A Theory of Cyber Attacks
A Step Towards Analyzing MTD Systems

Rui Zhuang Alexandru G. Bardas Scott A. DeLoach Xinming Ou
Kansas State University
Manhattan, KS USA
{zrui, bardasag, sdeloach, xou}@ksu.edu

ABSTRACT

Moving Target Defenses (MTD) have been touted as a game changing approach to computer security that eliminates the static nature of current computer systems – an attacker's biggest advantage. While promising, the dynamism of MTD introduces challenges related to understanding and quantifying the impact of MTD systems on security, users, and attackers. To analyze this impact, both the concepts of MTD systems and cyber attacks must be formalized. While a theory of MTD systems was proposed in [18], this paper presents a theory of cyber attacks that supports the understanding and analysis of the interaction between MTD systems and the attacks they hope to thwart. The theory defines key concepts that support precise discussion of attacker knowledge, attack types, and attack instances. The paper also presents concrete examples to show how these definitions and concepts can be used in realistic scenarios.

Categories and Subject Descriptors

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

Keywords

moving target defense; computer security; network security

General Terms

Science of Security

1. INTRODUCTION

Moving Target Defenses (MTD) have been touted as a game changing approach to computer security that eliminates the static nature of current computer systems [10]. While promising, the dynamism of MTD introduces challenges related to understanding and quantifying the impact of MTD systems on security, users, and attackers. MTD can be thought simply as constantly changing a computer system to reduce or move the exploitable attack surface, which are the resources available to attackers (e.g., software, ports, component vulnerabilities, etc.) that can be used to compromise the system.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
MTD'15, October 12 2015, Denver, CO, USA
Copyright is held by the owner/author(s).Publication rights licensed to ACM.
ACM 978-1-4503-3823-3/15/10 . . . $15.00.
http://dx.doi.org/10.1145/2808475.2808478.

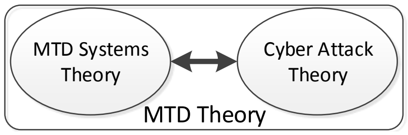

Figure 1: MTD Theory Overview

To date there is little research that shows how and why MTD techniques may work effectively in real systems against specific types of attacks. Part of this problem comes from the fact that MTD systems pose new challenges in understanding and quantifying the effectiveness of such systems. Part of what is needed is a comprehensive theory of MTD that defines what an MTD system is, how it works, and how it interacts with attacks and attackers to thwart those attacks. While a theory of MTD systems was proposed in [18], a theory of cyber attacks that supports the understanding and analysis of the interaction between MTD systems and attackers is still missing. In this paper, we propose a theory of cyber attacks that provides the concepts and definitions required to further study the effectiveness of MTD systems.

1.1 A Theoretical Framework Overview

Developing a science for cyber security has been acknowledged as a critical need within the research community [14]. The goal of this work is to create a robust body of knowledge related to cyber security based on theoretical and empirical research that can support the development of truly secure systems. Our work in moving target defense theory will not only define a set of common terms and essential problems, but it will also provide a systematic framework for analyzing the potential effectiveness of MTD solutions.

Our approach to developing a complete MTD theory is shown in Figure 1. The first step was to develop a theory of MTD Systems [18] that focuses on the system itself and how it adapts over time to achieve its security goals. The second step, and the subject of this paper, was to develop a theory of Cyber Attacks, which describes how an attacker uses specific types of attacks to achieve its ultimate goals. The final step will be to compose the two into an overall MTD Theory to define how elements of the MTD Systems and Cyber Attack theories interact. This is especially important in being able to understand the true effect of an MTD system as its effectiveness only makes sense in light of actions from an attacker for a specific type of attack.

1.2 MTD System Theory Background

MTD Systems Theory [18] defines *MTD systems* based on the concept of a *configurable system* , which is defined in terms of configuration parameters. *Configuration parameters* are used to capture the notion of the configuration units that a configurable system can control and is formalized as a ⟨name, value⟩ pair. Each configurable system thus has a *configuration state*, which is the aggregation of the current values of all the configuration parameters associated with the MTD system. An MTD system can also take actions, called *adaptations*, that allow it to modify the values of its configuration parameters and thus its configuration state.

MTD system theory builds on years of adaptive systems research and captures the goals (both operational and security) and policies of the system as essential elements. *Goals* and *policies* ensure that the system can achieve its intended purpose within stated constraints, and are used to determine valid configurations of an MTD system. Specifically, *adaptations* are defined as a sequence of actions that change the system from one starting configuration state to another valid configuration state.

Using our basic MTD system definitions, we defined various key concepts often used in cyber security circles. First, we defined the *configuration space* as the set of valid states in which an MTD system can exist. We also formally defined the concepts of *diversification* and *randomization* and showed how diversification increases the system configuration space while randomization techniques adapt the system configuration to increase the effectiveness of those systems.

We also formally defined three key research problems related to MTD systems. The *MTD Problem* was defined as how to select the next configuration state of the MTD system. Next, the *Adaptation Selection Problem* was defined as how to select the adaptations to perform in order to get to the next configuration state, while the *Timing Problem* was defined as when to carry out the adaptations to actually change the state of the system. However, to solve these problems, theories focused solely on an MTD system and its properties are insufficient. For example, the best choice for the next configuration state depends on a candidate state's impact on the attacker. In other words, does it invalidate an attacker's knowledge? Does it require the attacker to invest additional effort or resources? Does it thwart specific types of attacks that might compromise the system? Answering these questions requires a way to formally talk about the attacker as well as the target system.

2. CYBER ATTACK THEORY

Cyber attack success relies on information possessed by an attacker when the attack is launched and is often measured by the information gained or modified as a result of the attack. Thus, information must be an essential element of any theory of cyber attacks. MTD Systems Theory included the notion of a configuration parameter that captures information about the configuration of a computer system, or more generally a target device. This configuration information is clearly part of the information of interest in an attack. However, the information of interest to an attacker goes beyond simple configuration information. For example, a target system's execution status and data are not configuration information, but are critical to many types of attacks. To capture all information of interest, we introduce a new concept called an *information parameter*, where the system's configuration parameters is a subset of the system's information parameters.

Thus, a target device is described by a set of information parameters and an attacker expends effort to gain or modify a target device's information parameters. Figure 2 highlights this relationship. In addition, an attacker generally has a lot of information that is not necessarily related to the target, which consists of knowledge about other target systems, specialized skills, or general purpose knowledge.

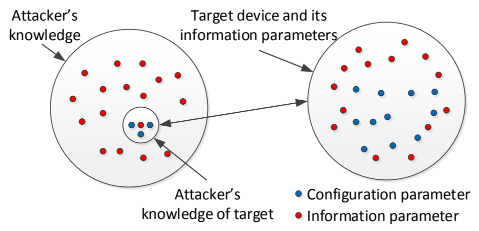
Attacker's knowledge
Target device and its information parameters
Attacker's knowledge of target
● Configuration parameter
● Information parameter

Figure 2: Attacker and Target System Overview

In Cyber Attack Theory, we formally define attacks and attacker's knowledge in terms of information parameters. By reasoning over the relationship between an MTD system's configuration parameters and an attack's information parameters, we can formally describe and analyze interactions between attackers and MTD systems.

Next, we introduce a scenario to help motivate and explain Cyber Attack Theory.

