# A Survey of Moving Target Defenses for Network Security

Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Dijiang Huang, and Subbarao Kambhampati

***Abstract*—Network defenses based on traditional tools, techniques, and procedures (TTP) fail to account for the attacker's inherent advantage present due to the static nature of network services and configurations. To take away this asymmetric advantage, Moving Target Defense (MTD) continuously shifts the configuration of the underlying system, in turn reducing the success rate of cyberattacks. In this survey, we analyze the recent advancements made in the development of MTDs and highlight (1) how these defenses can be defined using common terminology, (2) can be made more effective with the use of artificial intelligence techniques for decision making, (3) be implemented in practice and (4) evaluated. We first define an MTD using a simple and yet general notation that captures the key aspects of such defenses. We then categorize these defenses into different sub-classes depending on *what* they move, *when* they move and *how* they move. In trying to answer the latter question, we showcase the use of domain knowledge and game-theoretic modeling can help the defender come up with effective and efficient movement strategies. Second, to understand the practicality of these defense methods, we discuss how various MTDs have been implemented and find that networking technologies such as Software Defined Networking and Network Function Virtualization act as key enablers for implementing these dynamic defenses. We then briefly highlight MTD test-beds and case-studies to aid readers who want to examine or deploy existing MTD techniques. Third, our survey categorizes proposed MTDs based on the qualitative and quantitative metrics they utilize to evaluate their effectiveness in terms of security and performance. We use well-defined metrics such as risk analysis and performance costs for qualitative evaluation and metrics based on Confidentiality, Integrity, Availability (CIA), attack representation, QoS impact, and targeted threat models for quantitative evaluation. Finally, we show that our categorization of MTDs is effective in identifying novel research areas and highlight directions for future research.**

***Index Terms*—Cyber security, network security, moving target defense, artificial intelligence, cyber deception, game theory, attack representation methods (ARMs), cyber kill chain (CKC), advanced persistent threats, software-defined networking (SDN), network function virtualization (NFV), qualitative metrics, quantitative metrics, risk analysis, QoS metrics.**

Manuscript received April 30, 2019; revised October 24, 2019; accepted February 13, 2020. Date of publication March 26, 2020; date of current version August 21, 2020. This work was supported in part by the Naval Research Laboratory under Grant N00173-15-G017 and Grant N0017319-1-G002, in part by the Air Force Office of Scientific Research under Grant FA9550-18-1-0067, in part by the National Aeronautics and Space Administration under Grant NNX17AD06G, in part by the Office of Naval Research under Grant N00014-16-1-2892, Grant N00014-18-1-2442, and Grant N00014-18-12840, in part by NSF U.S. under Grant DGE-1723440, Grant OAC-1642031, Grant SaTC-1528099, and Grant 1723440, in part by NSF China under Grant 61628201 and Grant 61571375, in part by the JP Morgan AI Research Faculty Award, and in part by DARPA CHASE under Grant W912CG-19-C-0003 (via IBM). The work of Sailik Sengupta was supported by the IBM Ph.D. Fellowship. The work of Abdulhakim Sabur was supported by a scholarship from Taibah University through Saudi Arabian Cultural Mission. *(Sailik Sengupta and Ankur Chowdhary contributed equally to this work.) (Corresponding author: Sailik Sengupta.)*

Sailik Sengupta, Ankur Chowdhary, Dijiang Huang, and Subbarao Kambhampati are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: ssengu15@asu.edu; achaud16@asu.edu; dijiang@asu.edu; rao@asu.edu).

Abdulhakim Sabur is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA, and also with Taibah University, Medina 42353, Saudi Arabia (e-mail: asabur@asu.edu).

Adel Alshamrani is with the Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia (e-mail: asalshamrani@uj.edu.sa).

Digital Object Identifier 10.1109/COMST.2020.2982955

## I. Introduction

The NETWORK and cloud infrastructures have become both ubiquitous and more complex in the past few years. Gartner predicts that by the year 2025, 80% of the entire IT infrastructure which includes deployed applications, technologies and services will be cloud-based [1]. While the performance aspects with regards to storage capacity, networking efficiency, and hardware have received due attention and evolved with business demands, aspects that govern the security of cloud infrastructure are still managed using traditional means. Given that security breaches can lead to loss of customer trust and worsen business reputation, a key question is *how effective are these traditional security approaches?* Is monitoring network traffic for malicious patterns, routinely patching known vulnerabilities, and relying on the network and perimeter defense such as Firewalls, Intrusion Detection Systems, Anti-malware tools, *etc.* enough to detect and thwart attacks?

