration (e.g., in terms of adapting the virtual machines (VMs) running on them, the applications/operating systems running on the VMs, etc.), the Configuration Manager also determines the configuration policies required for operation of the SDNA.

An MTD system creates significant challenges in a network setting, the most important of which is allowing legitimate users/roles to locate required resources in the midst of the adaptations. Thus, there must be a mechanism that allows valid roles to communicate securely via valid access paths by mapping the semantic views defined by these roles into the current state of the network. We propose using the Self-shielding Dynamic Network Architecture (SDNA) as described in Section 3, to provide this functionality. A key motivation of our design is that if an attacker deviates from the access paths allowed by the SDNA they become extremely obvious, allowing the generation of alerts and tracking of the attacker's activities. SDNA implements the network communication policy (derived from the Role Model)

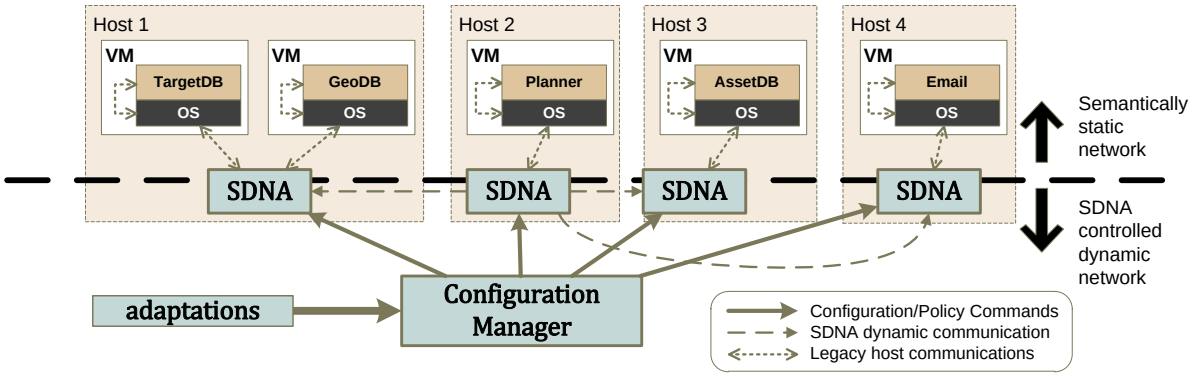

Figure 2: Self-shielding Dynamic Network Architecture

to adhere to the logical paths. If an SDNA-protected node is compromised, the attacker would also be limited to those logical paths and thus we can assume that a successful attack must follow the pre-defined service access paths, which dramatically reduces the attack surface of the system.

3. INCORPORATING A SELF-SHIELDING DYNAMIC NETWORK ARCHITECTURE

In our vision, the Self-shielding Dynamic Network Architecture (SDNA) serves as the security policy enforcement unit for each role/VM. As such, SDNA's purpose is to (1) coordinate with the Configuration Manager via policies to provide legitimate access to services as the location of those services change, and (2) to limit attackers that have compromised a protected node to following defined paths, significantly limiting the scope of attacks and simplifing intrusion detection and prevention. SDNA is designed to maintain compatibility with existing operating systems, applications, and network hardware to avoid affecting legitimate users while simultaneously dramatically impacting the actions of malicious users [3]. By inserting a hypervisor in each host as shown in Figure 2, SDNA makes the network appear dynamic to observers while retaining the semantics necessary for transparency to legitimate users. In the network "below" SDNA, packets traverse a dynamic network while on the OS and application side of SDNA, the dynamics are concealed to provide an abstract but semantically valid view of the network.

SDNA provides cryptographically-secure mechanisms to add network dynamics that prevent attackers from gathering and acting on information about the network using a combination of existing networking techniques, hypervisor technology, Common Access Card (CAC)-based authentication, and IPv6.

The objective of these dynamics is to, first, force the attacker to spend significant resources to carefully guide attacks, as packets sent that do not correctly follow the network dynamics allow the attack to be easily and immediately detected. Second, SDNA reveals a view of the network that is sanitized, ambiguous, and time-varying to any attempts to probe or map the network (including attempts from the SDNA-protected VM). Finally, we inhibit malicious behaviors from within the protected VM by varying the availability of services based on user needs and credentials.

The challenge of achieving such a vision lies in the need to impose dynamics for an attacker while simultaneously hiding the dynamics from existing operating systems, applications, user expectations, routers, switches, and other components. One aspect not to be overlooked when imposing dynamics in this way is that the network's semantic meaning must be preserved at some level of abstraction. As a result, such techniques need mechanisms to share information about the dynamics as well as to ensure that those "in the know" regarding the network dynamics are legitimate and that the information available is abstracted to limit its usefulness.

3.1 Security Controls Provided by SDNA

SDNA uses a combination of techniques including packet manipulation via a node's hypervisor, communication through intermediate nodes, DNS manipulation, and credentials. These techniques allow manipulation of the network's appearance in ways that improve the security of existing technologies such as firewalls, VPNs, and IDSs. Figure 2 shows how SDNA's dynamics can be enacted and managed by a set of policies provided by a Configuration Manager.