2.1 Motivating Scenario

We start with a simple military mission planning system, which is shown in Figure 3. Authorized users can remotely access the mission planner to construct military type missions. The Planner (a web server with a user interface) allows users to carry out authorized actions, such as adding new strategies, establishing plans/tactics or allocating resources. To support these actions, the Planner accesses three associated databases – the AssetDB, GeoDB and TargetDB.

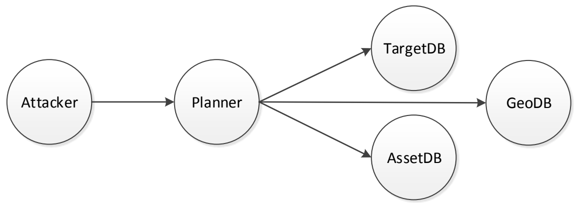

Figure 3: Motivating Attack Scenario

In this scenario, attackers try to exploit a vulnerability in the Planner. The vulnerability details are unimportant, but we assume that if the vulnerability exists, the attacker can exploit it. A second scenario, featuring a more concrete code reuse attack and an address space layout randomization (ASLR) MTD, is provided in Section 3.

2.2 Targets

We start our presentation of Cyber Attack Theory by defining the concept of a target, which is based on the concept of information parameters introduced above. We formally define a target and a target system from the attacker's perspective and then build on them by defining concepts associated with attackers and cyber attacks.

When we talk about a target, we view it as a device or system of devices that can be described by all kinds of information, such as configuration parameters, execution state and various other information types. Thus, we start by formally defining an information parameter.

2.2.1 Information Parameter

We define an information parameter as a name value pair with an associated type, which defines the domain of possible values an information parameter can assume.

DEFINITION 2.1. *An* information parameter, *$\psi = \langle n, v \rangle$, is a unit of information that can take on a value based on its type, where n is the name and v is a value.*

DEFINITION 2.2. *An* information parameter type, *Ψ, represents the domain of possible values that an information parameter value can assume. We denote the domain of information parameter ψ_j as Ψ_j. An assignment of some value z in Ψ to ψ is denoted as $\psi.v \leftarrow z$.*

An information parameter is basically a variable to which we can assign values from its domain. This is essentially the same definition as that of a configuration parameter. The difference is that a configuration parameter captures information only about the configuration of a device, types of software on that device, specific software settings, etc. Note also that a device's the configuration parameters are a subset of its information parameters.

In order to aggregate all the information parameters of a device or system, we define the notion of a composite information parameter.

DEFINITION 2.3. *A* composite information parameter, *ψ, is an information parameter that is composed of a set of sub information parameters, $\psi = \langle n, \{\psi_1, \psi_2, \ldots, \psi_n\} \rangle$, where n is the name, and $\psi_1, \psi_2, \ldots, \psi_n$ are sub information parameters. The domain of a composite information parameter, ψ, is derived from the sub information parameter domains, $\Psi = \Psi_1 \times \Psi_2 \times \ldots \times \Psi_n$.*

For example, the Planner in Figure 3 has a set of information parameters that might include $\psi_1 = \langle \mathit{pageviews}, 15/d \rangle$, $\psi_2 = \langle \mathit{visitors}, 250 \rangle$, $\psi_3 = \langle \mathit{memory}, 8GB \rangle$, $\psi_4 = \langle \mathit{cpu}, \mathit{intel\ i5} \rangle$, $\psi_5 = \langle \mathit{eax}, 0x65442224 \rangle$, $\psi_6 = \langle \mathit{os}, \mathit{Ubuntu}\ 14.04 \rangle$, and $\psi_7 = \langle \mathit{ip}, 222.20.22.22 \rangle$. Here, ψ_1 and ψ_2 are information parameters but not configuration parameters because they are statistically determined by visitors. ψ_5 is an information parameter but not a configuration parameter because the content of EAX reflects the execution status of a program. The remaining information parameters are also configuration parameters.

2.2.2 Target

Targets are generally thought of as devices on a computer network that an attacker may compromise, modify or gain information from. However, we also include humans as potential targets because users such as administrators, developers, and clients are also prime targets of attackers.

DEFINITION 2.4. *A* target, *d, is any device an attacker may try to obtain information from, break into, alter or destroy. In cyber space, the targets are computer network devices and communication channels, such as machines (either physical or virtual), humans, routers, switches, cellphones, cables, optical fiber, etc.*

In our example system, the Planner, AssetDB, GeoDB, TargetDB and all users of the mission planning system are potential targets for an attacker. To link the information parameters to the target they represent, we define the predicate *describes*.

DEFINITION 2.5. *If an information parameter, ψ, represents some aspects of the configuration, data or execution state of a device, d, we say ψ describes d, which we denote as the predicate describes(ψ, d).*

In the mission planning example ψ_1 through ψ_7 can all be used to *describe* the Planner. If we have a composite information parameter that captures *all* the relevant information that describes a specific device, we say that composite information parameter is complete as defined below.

DEFINITION 2.6. *Each device d has a unique* complete information parameter *ψ_d whose value captures all the information that describes d and only information that describes d. Formally, this is stated as*

$$\forall d, \exists \psi_d, \mathit{describes}(\psi_d, d) \wedge (\forall \psi, \psi \in \psi_d \Rightarrow \mathit{describes}(\psi, d)) \wedge (\neg \exists \psi, \psi \notin \psi_d \wedge \mathit{describes}(\psi, d))$$

It should be noted that complete information parameters refer to the information parameters *defined* for that device. Obviously, the set of information parameters for a device could be very large, of which only a subset are of practical use as we will see later.

Using the notion of a complete information parameter, we define the state of a particular target device or system as the current values associated with each information parameter in the device's complete information parameter.

DEFINITION 2.7. *The* state, *s_d, of target d, is current value of the complete information parameter of d, ψ_d. We can also refer to the state of d at time t.*

Obviously, the state of a target captures the value of ψ_d at a given point in time. This fact will become important when we discuss the effect of attacks in Section 2.4.

2.2.3 Target System

Now that we have defined a target, a target system simply becomes the composition of a set of targets. This essentially allows us to capture the large, complex computer systems that are the target of many attackers today.

DEFINITION 2.8. *A* target system, *D, consists of a set of targets, $D = \{d_1, d_2, \ldots, d_k\}$.*

DEFINITION 2.9. *Each target system, D, has a* system information parameter, *ψ_D, which is the set of each target device's complete information parameter, which is defined as $\psi_D = \{\psi_{d_i} | d_i \in D\}$ and assumes each target's complete information parameter name is unique.*

Combining the Definitions 2.9 and 2.5 allows us to define *describes* for a target system as $\mathit{describes}(\psi_D, D) \Leftrightarrow \forall\ d \in D, \mathit{describes}(\psi_d, d)$.

The mission planning system (MP) can be viewed as the target $D = \{d_{Planner}, d_{AssetDB}, d_{TargetDB}, d_{GeoDB}\}$ with a system information parameter ψ_{MP} where $\mathit{describes}(\psi_{MP}, MP)$ is true. In addition, since each target device has a complete information parameter ψ_{d_i}, each device also has a complete information parameter such as $\psi_{Planner}$ where $\mathit{describes}(\psi_{Planner}, \mathit{Planner})$. Like the information parameter, the state of the target system is simply a combination of each target device's state.