There exist multiple shortcomings in these traditional defense mechanisms. First, the attacker, with time on their side, can spend reconnaissance effort in modeling the cloud system (the defenses in place) and then, carefully plan their attacks. Second, implementations of these defenses in practice are often far from ideal, thereby allowing attackers even more opportunities to exploit the system. A report from 2016 predicts that by the end of 2020, 99% of the vulnerabilities exploited will be known to security and IT professionals since a year ago [2]. A major reason for this is the time and complexity associated with the process of routinely patching vulnerabilities. Often, the fear of degradation in the Quality of Service (QoS) provided to customers deters Cloud Service Providers (CSPs) from making changes to the existing system configuration. Third, *zero-day* attacks developed using the information the attacker gathers during

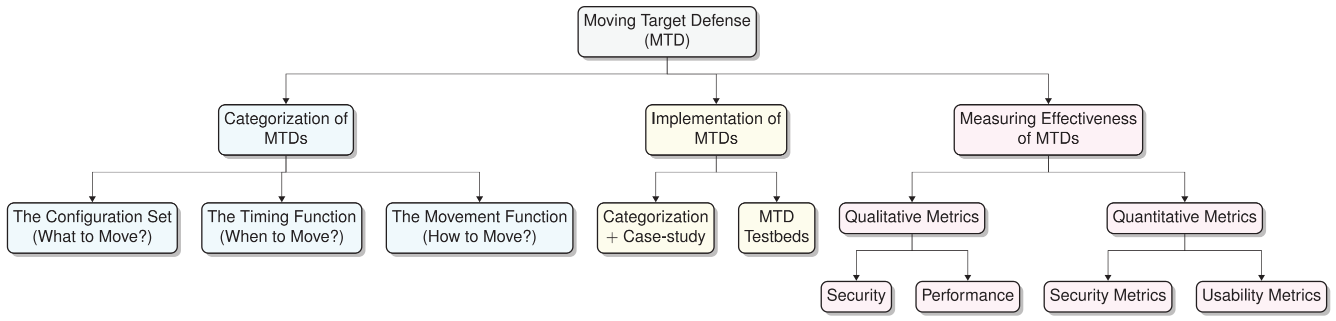

Moving Target Defense (MTD)
Categorization of MTDs
Implementation of MTDs
Measuring Effectiveness of MTDs
The Configuration Set (What to Move?)
The Timing Function (When to Move?)
The Movement Function (How to Move?)
Categorization + Case-study
MTD Testbeds
Qualitative Metrics
Quantitative Metrics
Security
Performance
Security Metrics
Usability Metrics

Fig. 1. We survey various defense techniques based on the paradigm of Moving Target Defense (MTD) and showcase that a common terminology and directions for future work naturally emerges due to the categorization in our survey.

the reconnaissance phase can render traditional defenses useless.

To address the shortcomings of existing defense methods, *Moving Target Defense* (MTD) has emerged as a solution that provides *proactive defense* against adaptive adversaries. The goal of MTD is to constantly move between multiple configurations in a cyber-system (such as changing the open network ports, network configuration, software, *etc.*) thereby increasing the uncertainty for the attacker; in effect, diminishing the advantage of reconnaissance that an attacker inherently has against traditional defense mechanisms. The advantages of MTDs go away if the shifting mechanism is deterministic because the attacker, with time on their side, will eventually be able to predict this movement and design attacks accordingly. Thus, for MTDs to be effective, they need to have implicit randomness built into them. This survey categorizes MTDs based on what they shift, when they shift and how they shift.

The dynamic aspect of MTD adds an extra layer of complexity in implementing these defenses. To address this, one can leverage advances in networking technology. First, *Network Functions Virtualization* (NFV) [3] has emerged as a technology to provide a virtualized implementation of hardware-based equipment such as firewalls, routers, *etc.* through *Virtual Machines* (VMs) or containers running on top of a physical server in cloud computing environments. Given MTDs, often, need more hardware than conventional software systems, such virtualization helps to reduce the cost of implementation. Second, *Software-Defined Networking* (SDN) [4], which serves as an enabling technology for NFV, provides a centralized security policy enforcer. The programmable interface afforded by SDN can help administrators implement optimal movement strategies for MTDs. Futhermore, enabling MTD implementation at scale will help us evaluate the effectiveness of these defenses in practical settings.

The key contributions of this survey are: (1) provides an umbrella under which we can categorize the array of MTD techniques proposed for network environments, (2) introduces a common language that can be utilized to describe (and understand) the assumptions and threat models of various MTDs, (3) gives an overview of how these defenses are implemented by researchers, highlighting testbeds and case studies to guide its large-scale deployment and (4) discusses how MTDs have been evaluated from a qualitative and a quantitative standpoint, in effect, shedding light on how effective these tools and techniques are with regards to security and performance. Figure 1 gives a quick overview of this survey.

The rest of the survey is organized in the following manner. In Section II, we introduce the reader to some background knowledge about the various stages of an attack in cloud systems, popular for detections and defenses against malicious traffic, and formal frameworks to capture knowledge about attacks in networking systems. In Section III, we propose a universal notation that captures the key aspect of all the MTDs proposed and use it to investigate and categorize the defenses depending on how they answer the questions (1) *what to move*, (2) *when to move* and (3) *how to move*. In this regard, we also highlight how the different cyber surfaces - discussed under *what to move*– relate to the various stages of an attack described in Section II. In Section IV, we discuss how the various MTD works have been implemented in practice, with special emphasis on the role of (and the effectiveness) of SDN and NFV in enabling them. We showcase examples of existing MTD testbeds and perform case-studies that can help security personnel in the adoption of MTD solutions for production-grade networks. In Section V, we elaborate on various qualitative and quantitative metrics that have been presented in the literature; this can help a cloud administrator decide if a defense mechanism is secure enough. In Section VI, we highlight areas, in terms of our categorization, that have received less attention and discuss an array of future research directions. Finally, we conclude the survey in Section VII.