- **Protection against information gathering and sharing** – SDNA prevents nodes from determining the actual nodes they communicate with or that provide specific services.
- **Protection of vulnerable services** – Services can be protected against all attacks by redirecting packets unless its originator has been verified, thus protecting the OS by removing several common attack vectors. This forces the attacker into revealing a set of valid credentials under their control (limiting their future use) and creating an audit trail (allowing faster recovery from the attack).
- **Fine-grained security controls** – SDNA allows policies to be easily defined and applied down to granularity of creating individualized views of each service/role for each user (i.e. potential attacker). Since the OS of a compromised node does not directly access the network, attackers cannot overcome this protection.

3.2 Implementation and a concrete example

SDNA has been tested with existing operating systems, applications, and on network devices including laptops, PCs, routers, and switches. While several deployment options are supported by our implementation, in this paper we describe

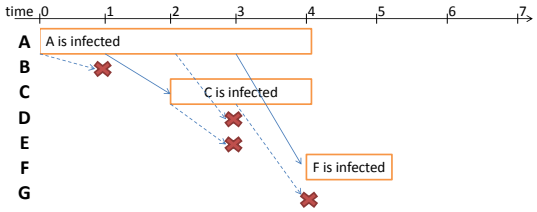

Figure 3: Worm's actions in unprotected network

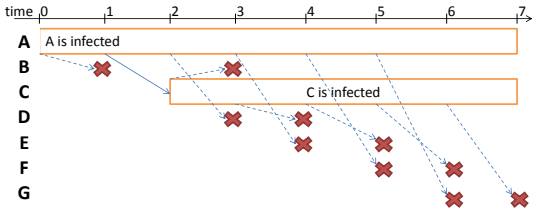

Figure 4: Worm's actions under SDNA's dynamics

an unmodified operating system (Windows, Linux, etc.) separated from SDNA logic using the Xen hypervisor.

To show how adding dynamics into a network can improve security, a simple concrete example is provided where a worm has infected a node and tries to spread to the rest of the network. We assume several nodes share a common vulnerability that allows remote infection via a single packet. In a typical IPv6 network, the worm will likely attempt to send the infection packet to other nodes by observing IPs used, guessing the IP assignment scheme, or querying DNS hostnames. We assume that initial infection of node A results in an ordered list of six targets (B thru G). As shown in Figure 3, suppose node C (the second host attempted) results in a successful infection and thus node A can divide the remaining effort needed to spread through the network with node C. If F is the only additional host found to be vulnerable and infected, a total six infection packets will be sent and two additional nodes will have been infected. Assuming the attacker knows the IDS will trigger an alert if more than one infection packet is sent per minute by a node given, the entire process takes 4 minutes.

In an SDNA enabled network, the attack is affected significantly as shown in Figure 4. Suppose SDNA has been given policies that require C to be available to A (e.g. A has a user that requires access to C's service/role), but F is not available to either A or C. In this case, node A can still infect node C. However once C is infected, the remaining scanning cannot be distributed. For example, while A could not infect or contact B, it may be due to dynamics that hide B from A but not C. C must therefore re-scan B. For the same reason, A and C must each attempt to contact nodes D thru G as a node that is not available to A it may be available to C and vice versa. Infection of F is not possible in this case since neither A and C have a policy-defined availability to F. Beyond simply limiting the spread of the worm to node C instead of both C and F as in the previous case, the amount of time, effort, and risk required by the attacker has been increased since a total of 11 infection packets must now be sent (6 by A, 5 by C) and the attack takes 7 minutes to complete (note that C doesn't begin scanning until after it is infected and the same IDS detection threshold is present).

4. DISCUSSION

A drawback to any MTD system is that there are a set of critical roles that must exist. In other words, the network must still function to allow the mission to be conducted. However, by slowing the attacker's use of these roles and creating an abstract view of the network that requires additional time and effort from the attacker, the system's security can be improved even in the case of a compromised node in the network. This power of combining SDNA with virtual machine (VM) level dynamics through a cohesive MTD policy system is made clear by this example. If the MTD includes VM refreshing as an adaptation mechanism, SDNA will slow the spread of such worms to allow time to eventually refresh all VMs (including those infected) thus removing the worm from the system.

In our approach, if a node is compromised, other nodes along valid access paths become potentially vulnerable since communication with them may appear to be valid. However, due to SDNA, the attacker must follow the exact communication paths defined by the configuration policies and the abstract view provided by SDNA requires more effort from the attacker, thus dramatically reducing the attack surface and simplifying detection. Here, adaptation comes to the rescue as, eventually, the compromised node's VM will be refreshed and the attacker's privileges will be lost.

While SDNA has been implemented and demonstrated in a variety of configurations, a complete MTD system is currently being developed. Building on SDNA will provide a solid foundation for our MTD work as communications between the network nodes will occur in a secure manner. By removing the ability to map the network based on analyzing communication patterns and packets while simultaneously changing the network topology (randomly or based on intrusion alerts), we believe we can eliminate the advantages currently enjoyed by attackers thus greatly increasing network security.

5. ACKNOWLEDGEMENT

This work was supported by the Air Force Office of Scientific Research under award no. FA9550-12-1-0106, the Air Force Research Laboratory under award no. FA8750-11-C-0179,and U.S. National Science Foundation under award no. 1038366 and 1018703.

6. REFERENCES

[1] National Cyber Leap Year Summit 2009 co-chairs' report, networking and information technology research and development. Technical report, Sept. 2009.

[2] S. A. DeLoach, W. Oyenan, and E. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16:13–56, 2008.

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

[4] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal. Simulation-based approaches to studying effectiveness of moving-target network defense. In *National Symposium on Moving Target Research*, 2012.