Definition 2.10. *The* target system state S_D *of target system* D *is the current value of the system information parameter* ψ_D*. We also refer to the state of* D *at time* t*.*

At this point, we have defined key concepts related to targets and target systems, which in some ways overlaps the definitions in MTD Systems Theory. (MTD systems are also target systems, etc.) However, it is important to note that targets are defined from the attacker's perspective, which captures the fact that the attacker may not know the policies and constraints associated with the system. These details are often the objective of preliminary attacks on the system. As we define attackers and attacks, we show how this information can be gained via attacks.

2.3 Attackers

To begin to understand attacks, including how and why they are launched, we must start with the attacker. While an attacker is usually interpreted as an individual person, we extend that notion slightly.

Definition 2.11. *An* attacker*,* x*, represents a single intruder or team of intruders, where an intruder can be either a human or an automated program.*

Thus, an attacker is not limited to an individual person. Attackers may be groups of people where each is responsible for a part of a coordinated attack. Attacker may also be software programs that automate the intrusion process.

As discussed above, for an intrusion to be successful, an attacker must make an effort to investigate the target, which leads to the attacker possessing additional or updated knowledge about the target. We also represent this knowledge as information parameters.

Definition 2.12. *A* knowledge unit *is an information parameter possessed by an attacker. We say attacker* x *possesses knowledge* ψ^x*, which includes all the attacker's knowledge units* x*. If* $\psi_1, \psi_2 \ldots \psi_n$ *are the knowledge units* x *possesses, then* $\psi^x = \{\psi_1, \psi_2 \ldots \psi_n\}$*. Note: attacker's knowledge may not be true; it only represents what the attacker believes to be true.*

Here, we see ψ^x represents all the knowledge an attacker can use to attack a target. If the attacker's knowledge is not sufficient to attack a specific target to achieve the attacker's objective, the attacker will be forced to perform preliminary attacks to gain that knowledge. To specifically talk about an attacker's knowledge about a given target, d, or target system, D, we define ψ^x_d and ψ^x_D.

Definition 2.13. *Attacker* x *has* knowledge*,* ψ^x_d*, about target* d*, where* $\psi^x_d = \{\psi \mid \psi \in \psi^x \wedge \psi \overset{n}{\in} \psi_d\}$*. Similarly, attacker* x*'s knowledge about target system* D *is represented as* ψ^x_D*, where*

$$\psi^x_D = \{\psi \mid \psi \in \psi^x \wedge \psi \overset{n}{\in} \psi_D\} = \bigcup_{d \in D} \psi^x_d.$$

The operator $\overset{n}{\in}$ is used to capture the fact that an attacker's knowledge ψ^x_d and ψ^x_D includes the information parameters that have the *same name* as those in the target's complete information parameter, although their values associated may be different. Formally, we recursively define $\overset{n}{\in}$ as

$$\psi \overset{n}{\in} \psi' \Leftrightarrow \exists\ \psi_i \in \psi'.v \text{ s.t. } (\psi.n = \psi_i.n \vee \psi \overset{n}{\in} \psi_i).$$

Capturing attacker knowledge supports reasoning about what an attacker knows, what information is gained during an attack, etc. This reasoning will be the key to understanding the affect of MTD adaptation on an attacker's attempt to penetrate or compromise specific targets since MTD adaptation works by invalidating attackers' knowledge of their targets. To know when an attacker's knowledge of a target is valid, we define the predicate *holds*.

Definition 2.14. *If during a time period,* $[t_1, t_2]$*, a logical statement,* l*, defined over* ψ^x *and* ψ_D *is true, we say* $holds(l, [t_1, t_2])$*. If* $t_1 = t_2 = t$*, it simplifies as* $holds(l, t)$*.*

For example, an attacker might have the following knowledge about the Planner: $\langle memory, 8GB\rangle$, $\langle cpu, intel\ i5\rangle$, $\langle os, Windows\ 8.1\rangle$, and $\langle ip, 222.20.22.22\rangle$. However, as defined in Section 2.2.1, the true value of the operating system for the Planner is $\langle os, Ubuntu\ 14.04\rangle$ and thus the attacker's knowledge does not *hold*. Obviously, any attacks against the Planner that assumes a Windows OS will fail.

2.4 Attacks

Now that we have defined the concepts of attackers and targets, we turn to defining the attacks themselves. Attacks are a key aspect of our theory as they define the effect of an attacker's interaction with the target system. We define attacks in terms of their affect on system information parameters or attacker knowledge. To help us capture the modification of information, we start by defining an assignment of values between two information parameters.

Definition 2.15. *An* assignment o *is a tuple of information parameters* $\langle\psi_1, \psi_2\rangle$*, that when executed, copies the value of* ψ_2 *into* ψ_1*, which is denoted as* $\psi_1.v \leftarrow \psi_2.v$*. Formally, we define the* execute *operation as* $execute(o) \Leftrightarrow o.\psi_1.v \leftarrow o.\psi_2.v$*.*

Essentially, the execution of an assignment affects only the value associated with the first information parameter. The information parameters themselves do not need to have the same name, even though it is often used to denote the copying of information from a target to an attacker's knowledge or vice versa. From the atttacker's perspective, when an assignment $o = \langle\psi_1, \psi_2\rangle$, where ψ_2 is a target system information parameter and ψ_1 belongs to attacker's knowledge, it is called a *gain* assignment. In contrast, when ψ_2 belongs to an attacker's knowledge and ψ_1 is a target system information parameter, it is called a *modify* assignment, which implies that the attacker has successfully modified the target system, and the attacker's knowledge of the target system is updated accordingly.

For example, a successful intrusion that obtains the IP address of the Planner can be captured by an assignment $\langle\psi^x_{Planner}.ip,\ \psi_{Planner}.ip\rangle$, which sets the Planner's IP address to the address the attacker's has in its IP address information parameter for the Planner. [1] We can also define an intrusion that modifies the system time of the Planner via two assignments $\langle\psi_{Planner}.time,\ \psi^x_t\rangle$, and $\langle\psi^x_{Planner}.time,\ \psi^x.t\rangle$, where the first assignment sets the Planner's system time to $\psi^x.t$ (some time the attacker wants to set the system time to) while the second assignment updates the attacker's own knowledge of the Planner's system time.

[1]Here we use the '.' notation to refer to the ip information parameter in the attackers knowledge about the Planner.

Table 1: Attack Type Specification

Type	Ω_{pre}	Ω_{post}
ϕ_1	$\psi_{d_1}.ip \neq \psi^x_{d_1}.ip$	$\langle \psi^x_{d_1}.ip, \psi_{d_1}.ip \rangle$
ϕ_2	$\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port \neq \psi_{d_1}.port$	$\langle \psi^x_{d_1}.port, \psi_{d_1}.port \rangle$
ϕ_3	$\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port = \psi_{d_1}.port \wedge \psi^x_{d_1}.os \neq \psi_{d_1}.os$	$\langle \psi^x_{d_1}.os, \psi_{d_1}.os \rangle$
ϕ_4	$\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port = \psi_{d_1}.port \wedge \psi^x_{d_1}.os = \psi_{d_1}.os \wedge \psi^x_{d_1}.vul \neq \psi_{d_1}.vul$	$\langle \psi^x_{d_1}.vul, \psi_{d_1}.vul \rangle$
ϕ_5	$\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port = \psi_{d_1}.port \wedge \psi^x_{d_1}.os = \psi_{d_1}.os \wedge \psi^x_{d_1}.vul = \psi_{d_1}.vul$	$\langle \psi_{d_1}.exa, \psi^x.exa \rangle$, $\langle \psi^x_{d_1}.exa, \psi^x.exa \rangle$
ϕ_6	$\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port = \psi_{d_1}.port \wedge \psi^x_{d_1}.exa = \psi_{d_1}.exa \wedge \psi^x_{d_1}.root \neq \psi_{d_1}.root$	$\langle \psi^x_{d_1}.root, \psi_{d_1}.root \rangle$

2.4.1 Attack Types

Next, we use assignments to define the post-conditions of attacks, which are defined over ψ_D and ψ^x. We start by defining an attack type, which is a template for actual attack instances, which we define later.