## II. Background and Related Work

In this section, we first discuss related surveys in the area of Moving Target Defense (MTD) that look at the aspects of characterizing and evaluating proactive defenses. This helps us to situate our survey and highlight several aspects of MTDs that existing surveys ignore. Second, we describe the various stages of an attack that establishes a realistic threat model against a cyber-system. This helps us understand which step(s) of an attack a particular MTD technique seeks to disarm. Third, we highlight the traditional defense methods that are presently used to detect or reduce the impact of cyberattacks. As seen later, MTD mechanisms often leverage these traditional defenses, adding movement to the way they are deployed; this makes it harder for an attacker to fool the overall

TABLE I
OUR CATEGORIZATION OF MTDS IS A SUPER-SET OF THE CATEGORIZATION CONSIDERED IN EARLIER WORKS. THE ANALYSIS OF MTDS IN THE CONTEXT OF ADVANCED PERSISTENT THREATS (APTS), A UNIFIED VIEW TO ANALYSE THE WHAT, WHEN AND HOWS OF MTDS AND CASE STUDY OF THEIR IMPLEMENTATION ARE ENTIRELY NEW

| Research Work | Cyber Surface Shift Analysis | | | Relation to APTs | MTD Implementation | MTD Evaluation | | Research Directions |
|---|---|---|---|---|---|---|---|---|
| | What? | When? | How? | | | Qualitative | Quantitative | |
| [5] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| [6] | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [7] | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| [8] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| [9] | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |
| [10] | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Our Survey | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

defense. Finally, we provide an overview of existing databases (NVD, OVSDB, CVSS) [11], [12] that are used to obtain domain knowledge about known vulnerabilities and popular attack representation methods such as attack graphs and attack trees.

## A. Related Works and the Need for This Survey

We present a comparison of our survey to existing surveys in Table I. Firstly, we observe that most existing surveys [6], [9] provide only partial coverage of topics relating to *what, when, and how* to move the elements of the network. Section III provides a more holistic view of various Moving Target Defenses (MTDs). Moreover, the techniques surveyed do not talk about modeling Advanced Persistent Threat (APT) scenarios [13], [14]. Our survey, on the other hand, provides an overview of APT and its relation to the attack and defense surfaces, thereby helping us to highlight how a particular MTD may be effective against both known attacks as well as unknown attacks.

We provide an in-depth analysis of how MTDs are implemented and the role of networking technologies such as SDN and NFV in enabling them. We categorize the MTDs based on the maturity level of their implementation– ranging from simulation-based analysis to research test-beds and production-level industrial products. Table V summarizes various MTDs highlighting their use of centralized networking paradigms such as SDN/NFV (yes/no), and the level of maturity at which they have been implemented (analytic/simulation/emulation/commercial). This categorization has not been considered in previous surveys.

Some existing surveys [8], [9] have taken a look at categorizing the evaluation metrics for understanding the effectiveness of MTDs but these works do not talk about the different components that contribute to the evaluation of MTDs. In our analysis, we consider both security and performance metrics associated with each system configuration and with the ensemble, enabling us to highlight directions that can help improve existing MTDs solutions and mechanisms.

Beyond simply providing a categorization of existing work, the goal of our survey is to establish a common language for researchers who develop MTDs. This will help in making evident the aspects that have been considered and those that have been assumed away in the development of a particular MTD in the future. Our categorization also helps to identify promising directions for future research such as prevention, and hybrid surface shifting, improving the modeling of APTs, *etc.* (see Figure 14).

## B. Attack Modeling Techniques

Organizations utilize advanced infrastructure management tools and follow best practices such as software patching, hardening, analysis of the system's log for reducing the attack surface. Yet, skilled adversaries manage to compromise the network assets by utilizing zero-day attacks, customized malware, *etc.*, that are often difficult to detect or prevent using intrusion detection systems and anti-virus tools. For the effective deployment of intelligent cyber-defenses, it is crucial to collect information (called the threat model) about the attack process followed by an adversary.

An intelligence-driven approach focused on studying particular threats from an attacker's perspective is key for the detection and mitigation of sophisticated attacks in networks, which are characterized as *Advanced Persistent Threats* (APTs) [13], [14]. To understand the collection, correlation and categorization of data related to a cyberattack, Lockheed Martin defines the *Cyber Kill Chain* (CKC) [15]. The evidence-based knowledge derived from their study can help us in understanding and deploying appropriate defense measures. Thus, we will first describe the different phases of the CKC followed by a brief description of APTs and how they can be viewed through the lens of CKC. This setup will later help us understand how MTDs can be effective against the different phases of an APT.

*1) Reconnaissance:* The attacker gathers information about the target environment in this phase. For example, the attacker can perform passive monitoring using automated tools such as trace-route and Nmap to perform network probes.

*2) Weaponization:* The attacker, based on information obtained in the reconnaissance phase, utilizes tools and techniques such as a phishing e-mail, a malware-infected document, *etc.* to create a targeted attack payload against the victim.

*3) Delivery:* The transmission of infected payload occurs during this stage. For example, the attacker may leave a malware-infected USB on the victim site or send an email to the Chief Financial Officer (CFO) of the company. This step requires the attacker to evade the authentication and thus, the people (and not the technology) become a more important target during this phase. Thus, a trained workforce can help in reducing the attack surface area.

*4) Exploitation:* The attack detonation takes place during this stage. This phase involves the exploitation of a vulnerability and the attacker gaining elevated privileges on the victim's resources by utilizing specially crafted payload that exploits a known (or zero-day) vulnerability.

*5) Installation:* Once the attacker gains elevated privileges in the exploitation stage, they may install malware on the victim's machine or choose to harvest useful information in the victim's database. Tools that can analyze abnormal behavior such as anti-malware, *host-based IDS* (HIDS), *etc.* become quite important in attack detection during this stage.

*6) Command and Control (C&C):* After the installation phase is complete, the attacker contacts the *C&C* to maintain remote control over the victim machine. Tools such as *network-based IDS* (NIDS) and outbound *firewall rules* are quite useful in detecting and blocking malicious outbound connections requests.

*7) Actions on Objectives:* During this phase, the attacker executes the actions to achieve the attack goals, such as data-exfiltration, service disruption, *etc.* Two other important behaviors often observed in this attack-step are *pivoting*, which involves identifying similar target nodes that have already been exploited, and *lateral movement*, which involves identifying new systems that can be exploited in the network.

### *C. Advanced Persistent Threats (APTs)*

*Advanced Persistent Threats* (APTs) refers to a distinct set of attacks against a high-value target organization that differs from normal cyber attacks in several ways [13]. First, APTs are achieved by a group of highly-skilled attackers who are well-funded. According to *Mandiant Report* [16], APTs such as *Operation Aurora, Shady Rat, and Red October* have been used on a global scale for industrial espionage [17], [18]. Oftentimes, the attackers mounting APT work closely with government organizations. Second, an APT attacker is extremely persistent; they (1) pursue the objectives repeatedly, often over an extended period, (2) can adapt to a defender's efforts in trying to resist the attack, and (3) determined to maintain a level of access within the defender's system required to achieve their objectives. Third, the key objective of APTs is to either exfiltrate information or undermining the key services in the network using multiple attack vectors [19].

a) *Relation between APTs and CKC:* An APT can be broken down into five phases– reconnaissance, foothold establishment, lateral movement, data exfiltration, and post exfiltration [13]. These can be mapped to different phases of the Cyber Kill Chain. The *reconnaissance* phase in APTs maps directly to the *reconnaissance* phase in CKCs, which was described earlier. The *foothold establishment* phase comprises of the *weaponization* and the *delivery* phases of CKC. The attackers use information gathered from the reconnaissance phase in order to prepare an attack plan and eventually exploit vulnerabilities uncovered in the target organization's Web application, databases, and other software.

Once the attacker has gained a foothold into the target environment, they can try to move laterally in the target environment and gain access to other sensitive hosts and critical information in the organizational network in the lateral movement phase of APT. In this setting, the attacker needs to continuously explore (perform reconnaissance) and exploit the various components of the defender's system, mapping to the exploitation phase of the CKC.

TABLE II
HIGHLIGHTS HOW EXISTING DEFENSE MECHANISMS ARE RELATED TO THE DIFFERENT PHASES OF THE CYBER KILL CHAIN

| Phase | Network Defense Techniques | | | | |
|---|---|---|---|---|---|
| | Detect | Deny | Disrupt | Degrade | Deceive |
| Reconnaissance | Web Analytics | | | | |
| Weaponization | NIDS | NIPS | | | |
| Delivery | Vigilant user | Proxy filter | AV | Queuing | |
| Exploitation | HIDS | Patch | DEP | | |
| Installation | HIDS | chroot | AV | | |
| C2 | NIDS | ACL | NIPS | Tarpit | DNS redirect |
| Actions on Objectives | Audit log | | | QoS | Honeypot |

In the data ex-filtration phase of APT, the attacker tries to exfiltrate the collected data to their *command and control* (*C&C*) center. Most of the intrusion detection and prevention systems do ingress filtering and not egress filtering, thus, data exfiltration can often go undetected. The goal of the attacker in *post ex-filtration* phase is to maintain persistence in the system for a long duration of time until the funding for attack campaign is finished or the goals of attacking the organization/government are fulfilled.

### *D. Defense Methods*

In this section, we provide a brief overview of the various defense techniques and highlight how each of these defense mechanisms is effective in various parts of a Cyber Kill Chain (CKC) in Table II. This discussion will help the reader better understand some of the MTD techniques that use these traditional defenses as a building block.

*1) Web Analytics:* A huge amount of security-related information is present on the Web– in *social engineering* websites, phishing sites, and dark-Web forums– including discussion about a particular target product or company. As discussed by Glass and Colbaugh [20], the problem of exploring and analyzing the Web for information should provide (1) Security relevant information discovery capabilities (2) *Situation awareness* by performing real-time inference over the available information, and (3) predictive analysis to provide early warning for any likely security attacks.

*2) Intrusion Detection Systems (IDS):* There are two types of IDS agents that can be deployed in the network under attack, i.e., Network-based IDS (NIDS) and Host-based IDS (HIDS). NIDS such as *Bro* [21] and *Snort* [22], use signature match techniques based on known attack patterns and can flag incoming network traffic as malicious or benign. HIDS such as *auditd* [23] or *tripwire* [24], on the other hand check the Indicators of Compromise (IOCs) on the physical server or VM, in order to identify malicious activity, e.g., privilege escalation attempt by normal user.