Definition 2.16. *An* attack type*, ϕ, is a tuple $\langle \Omega_{pre}, \Omega_{post} \rangle$ where Ω_{pre} is a logical statement defined over the target system's information parameter ψ_D and the attacker's knowledge ψ^x, and Ω_{post} is a set of assignments over ψ_D and ψ^x.*

To simplify our discussion, we assume that the logical statements are valid and that the names of the information parameters in ψ^x and ψ_D are unique.

In the mission planning system example, assume attacker x has the goal to exploit the Planner to obtain root privileges. To achieve this objective, x considers a sequence of attack types, $\phi = \{\phi_1, \phi_2, \ldots, \phi_6\}$, where the effects of the attacks are as follows:

- ϕ_1 - gain the IP address of the Planner
- ϕ_2 - gain the port number of a specific app
- ϕ_3 - gain the operating system type
- ϕ_4 - obtain an exploitable vulnerability of the app
- ϕ_5 - deploy an exploit agent on the Planner
- ϕ_6 - connect to the agent (e.g., via reverse shell) and gain the root privilege

Table 1 shows the specification of these attack types. Each attack type's precondition is a logic statement that explicitly reflects the relationship between an attacker's knowledge of the target and the target's true information. Also, notice that each attack type's precondition depends on the previous attack type's post-condition. For example, ϕ_3 requires that the attacker's knowledge about the target's IP address and the port number is correct. This means the attacker must have a way to gain prior knowledge before the actual attack, which is done via the post-conditions of ϕ_1 and ϕ_2. Also note that the post-condition of ϕ_5 can be viewed as a *modify* assignment while $\phi_1, \phi_2, \phi_3, \phi_4$ and ϕ_6 are all *gain* assignments.

This approach not only allows us to explicitly define an attack type based on the relationship between an attacker's knowledge and target system, but it also provides insight into the key information parameters associated with specific attacks and targets. This relationship will be instrumental when we tie Attack Theory to existing MTD System Theory in order to analyze which configuration parameters can be modified to thwart different types of attacks and to formally define the attack surface for specific types of attacks.

We also note that a precondition's logical statement only captures necessary information-based conditions for an attack to succeed. It does not include all the sufficient conditions. We discuss this in more details in Section 2.4.2.

While the attack specifications in Table 1 provide a precise description of individual attacks, attackers generally combine a sequence of low-level attacks to achieve some higher-level objective. To capture this reality, we need to provide the ability to analyze the composition of a set of low-level attack types. However, before defining attack type composition, we define three helper functions: *transform*, *union replace*, and *substitution*.

Definition 2.17. *We define two* transform functions*, ξ_x and ξ_d, that compute a set of information parameters from the assignments in Ω_{post}. $\xi_x()$ extracts information parameters belonging to ψ^x while $\xi_d()$ extracts information parameters that describe target d as defined below.*

$$\xi_x(\phi.\Omega_{post}) = \{\langle n, v\rangle | \langle \psi_1, \psi_2 \rangle \in \phi.\Omega_{post} \wedge \psi_1 \in \psi^x \wedge n = \psi_1.n \wedge v = \psi_2.v\}$$

$$\xi_d(\phi.\Omega_{post}) = \{\langle n, v\rangle | \langle \psi_1, \psi_2 \rangle \in \phi.\Omega_{post} \wedge \psi_1 \in \psi_D \wedge n = \psi_1.n \wedge v = \psi_2.v\}$$

The purpose of the transform function is to extract information parameters contained in Ω_{post}. For example, if we take Ω_{post} from ϕ_1, $\xi_x(\phi_1.\Omega_{post}) = \{\langle ip,\ \psi_{d_1}.ip.v\rangle\}$, and $\xi_d(\phi_1.\Omega_{post}) = \emptyset$. Similarly, for ϕ_5, $\xi_x(\phi_5.\Omega_{post}) = \{\langle Exa, \psi^x.Exa.v\rangle\}$, and $\xi_d(\phi_5.\Omega_{post}) = \{\langle Exa,\ \psi^x.Exa.v\rangle\}$. To define the union replace function, we first define an operator to determine that an information parameter name does not exist in a set of information parameters or assignments. If $\hat{\psi}$ is a set of information parameters, then we recursively define the operator $\overset{n}{\notin}$ as

$$\psi \overset{n}{\notin} \hat{\psi} \Leftrightarrow \nexists\ \psi_i \in \hat{\psi}, \text{ s.t. } (\psi.n = \psi_i.n \vee \psi \overset{n}{\in} \psi_i.v).$$

And when $\hat{o}$ is a set of assignments, $\overset{n}{\notin}$ becomes

$$o \overset{n}{\notin} \hat{o} \Leftrightarrow \nexists\ o_i \in \hat{o}, \text{ s.t. } (o_i.\psi_1.n = o.\psi_1.n \vee o.\psi_1 \overset{n}{\in} o_i.\psi_1.v).$$

Using the $\overset{n}{\notin}$ operator, we now define the union replace function, which updates one set of information parameters based on a second set of information parameters. Essentially, the union replace function replaces the information parameter values in the first set with those of the second set if the names match. Additionally, if information parameters exist in the second set but not the first, these new information parameters from the second set are added to the first set. We also overloaded the union replace operator to work on two sets of assignments as well.

Definition 2.18. *To update one set of information parameters $\hat{\psi_1}$ based on a second set $\hat{\psi_2}$, we define a* union replace *function, $\hat{\psi_1} \uplus \hat{\psi_2} = \{\psi | (\psi \in \hat{\psi_1} \wedge \psi \overset{n}{\notin} \hat{\psi_2}) \vee \psi \in \hat{\psi_2}\}$. Likewise, union replace over assignments is defined as $\hat{o}_1 \uplus \hat{o}_2 = \{o | (o \in \hat{o}_1 \wedge o \overset{n}{\notin} \hat{o}_2) \vee o \in \hat{o}_2\}$*

Next, we define a *substitution* function σ that substitutes the values from a set of information parameters into a logical statement. We use σ to substitute the information parameters values from an attacker's knowledge into the corresponding information parameter in the precondition of a given attack type, ϕ.

Definition 2.19. *Given a logical statement l and a set of information parameters $\hat{\psi}$ we define the* substitution *function $\sigma(l, \hat{\psi})$ as a mapping from names in l to values of information parameters in $\hat{\psi}$ such that the name in l matches the name of the information parameter in $\hat{\psi}$.*

In general, we use the σ function to substitute the values of information parameters in the attacker's knowledge to variable names in Ω_{pre}. This mapping allows us to evaluate the precondition Ω_{pre}.