*3) Intrusion Prevention Systems (IPS):* A network security threat prevention technology such as IPS [25] examines network traffic flow to detect and prevent vulnerability exploits. The exploits come in the form of malicious inputs to the target application or service. The IPS is often deployed behind the firewall and provides a complementary layer of analysis. Compared to IDS, the IPS system takes automated action based on traffic pattern match such as (1) dropping malicious traffic packets, (2) blocking requests from a source, (3) resetting connection, and (4) sending an alarm to the administrator.

*4) Proxy Filtering:* A (reverse) proxy server such as *nginx*, can act as an intermediary between the attacker located on the public network and private network. A proxy can help in shielding the real network from the attacker.

*5) Access Control List (ACL):* ACL can be applied at different enforcement points in a network. ACLs, such as *netfilter* [26], investigate network traffic and based on properties such as source/destination IP address, port number, protocol *etc.* decide either to *permit* or *deny* it.

*6) Data Execution Prevention (DEP):* DEP is a security feature that can be deployed in the form of hardware or software to prevent malicious code from running on the host. They *monitor programs* run on the host and ensure it uses memory in an expected (or safe) manner.

*7) Anti-Virus (AV):* A software program designed to protect the hosts from malicious programs such as a virus, spyware, worms, rootkits, key-loggers, *etc.* The AVs can be classified into two types– *signature-based AV* and *behavior-based AV*. The signature-based AVs check the malicious program against the database of known malicious programs. On the other hand, the behavior-based AVs check the program behavior by running it in a sandbox environment.

*8) Tarpit:* This defense technique involves purposeful introduction of delay when responding to queries. The key idea behind this defense mechanism is that adversaries will give up on a target if it takes too long to achieve the defined goal.

*9) QoS:* The *network traffic* can be segmented on the service type and importance. The segmented traffic can be analyzed for bottlenecks and threats. The malicious traffic can be slowed down in order to increase the *Cost of Attack (COA)* or selectively dropped.

*10) DNS Redirect:* The malicious connection requests can be served a different response than was expected. The attacker may seek to connect with command and control center using a page with malware, but DNS redirect will kill this chance.

*11) Honeypot:* A security mechanism, which can be used to detect, deceive or in some cases counter a malicious user trying to gain access to key network services [27]. Honeypots such as *Cowrie* [28] can help in better understanding the attacker's tools and tactics. The connection requests from the attacker are directed to a decoy service, which mimics the behavior of the normal service and logs the attacker's activity.

Although there exists a large set of defense mechanisms, attackers often use clever techniques to evade detection or prevention. SANS [29] discusses methods like *obfuscation*, *fragmentation*, *encryption* and *Denial of Service* (DoS) attacks against signature-based detection tools. Detection based on HIDS can also be evaded by a skilled attacker using techniques such as file location manipulation (using directories or files white-listed by IDS), application hijacking, *etc.* Furthermore, machine learning models that can help overcome some of the shortcomings of signature-based detection tools can themselves be fooled by adversarial attacks [30], [31]. On similar lines, deception techniques such as DNS redirect and Honeypot can help in deceiving the attacker temporarily, but over a longer period of time, the attacker will eventually change their attack methodologies thereby reducing the effectiveness of these defenses. Stojanovski *et al.* [32] performed experimental analysis on how to bypass *DEP protection* on *Windows XP* systems. Thus, there is a need to perform intelligent manipulation to assure the attacker's likelihood of reaching the desired goal is limited, even if they can evade the traditional detection methods. Additionally, the defense mechanisms discussed in Section II-D target known attacks often with easy to detect signature patterns.

*a) Proactive defenses against APTs:* The differences between APTs and regular cyberattacks, discussed in Section II-C, make it arduous to use traditional defenses and pre-specified threat models to address APTs as a whole. In this regard, proactive defenses can prove to be effective against APTs. While MTDs, as we will see later, makes it difficult for APT attackers by dynamically shuffling various system components (see Fig 3), other proactive defenses such as cyber-deception can prove to be effective in gathering threat-model information. For example, Shu and Yan propose a cyber deception to protect *FTP* services against APT attackers [33]. In their research work, a defender reroutes attack traffic to a host, which may be a honeypot, and the defender ensures that an attacker is not able to notice a connection difference between the real IP address and the honeypot trap. By observing attacker's behavior on the honeypot, the defender updates the threat-model and in turn, hardens their FTP services. A key aspect of this work is to make the attackers continuously believe that they are interacting with the original environment as opposed to a honeypot.

## *E. Domain Information for Modeling Cyber-Attacks*

Defenders almost always have information about the system they want to protect. This knowledge can help in enhancing the threat-model and, in turn, improve the effectiveness of MTD techniques. Such information may range from knowledge of the network architecture, the capacity of the individual machines, known vulnerabilities (and an idea of how dangerous they are), *etc.* We discuss here a few popular models and data sources that have been leveraged by researchers.