Using these helper functions, we now formally define a composite attack type. Intuitively, a composite attack type is a sequence of sub attack types. The preconditions of the composite attack type is the conjunction of all the preconditions from the sub attack types that are not satisfied by previous sub attacks. Likewise, the post-condition is the union of the sub attack post- conditions where an assignment to an information parameter later in the sequence takes precedence over assignments to the same information parameter earlier in the sequence.

Definition 2.20. *A* composite attack type, *ϕ, is a sequence of attack types, $\phi = [\phi_1, \phi_2, \ldots, \phi_n]$, where each sub attack type's pre and post-conditions ($\phi_i.\Omega_{pre}$ and $\phi_i.\Omega_{post}$) are defined over a target system's complete information parameter $\psi_{D(i)}$ and the attacker's knowledge $\psi^{x(i)}$. The composite attack type's pre and post-conditions are defined as*

$$\begin{aligned}
\phi.\Omega_{pre} &= \bigwedge_{1 \le i \le n} \sigma(\phi_i.\Omega_{pre}, \psi^{x(i-1)}) \\
\phi.\Omega_{post} &= \biguplus_{1 \le i \le n} \phi_i.\Omega_{post} \\
where: & \ \psi_{D(0)} = \{\}, \psi^{x(0)} = \{\} \\
& \ \psi^{x(i)} = \psi^{x(i-1)} \uplus \xi_x(\phi_i.\Omega_{post}) \\
& \ \psi_{D(i)} = \psi_{D(i-1)} \uplus \xi_x(\phi_i.\Omega_{post})
\end{aligned}$$

Note, we use a sequence above to reflect the relationship that the subsequent attack types depend on previous attack types. While not specifically defined, there is nothing in our theory to limit the analysis of parallel attacks. For completeness, we define an *atomic attack type* as an attack type that cannot be further decomposed into sub attack types. To demonstrate how pre and post-conditions for composite attack types are computed, we show how are ϕ_1 and ϕ_2 are composed into attack type ϕ in Figure 4.

Generally speaking, an attack type acts as a template for an actual attack. This relationship is similar to that of an object-oriented class and an object or instance of that class. One attack type can be implemented by many different attacks. For example, attack type ϕ_1 attempts to gain the IP address of the Planner. To implement ϕ_1, an attacker might use automated IP scanning tools, guess the IP address, or obtain it through social engineering. Although these are different attacks, they all implement a same attack type.

2.4.2 Attack Instances

Definition 2.21. *An* attack *is a process performed by attacker x against target d implementing attack type ϕ during time period $[t_s, t_f]$. We denote this attack as $\oint_{t_s}^{t_f}(x, d, \phi)$. Each attack, has a success likelihood against static systems of P_{static} and a duration of $T_a = t_f - t_s$.*

As indicated in [3, 18], quantifying the effectiveness of MTD systems is still an open issue, however, the development of an attack theory will greatly benefit our understanding of the interaction between the attacker and MTD system. This will include an understanding of the cost factors

Initialization:

$$\psi^x(0)\{\}, \psi_{D(0)} = \{\}, \Omega_{pre} = \{\}, \Omega_{post} = \{\}$$

compose ϕ_1 :

$$\begin{aligned}
\psi^x(1) &= \psi^x(0) \uplus \xi_x(\phi_1.\Omega_{post}) \\
&= \{\} \uplus \{\langle ip, \psi_{d_1}.ip \rangle\} \\
&= \{\langle ip, \psi_{d_1}.ip \rangle\} \\
\psi_D(1) &= \psi_D(0) \uplus \xi_D(\phi_1.\Omega_{post}) \\
&= \{\} \uplus \{\} \\
&= \{\} \\
\Omega_{pre} &= \sigma(\phi_1.\Omega_{pre}, \psi^x(0)) \\
&= \sigma(\phi_1.\Omega_{pre}, \{\}) \\
&= \psi_{d_1}.ip \neq \psi^x_{d_1}.ip \\
\Omega_{post} &= \{\} \uplus \phi_1.\Omega_{post} \\
&= \{\langle \psi^x_{d_1}.ip, \psi_{d_1}.ip \rangle\}
\end{aligned}$$

compose ϕ_2 :

$$\begin{aligned}
\psi^x(2) &= \psi^x(1) \uplus \xi_x(\phi_2.\Omega_{post}) \\
&= \{\langle ip, \psi_{d_1}.ip \rangle\} \uplus \{\langle port, \psi_{d_1}.port \rangle\}) \\
&= \{\langle ip, \psi_{d_1}.ip \rangle, \langle port, \psi_{d_1}.port \rangle\} \\
\psi_D(2) &= \psi_D(1) \uplus \xi_D(\phi_2.\Omega_{post}) \\
&= \{\} \uplus \{\} \\
&= \{\} \\
\Omega_{pre} &= \sigma(\phi_2.\Omega_{pre}, \psi^x(1)) \wedge \Omega_{pre} \\
&= \sigma((\psi^x_{d_1}.ip = \psi_{d_1}.ip \wedge \psi^x_{d_1}.port \neq \psi_{d_1}.port), \\
&\qquad \{\langle ip, \psi_{d_1}.ip \rangle\}) \wedge \Omega_{pre} \\
&= \psi^x_{d_1}.port \neq \psi_{d_1}.port \wedge \psi_{d_1}.ip \neq \psi^x_{d_1}.ip \\
\Omega_{post} &= \Omega_{post} \uplus \phi_2.\Omega_{post} \\
&= \{\langle \psi^x_{d_1}.ip, \psi_{d_1}.ip \rangle, \langle \psi^x_{d_1}.port, \psi_{d_1}.port \rangle\}
\end{aligned}$$

Figure 4: Composition of attack types ϕ_1 and ϕ_2 into attack type ϕ.

related to the attacker and MTD actions, which we believe are closely tied to the duration of the attacks and the impact on the attacker's intrusion success likelihood. Other cost factors, such as attacker effort, are directly related to the time and intrusion success likelihood.

However, explicitly including T_a and P_{static} in the definition of the attack does not necessarily mean that we will assign specific values to them. Coming up with real numbers for these factors is hard [1], although there has been work trying to estimate the mean time-to-compromise [8] and to measure P_{static} [4]. Quantifying T_a and P_{static} is out of the scope of our work. However, we do believe that T_a and P_{static} can be impacted by MTD designers by manipulating MTD system parameters such as the diversification of the configuration space and the adaptation interval. One of our future goals is the development of an analytical model that can inform designers as to how particular parameter settings will impact the effectiveness given attack parameters such as T_a and P_{static}. Conversely, MTD designers will also be able to judge how effective a given MTD system will be based on various values of T_a and P_{static}.

Definition 2.22. *An* atomic attack *is an attack that implements an* atomic attack type *and cannot be decomposed into sub attacks. An* atomic attack *$\oint_{t_s}^{t_f}(x, d, \phi)$ is successful with probability P_{static} if and only if its precondition Ω_{pre} is true from t_s to t_f. If successful $\oint_{t_s}^{t_f}(x, d, \phi)$ ensures $execute(\Omega_{post})$ is true precisely at t_f.*

If P_{static} is true, that indicates that all the sufficient conditions for the attack to be successful in a static system, with the exception of those specified in Ω_{pre} are true. However, Ω_{pre} captures those necessary conditions that can be impacted by the MTD system. As long as Ω_{pre} remains true from t_s to t_f and P_{static} is true, the attack will be successful and the attack's post-conditions will be executed at t_f.