*1) Common Vulnerabilities and Exposures (CVEs):* are publicly known vulnerabilities and exposures that are categorized and can be referenced uniquely in the *National Vulnerability Database* (NVD) using a common vulnerability enumeration identifier (CVE-ID). This system is maintained and updated by the *Mitre* corporation regularly to

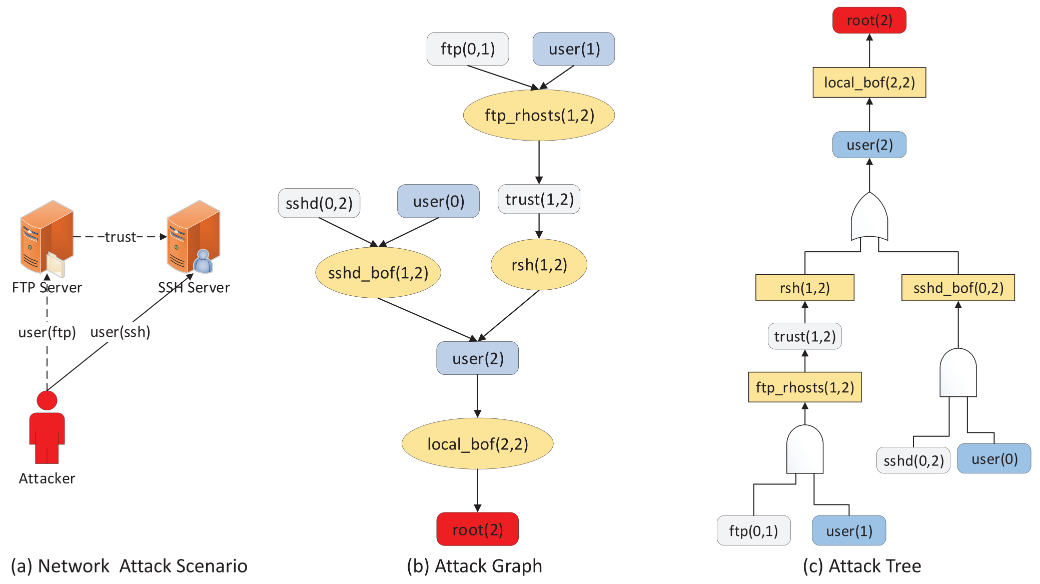


Fig. 2. Attack representations such as an attack graph (b) and an attack tree (c) for a simple network attack scenario (a) in a small-size cloud system with known vulnerabilities. Creation of these representations from network vulnerability and reachability information often suffers from scalability issues for real-world networks (see Table III).

make defenders and administrators aware of existing and new vulnerabilities.

*2) Common Vulnerability Scoring System (CVSS):* The use of the Common Vulnerability Scoring System (CVSS) for rating attacks is well studied in security [34]. For (most) CVEs listed in the NVD database, we have a six-dimensional *CVSS v2* vector [12], which can be decomposed into multiple components that represent Access Complexity (AC), i.e., how difficult it is to exploit a vulnerability, and the impact on Confidentiality, Integrity, and Availability (CIA) gained by exploiting a vulnerability. The values of AC are categorical {EASY, MEDIUM, HIGH}, while CIA values are in the range [0, 10]. These scores are also known as the Exploitability Score (ES) and the Impact Score (IS).

Although a defender may be aware of the known vulnerabilities present in their system (by being aware of the published CVEs that affect them), the knowledge of how they affect their system, in particular, may help them in making better decisions for security. The attack representation can be useful to quantify the attack and defense surface for MTD. To this extent, we define two heavily used representation methods that can represent known attacks present in a system– the *attack tree* and the *attack graph* as shown in the Figure 2.

The Figure 2, shows a network attack scenario, where an attacker has access to FTP and SSH server over the network. This example illustrates different methods for representing multi-stage attacks, i.e., attack graph and attack tree which we defined above. The FTP server consists of a vulnerability that allows the attacker to obtain remote shell (rsh) on the system. The SSH server, on the other hand, consists of buffer overflow (*sshd_bof*) vulnerability. The goal of the attacker is to obtain root privilege on the SSH server, i.e., *root(2)*. The attack progression using attack graph and attack tree has been presented in Figure 2 (b), (c) respectively.

**Attack Tree** [35] as shown in Figure 2(c) is another method of representing system security. The Attack Tree represents the network attacks. Attack Tree represents a monotonic path taken by an attacker starting from a leaf node to reach a goal node. Attack Tree usually consists of set of *AND* nodes (sshd(0,2), user(0) - Figure 2(c)) and *OR* nodes (rsh(1,2), *sshd_bof*(0,2)). The *OR* nodes represent one or more ways in which a goal node can be reached, whereas *AND* nodes represent different conditions that must be fulfilled to achieve a goal node. Children of the node are refinements of this goal, and the attacks that can no longer be refined are represented by leaf nodes [36].

*Definition 1:* An Attack Tree [36] can be defined by three tuples $(N, \rightarrow, N_r)$

- N is all possible nodes in the tree;
- $S^+(N)$ is *multi-set* of all possible subsets of nodes N;
- $\rightarrow \subseteq N \times S^+(N)$ denotes transition relation;
- $N_R$ represents the *goal node* of the attack tree.

**Attack Graph** is a data structure used to represent attack propagation in a network with vulnerabilities as shown in Figure 2(b). The start state of the attack graph represents the current privileges of the attacker. The goal state of the attack graph represents a state in which the intruder has successfully achieved his attack goal, e.g., data-exfiltration, root privileges on a Web server, *etc.* Security analysts use attack graph for attack detection, network defense, and forensics [45]. We formally define the attack graph as follows.

*Definition 2:* Attack Graph (AG) An attack graph is represented as a graph $G = \{V, E\}$, where $V$ is set of nodes and $E$ is set of edges of the graph $G$, where

1) $V = N_C \cup N_D \cup N_R$, where $N_C$ denotes the set of conjunctive or exploit nodes, $N_D$ is a set of disjunctive nodes or result of an exploit, and $N_R$ is the set of a starting nodes of an attack graph, i.e., root nodes.

TABLE III
COMPLEXITY OF THE VARIOUS ATTACK REPRESENTATION METHODS

| **Category** | **Details** | **Complexity** |
|---|---|---|
| Automated Attack Analysis [37] | Multi-prerequisite graph based on vulnerability and reachability information | O(E+NlgN); N is attack graph nodes and E is graph edges |
| Attack Cost Modeling [38] | Time Efficient Cost Effective hardening of network using Attack Graph | $O(n^{\frac{d}{2}})$; d represents the depth of the attack graph |
| Attack Cost Modeling [39] | Model checking based attack graph generation using Markov Decision Process (MDP) | Approximation algorithm $\rho(n) = H(max_{a \in A}$ $\{\mu_G(a)\})$, where A is Attacks, $\mu$ is maximization function. |
| Scalable Attack Graph [40] | Scalable attack graph using logical dependencies. | $O(N^2) - O(N^3)$, where N is number of nodes in attack graph. |
| Attack Graph based Risk Analysis [41] | Scalable attack graph for risk assessment using divide and conquer approach | $O(r(n+c)^k)$, where r is small coefficient. |
| Attack Cost Modeling [42] | Attack Graph cost reduction and security problem solving framework Min. Cost SAT Solving. | NP-Hard problem, SAT solving methods employed. |
| Ranking Attack Graphs [43] | Asset Ranking algorithm for ranking attack graphs to identify attacks. *Page Rank* based algorithm | Similar to complexity of page rank algorithm. |
| Attack Cost Modeling [44] | Identifying critical portions of attack graph. Min. Cost SAT solving, Counter-example guided abstraction refinement (CEGAR) | NP-Hard problem, SAT solving methods used. |

2) $E = E_{pre} \cup E_{post}$ are sets of directed edges, such that $e \in E_{pre} \subseteq N_D \times N_C$, i.e., $N_C$ must be satisfied to obtain $N_D$. An edge $e \in E_{post} \subseteq N_C \times N_D$ means that condition $N_C$ leads to the consequence $N_D$. $E_{pre}$ represents the attack graph pre-conditions (ftp(0,1) and user(1) in the Figure 2(b)) necessary for vulnerability exploitation and $E_{post}$ are the post-conditions (rsh(1,2) in the Figure 2(b)) obtained as a result of exploit.

The time taken to construct attack representation methods (ARMs) grows exponentially with an increase in the number of hosts or the number of known vulnerabilities in the network [46]. Authors in [47] survey various research works that try to address this challenge. Ammann *et al.* [48] present a scalable solution that assumes monotonicity; it allowed them to achieve scalability of $O(N^6)$. To mitigate the state explosion problem, most of the existing solutions try to reduce the dependency among vulnerabilities by using some logical representation [49] or by imposing a hierarchical structure to reduce the computing and analysis complexity of constructing and using ARMs [50]. In the latter work, the authors proposed a two-layer AG generation approach where the goal was to develop a faster method by considering network reachability and vulnerability information at different layers. The graphs constructed have vulnerability information of the individual hosts in the lower graph while the topological information of the network is in the upper layer. Unfortunately, the effectiveness of these methods on real-world networks is uncertain.

Table III highlights the complexity of creating various attack graph and attack tree-based methods. As can be see, scalability is a major concern when coming up with a good attack representation, thereby impacting the effectiveness of MTD techniques. In [51], the authors present a scalable solution for approximating the attack graph of a large scale data-centric network by using distributed attack graph computation followed by semantic clustering. We discuss attack representation methods as one of the *Quantitative metrics* which can be used for measuring effectiveness of MTD in Section V.

## III. CATEGORIZING MOVING TARGET DEFENSES

The goal of *Moving Target Defense* (MTD) is to continually move the components of an underlying system in a randomized fashion; this ensures the information gathered by the attacker in the *reconnaissance phase* becomes stale during the *attack phase* given the defender has moved to a new configuration within that time. This increases the uncertainty for the attacker, making it difficult (or rather, more expensive) for them to successfully exploit the system.

An MTD can be described using a three tuple $\langle M, T, C \rangle$ where $M$ represents the movement strategy that directs the system *how to move*, $T$ denotes the timing function that represents *when* a switch action occurs in the MTD system and $C$ represents the configuration space or the set of configurations between which the system switches, answering the question of *what to switch*. Answers to these three questions can help us define a useful categorization for the various MTDs. We believe that this categorization (1) gives a holistic view of the various MTD systems, providing a heuristic sense of what modeling aspects, when done carefully, make a particular defense effective, and (2) highlights opportunities on how modeling of a particular component of an MTD can either be improved or amalgamated with other MTDs.

As we will see, the configuration set $C$ and the timing function $T$ are often a design choice made the system administrator depending on the threat model. However, the movement function $M : H \rightarrow C$ needs to be a stochastic function that given the history of system configurations $H$ as input, produces the next configuration the system should switch to. A stochastic function is needed because a deterministic function can easily be learned by an attacker over time and thus, does not help the defender in terms of security.