Like attack types, attacks themselves are generally composed of a sequence of smaller attacks to achieve a larger purpose. We now formally define a composite attack.

Definition 2.23. *An* composite attack *is an attack that implements a composite attack type. Given composite attack type ϕ that is composed of a sequence of attack types $[\phi_1, \phi_2, \ldots, \phi_n]$, a composite attack $\oint_{t_s}^{t_f}(x, d, \phi)$ that implements ϕ is composed of a sequence of attacks where each attack $\oint_{t_{i-1}}^{t_i}(x, d, \phi_i)$ implements ϕ_i and $t_0 = t_s \wedge t_n = t_f$. Formally, this is captured as:*

$$\oint_{t_s}^{t_f}(x, d, \phi) = \\ [\oint_{t_s}^{t_1}(x, d, \phi_1),\ \oint_{t_1}^{t_2}(x, d, \phi_2),\ \ldots,\ \oint_{t_{n-1}}^{t_f}(x, d, \phi_n)]$$

Thus, a composite attack is simply implemented by a sequence of attacks where each attack implements a corresponding sub attack type. Using the attacks defined in Table 1, an attack $\oint_{t_s}^{t_f}(x, d_1, \phi)$ that implements the composite attack type $\phi = [\phi_1, \phi_2]$ requires the composition of two sub attacks $\oint_{t_s}^{t_1}(x, d_1, \phi_1)$ that implements ϕ_1 and $\oint_{t_1}^{t_f}(x, d_1, \phi_2)$ that implements ϕ_2.

2.5 Exploration Space

So far, we have introduced two properties of attacks, beside the attack type definition itself, that are critical to analyzing attacks, the attack interval T_a and the static likelihood of success P_{static}. Next, we introduce a third concept that is important to the analysis of attacks and their interactions with MTD systems called the exploration space. Essentially, the exploration space captures the set of possible values an attacker must search in order to find the correct value of a specific information parameter or parameters in order to carry out specific attacks.

Figure 5 shows an overview of the relationships between an information parameter, ψ's, exploration space, its *configuration space* (as discussed in Section 1.2), and the attacker's effort to ascertain ψ's actual value. [2] The effort spent on gaining knowledge through preliminary attacks can be viewed as actions that reduce the attacker's uncertainty about ψ's value from the exploration space down to a single value. For a static system, this uncertainty can be safely assumed to monotonically decreasing with each additional attack. However, with MTD systems, this assumption is invalid. Instead, MTD systems make the attacker's uncertainty non-monotonic.

An exhaustive search of the entire exploration space Ψ to identify the correct value of ψ is not the preferred approach. However, there are times when an exhaustive approach are applicable. For example, in the mission planning example, an attack implementing ϕ_1 may scan all possible IP addresses in an IPv4 subnet to obtain the correct IP address of the Planner. However, attackers can also use *a priori* knowledge to reduce the search space as well. For example, while knowing that port numbers must be in the range of 0-65535 is of some use in searching $\psi_{Planner}$ for the website port number, knowledge that public facing websites usually use port number 80 may immediately reveal the correct value of $\psi_{Planner}.port$. An attacker can use social engineering to gain required knowledge. For example, an administrator might be fooled into leaking important system information such as IP addresses, operating systems, passwords, etc. No matter which approach is leveraged by attackers, gaining knowledge definitely requires effort on their part to reduce the size of the exploration space.

[2]If the information parameter in question is a target's complete information parameter or any other set of information parameters the values are simply tuples of values corresponding to the information parameters in the set.

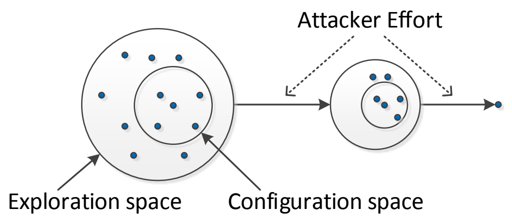

Figure 5: Exploration Space Overview (dots are possible values of the information parameter)

If an information parameter ψ is a configuration parameter of the MTD system as well, the exploration space of ψ may actually be larger than configuration space of ψ. Although a configuration parameter's valid values are typically limited based on system constraints and policies, attackers usually have no way of knowing what these constraints are. For example, constraints internal to the system may require $\psi_{AssetDB}.port$ to be either 43, 53, or 63. However, since attackers would not typically know this information, they would likely be forced to scan the entire range from 0 to 65535. Thus, in general, the exploration space of information parameter ψ equals its domain ψ.

Definition 2.24. *Given information parameter ψ with domain Ψ, the* exploration space *of ψ is*

$$ESpa_{\psi} = \Psi$$

and the size *of the exploration space is $|\Psi|$.*

Generally speaking, the exploration space is the maximum set of potential values attackers need to investigate in order to obtain the correct value. This concept itself is objective and comes with no specific assumptions about an attacker's capability, skill level, or knowledge about the related target system.

Similarly, the exploration space of a target system, D, is simply the domain of Ψ_D, which is the cross product of all ψ_D's sub information parameter domains.

Definition 2.25. *Given a target system D, with its complete information parameter ψ_D with domain of Ψ_D, the* exploration space *of D is defined as:*

$$ESpa_D = \Psi_D$$

Similarly, the size of this exploration space is $|\Psi_D|$.

Although theoretically, this definition gives us an intuition about the exploration space of a target system, it provides little insight to help us understand the exploration space for each individual attack. To do that, we need to define the exploration space of both atomic and composed attack types. To facilitate these definitions, we first define a function to extract the information parameters in an attack type whose value must be gained by an attacker.

Definition 2.26. *Given atomic attack type* $\phi = \langle\Omega_{Pre}, \Omega_{Post}\rangle$*, the function* δ *extracts all information parameters from* Ω_{Post} *whose value the attacker must gain. Formally, this is defined as*

$$\delta(\Omega_{Post}) = \{\psi | \forall o \in \Omega_{Post}, o.\psi_1 \in \psi^x \wedge o.\psi_2 \in \psi_D \wedge \psi \leftarrow o.\psi_1\}$$

Based on this function, we define the exploration space of an attack type.

Definition 2.27. *The* exploration space *of attack type* $\phi = \langle\Omega_{Pre}, \Omega_{Post}\rangle$ *is the cross product of the domain of each information parameter* $\psi \in \delta(\phi.\Omega_{Post})$*.*

$$ESpa_\phi = \prod_{\psi \in \delta(\phi.\Omega_{Post})} \Psi$$

The size of this exploration space is $\prod_{\psi \in \delta(\phi.\Omega_{Post})} |\Psi|$*.*

This definition requires that for an atomic attack type ϕ, if there are multiple information parameters in $\delta(\phi.\Omega_{Post})$, attacks implementing ϕ must attempt to gain the value of each of those information parameter simultaneously. Thus, the exploration space is the cross product of its information parameter domains. However, atomic attack types tend to be very simple and usually only attempt to gain the value of a single information parameter.

Because a composite attack is made of a set of sub attacks, one might think that the exploration space of the composite attack would not be as large as the cross product of all the information parameters for which it attempts to gain a value. However, this is untrue. Although the exploration space shrinks as sub attacks are successfully completed, as shown in Figure 5 as *attacker effort*, the overall exploration space remains the same.

Actually, the concept of sub attacks clearly demonstrates the argument that an attacker's effort is actually linear [1] instead of exponential as would be suggested by the cross product operation. Instead of finding all information parameter values simultaneously, a composite attack breaks that down into a series of steps, whose effort is generally small. Thus, if no changes occur to the system configuration, each atomic attack type can be viewed as an attempt to break into a single layer of defense. And, because the effort required to break into each layer is relatively low, once a layer is penetrated, the next layer is exposed and the attacker has almost unlimited time to attack it. Bellovin [1] claims that what is really needed for system security is an approach that makes the effort expended by the attacker exponential as opposed to linear. Clearly, by constantly adapting the values of the appropriate information parameters, MTD systems could eliminate that brittleness. Attackers can no longer assume that they can ascertain the value of each information parameter one at a time, but will effectively need to learn, and potentially relearn them all in a very short time frame, which pushes the attackers effort towards the exponential.

3. ASLR ATTACK EXAMPLE

This section describes an attack that defeats the PaX Address Space Layout Randomization (ASLR) [16]. Specifically, we use the *return-to-libc* attack presented in [13] to demonstrate how cyber attack theory can be used to analyze a concrete attack. ASLR is a security technique that guards against code reuse attacks, which work by overwriting memory locations to point to potentially malicious code. By randomizing memory locations, ASLR makes it difficult to correctly guess the memory locations of specific processes. More specifically, a process's address space contains three areas: the *executable* area, the *mapped* area and the *stack* area. Instead of fixing each area's base address, ASLR randomizes it by adding an extra variable to the base address when the process is created. For the Intel x86 architecture, PaX ASLR randomizes 16 or 24 bits for these areas. For instance, the mapped data area variable *delta_mmap* has 16 bits randomness, which means the attacker only needs to iterate from 0 to 65535 to determine its value.

We briefly review a concrete implementation of return-to-libc attack [13] before demonstrating the application of cyber attack theory to this example. The return-to-libc attack takes advantage of two aspects of PaX ASLR. First, PaX ASLR randomizes only the base addresses of the three memory areas but not the *layout* within each area. Second, the *layout* is fixed throughout a process and all its children's lifetime. The implementation of the return-to-libc attack first creates a memory hole in the Oracle 9 PL/SQL Apache module by creating an overflow buffer in the *ap_getline()* function in *http_protocol.c.* To conduct the attack, the base of the mapped area *mmap_base* and the offset of the *usleep()* function *usleep_offset* in *libc* are precomputed. (*libc* is the standard C-language library that is loaded into all Unix programs.). Then the value of *delta_mmap* is found by repeatedly overflowing the stack buffer with guesses for the absolute address of the *usleep()* function. An unsuccessful guess causes the child process to crash and be replaced by a new process with the same randomization offsets. A successful guess calls the *usleep()* function and hangs the connection for 16 seconds, which helps determine the value of *delta_mmap.* Once the value of *delta_mmap* is gained, the absolute locations of all functions in *libc* can be calculated. The final step is to smash the stack to point to another *libc* function, *system()*, which executes user supplied commands through command shell. Shell commands are sent to *system()* as an argument.

3.1 Applying Cyber Attack Theory

This section demonstrates how the ASLR attack can be formally described by our cyber attack theory.

Target System. For our purposes we assume the mission planning system is the target system D, with its complete information parameter $\psi_{MissionPlanning}$, and the Planner $d_{Planner}$ (or d_P for short) is the specific target of interest, which has its complete information parameter $\psi_{Planner}$. The information parameters of interests involved in this example are $\psi_{d_P \cdot ip}$, $\psi_{d_P \cdot apache\_port}$, $\psi_{d_P \cdot os}$, $\psi_{d_P \cdot mmap\_base}$, $\psi_{d_P \cdot usleep\_offset}$, $\psi_{d_P \cdot system\_offset}$, and $\psi_{d_P \cdot delta\_mmap}$.

Attacker. To carry out this attack successfully, the attacker x must have correct knowledge of the Planner, $\psi^x_{d_P}$, including the Planner's IP address, Apache web server port number, operating system, the *usleep()* and *system()* function offsets, and the values for *mmap_base* and *delta_mmap.* Formally, these are captured via attacker knowledge in the form of information parameters such as $\psi^x_{d_P}.ip$. In addition, the attacker should have general knowledge in ψ^x that captures special skills such as how to use *mmap*() under linux to obtain mapped area base locations, how to use *objdump* to gain the offsets of *usleep*() and *system*() functions, and how iterate over all potential *delta_mmap* values and judge which one is correct, etc.

Attack Type. To simplify our discussions, we assume the attacker has knowledge about the Planner's IP address, Apache port number and operating system via preliminary attacks. Formally stated using the IP address as an example, in a static system $holds(\psi^x_{d_P \cdot ip} = \psi_{d_P \cdot ip}, [t_s, t_f])$ is assumed to be true. Thus, the return-to-libc attack is a concrete implementation of attack type ϕ_4 in the mission planning scenario. Actually, ϕ_4 can be decomposed into three sub attack types as shown in Table 2. Note that, in ϕ_4's sub attack type preconditions, $\psi^x_{d_P \cdot ip} = \psi_{d_P \cdot ip} \wedge \psi^x_{d_P \cdot apache\_port} = \psi_{d_P \cdot apache\_port} \wedge \psi^x_{d_P \cdot os} = \psi_{d_P \cdot os}$ has been removed due to our assumption of attacker foreknowledge. The return-to-libc attack can also be viewed as a concrete implementation of ϕ_5, but we ommit that discussion due to space limitations.

Attack Instance. As discussed above, the return-to-libc attack implements the attack type ϕ_4. Based on ϕ_4's decomposition, it also precisely captures the knowledge that an attacker x must gain from the target system.

Analysis. Formalizing attacks using cyber attack theory explicitly reveals that many information parameters in the return-to-libc attack remain static and only *delta_mmap* changes. Additionally, *delta_mmap* only has 16 bit randomness, which means its diversity (the size of its configuration/exploration space) is only $2^{16} = 65,536$. The combination of static information parameters with a single dynamic information parameter leads to a limited exploration space and thus supports the paper's conclusion that 32 bit ASLR is not effective for the return-to-libc attack [13]. A switch to a 64 bit architecture would increase the diversity of *delta_mmap* to $2^{32} = 4,294,967,296$, increasing the effectiveness of ASLR. However, we should note that making static information parameters dynamic with sufficient diversity would do more to significantly increase the overall diversity of the system, even for 32 bit architectures.

4. DISCUSSION AND FUTURE WORK

This paper represents a second step toward our goal of defining a complete theory for Moving Target Defenses. As discussed previously [3, 18] the implementation of an effective MTD mechanism only makes sense in the context of a specific threat model. The cyber attack theory presented herein encourages MTD designers and researchers to formally specify and compose attack types to discover the exact system information parameters of interest and how they interact with specific attacks, the attacker's knowledge, and the target system. Specification and analysis of attacks will enable MTD designers to potentially ignore unrelated information parameters while focusing on those most critical, dramatically limiting the scope and cost of MTD systems without sacrificing effectiveness.

Although the relationship between attackers and targets and configuration changes and their effect on various attacks type is becoming evident, we have yet to formally link cyber attack theory to MTD system theory. Our next step is to formally define the relationships between these two theories to produce a complete theory of MTD, which will support formal specification of the interplay between high level attackers and system goals as well as the impact of different types of adaptations on various types of attacks.

In [18], we motivated the need for a new definition of attack surface for MTD systems, which we have yet to define. However, it seems clear that the attack surface must be related to the information parameters used by potential attack types as well as the the target system itself. Clearly, if an information parameter in the attack surface is a configuration parameter of the MTD system, it is highly likely that it should be a candidate for diversification and randomization via the MTD system.

In Section 2.5 we showed how, when attempting to gain the value of a set of information parameters, an MTD systems can force attackers to expend an exponential effort as opposed to the linear effort required in static systems [1]. The key to this result in MTD systems is forcing attackers to continually go back and regain information parameter values that have been changed by the MTD system. However, this result also underlies the need to know exactly which configuration parameters are the most important to disrupting attacks. This can only be determined by formally capturing the possible MTD configuration parameters and the attacks they are attempting to stop.

Hobson *et,al.* [3] claim that measuring unpredictability in MTD systems is mathematically possible if both the threat and the attack surface can be adequately quantified. In [17], we proposed an analytical model for analyzing the effectiveness of MTD systems that, when combined with MTD System Theory and Cyber Attack Theory, can be enhanced to provide a more fine grained model to show the interaction between key MTD parameters such as T_a, P_{static}, the attack surface, the configuration space and adaptation interval. The model will allow MTD designers to see how different parameter settings will impact security in terms of intrusion success likelihood and would be a powerful tool that will allow MTD designers to make trade-off decisions when implementing such systems.

5. RELATED WORK

Various approaches to modeling attacks and threats have been proposed within different areas of security. For instance, in knowledge sharing, Moore *et al.* [9] emphasize the need to better learn from previous attack data. Thus they propose a structured and reusable way to document information- security attacks that will allow security analysts to identify commonly occurring patterns derived from real attack data. On the other hand, Steffan and Schumacher [15] propose a method that combines a graph- based attack modeling technique (attack net) with a Web- based collaboration tool (WikiWeb) to improve knowledge sharing and collaboration between security experts.

Schneier [12] advances the idea of attack trees, a formal, methodical way of describing the security of systems, based on varying attacks. An attack is represented in a tree structure, whose root is the attack goal and whose leaves branches are different ways of achieving that goal. Further, Çamtepe and Yener [2] propose a formal methodology, based on attack trees, for network attack modeling and detection. They extend the attack trees to include attack expiration time and temporal dependencies between components.

In the attack graph space, Jajodia *et al.* [6] proposed an innovative approach to proactive cyber security via attack graphs called Topological Vulnerability Analysis, which combines vulnerabilities as real attackers might, uncovering all attack paths through a network based on scanning data. Further, to determine the security impact of software vulnerabilities on a network, Ou *et al.* [11] introduced Mul-

Table 2: Attack Type ϕ_4 Decomposition

Type	$\mathbf{\Omega}_{pre}$	$\mathbf{\Omega}_{post}$
$\phi_{4.1}$	$\psi_{d_P} \cdot mmap\_base \neq \psi^x_{d_P} \cdot mmap\_base$	$\langle \psi^x_{d_P} \cdot mmap\_base, \psi_{d_P} \cdot mmap\_base \rangle$
$\phi_{4.2}$	$\psi^x_{d_P} \cdot mmap\_base = \psi_{d_P} \cdot mmap\_base \wedge \psi^x_{d_P} \cdot usleep\_offset \neq \psi_{d_P} \cdot usleep\_offset$	$\langle \psi^x_{d_P} \cdot usleep\_offset, \psi_{d_P} \cdot usleep\_offset \rangle$
$\phi_{4.3}$	$\psi^x_{d_P} \cdot mmap\_base = \psi_{d_P} \cdot mmap\_base \wedge \psi^x_{d_P} \cdot usleep\_offset = \psi_{d_P} \cdot usleep\_offset \wedge \psi^x_{d_P} \cdot delta\_mmap \neq \psi_{d_P} \cdot delta\_mmap$	$\langle \psi^x_{d_P} \cdot delta\_mmap, \psi_{d_P} \cdot delta\_mmap \rangle$

VAL, an end-to-end framework and reasoning system that conducts multi- host, multi-stage vulnerability analysis on a network. Moreover, Ingols *et al.* [5] proposed NetSPA, which can rapidly build a multiple-prerequisite graph that enables defenders to quickly evaluate their network's security. More recently, Kordy *et al.* [7] published a comprehensive survey on attack and defense modeling approaches that are based on directed acyclic graphs (DAGs).

While some of the proposed approaches can complement our work, this paper defines key concepts that support a precise discussion of attacker knowledge, attack goals, attack types, and attack instances in the context of a dynamic target system. These concepts will allow us to precisely compare and contrast approaches such as those discussed.

6. CONCLUSIONS

This paper presents a theory of cyber attacks, a critical step towards understanding and analyzing moving target defenses. We introduced a new concept called *information parameters* that supported the information- based definitions of target systems and attacker's knowledge. We also used information parameters to define the concepts of attack type, attack instance, and exploration space. To enhance the understanding of our theory, we presented a mission planning system scenario and showed how cyber attack theory can support the analysis of such attacks using a real world *return-to-libc* attack example.

Acknowledgments. This work was supported by the U.S. Air Force Office of Scientific Research (FA9550-12-1-0106) and National Science Foundation (0954138, 1018703). Opinions, findings, conclusions or recommendations are the author's and do not necessarily reflect the agencies' views.

7. REFERENCES

[1] S. M. Bellovin. On the brittleness of software and the infeasibility of security metrics. *Security & Privacy, IEEE*, 4(4):96–96, 2006.

[2] S. A. Çamtepe and B. Yener. A formal method for attack modeling and detection. *SA Camtepe, B. Yener*, 2006.

[3] T. Hobson, H. Okhravi, D. Bigelow, R. Rudd, and W. Streilein. On the challenges of effective movement. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 41–50. ACM, 2014.

[4] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *Journal of Computer Security*, 21(4):561–597, 2013.

[5] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, 2006.

[6] S. Jajodia and S. Noel. Advanced cyber attack modeling, analysis, and visualization. Technical report, George Mason University, Mar. 2010.

[7] B. Kordy, L. Pietre-Cambacedes, and P. Schweitzer. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *CoRR*, abs/1303.7397, 2013.

[8] D. J. Leversage and E. James. Estimating a system's mean time-to-compromise. *Security & Privacy, IEEE*, 6(1):52–60, 2008.

[9] A. P. Moore, R. J. Ellison, and R. C. Linger. Attack modeling for information security and survivability. Technical report, CMU/SEI Report Number: CMU/SEI-2001-TN-001, Mar. 2001.

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

[11] X. Ou, S. Govindavajhala, and A. W. Appel. Mulval: A logic-based network security analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium*, 2005.

[12] B. Schneier. Attack trees. `https://www.schneier.com/paper-attacktrees-ddj-ft.html`, 1999.

[13] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 298–307. ACM, 2004.

[14] M. P. Singh. Towards a science of security. `http://www.computer.org/portal/web/computingnow/archive/january2013`, 2013. Online, accessed June 30, 2014.

[15] J. Steffan and M. Schumacher. Collaborative attack modeling. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, 2002.

[16] P. Team. PaX address space layout randomization (ASLR), 2003.

[17] R. Zhuang, S. A. DeLoach, and X. Ou. A Model for Analyzing the Effect of Moving Target Defenses on Enterprise Networks. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 73–76. ACM, 2014.

[18] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40. ACM, 2014.