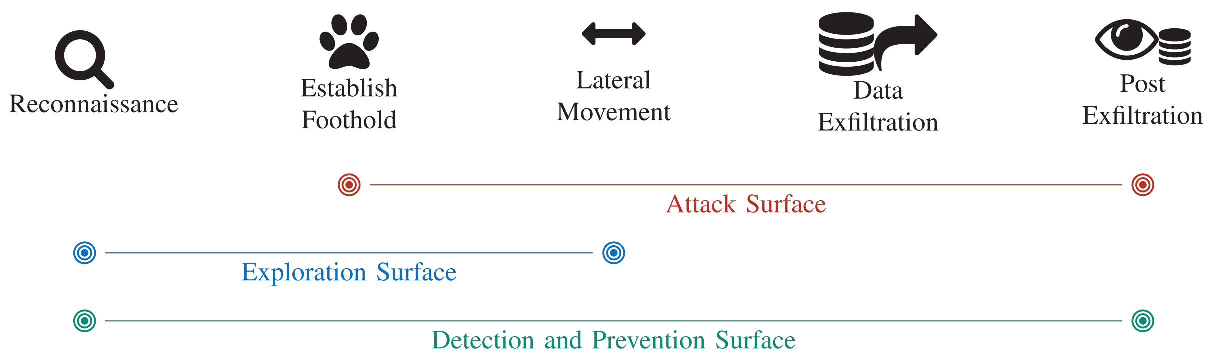


Fig. 3. A mapping between the different phases of Advanced Persistent Threat (APT) and various surfaces of a cyber system that various MTDs seek to move. Shifting of the exploration surface and the attack surface are effective only against some phases on an APT, whereas shifting the detection and the prevention surface is effective throughout the APT life-cycle.

## *A. The Configuration Set C – What to Switch?*

At an abstract level, from the perspective of an attacker, every software system can be categorized into four surfaces:

- Exploration Surface
- Attack Surface
- Detection Surface
- Prevention Surface

In this section, we use these surfaces as a basis for categorizing what the different MTDs shift. In the context of multi-stage attacks like APTs, an adversary needs to exploit all the different surfaces, but not necessarily in a predefined order (see Fig. 3). For example, an adversary may first try to explore the target network and try to figure out its topology, bandwidth, software deployed on the different nodes, *etc.*, but it may also need to perform reconnaissance at an advanced stage, say, to verify if it has actually gained access and established a foothold, i.e., a new vantage point, within the system. The knowledge gained in the exploration phase helps them to execute attacks that exploit the system, to move to different points in the network, and exfiltrate important information. As both exploration and attack traffic are malicious in nature, they tend to differ from legitimate user traffic. At that point, the detection and prevention surfaces come into play.

In Fig. 4, we show a Venn diagram that categorizes existing MTDs based on the surface they shift. Although this categorization is at a level of abstraction, we discuss various works and show how different MTDs define the elements of the set $C$. Most MTDs shift one surface at a time, rarely considering scenarios where other surfaces can be shifted in conjunction. After discussing the various MTDs, we briefly mention some unexplored research areas that can lead to the development of new defenses that shift multiple surfaces.

*1) Exploration Surface Shifting:* The goal of shifting the exploration surface is to ensure that the model of a system that an attacker can gather by exploration actions such as by scanning for open-ports, sending non-malicious traffic to uncover system topology, discover vulnerabilities, *etc.*, are noisy or inaccurate. Thus, an adversary, with this faulty information from the reconnaissance phase, is left with no other choice but to work with faulty view of the attack surface.

In [52], [53], the authors propose the concept of Random Host Mutation (RHM) – moving target hosts are assigned random virtual IP addresses (vIP) in an unpredictable and decentralized fashion. Formally, $C$ represents the set of bipartite graphs, where each configuration represents a mapping from the set of vIP addresses in the Virtual Address Range (VAR) to the set of hosts in the network. Every switch action changes the one-to-one mapping of hosts in the system to VARs. Another work [54] tries to implement an MTD on the same set of configurations using a centralized approach based on *(SDN)* technologies like *OpenFlow*.

Another line of work focuses on reducing the quality of information that an attacker can gather through exploration. In [55], the authors use a centralized approach to obfuscate the network links so that the topology that an attacker can retrieve via crossfire attacks is noisy and unreliable. In this work, each state is a possible path from the source to the destination of the crossfire attack. Thus, the configuration space $C$ represents the set of all paths from a source point to a destination point in the network. The MTD paradigm comes into play because the administrator chooses a path, in a randomized fashion, so that the attacker is not able to get reliable path information between any two points in the network. Given attacks may only succeed if the packet is sent over a particular path, attackers are forced to use a lot of attack traffic in order to succeed in exploiting the system or even obtain a reasonable estimate of the network topology. Such behavior increases their chances of getting caught or dealing with an inaccurate (and hopefully, not useful) model of the system. On similar lines, there have been works that either try to move the fingerprint of a protected host [56] or randomly alter the time schedule that guides when a host transmits information, reducing the effectiveness of selective jamming attacks against smart meters [59]. In [57], the authors propose to send back incorrect information to an attacker trying to query information about the hosts on the network. Although they come up with deterministic strategies for responses, the possibility of sending back different lies opens up when the underlying attack surface uses an MTD. These works are termed as *cyber-deception* because the defender is trying to deceive the adversary by feeding them false information about the system. In such cases, the goal of the defender is to ensure that the adversary's model of the underlying network is incorrect as opposed to just being incomplete or noisy.

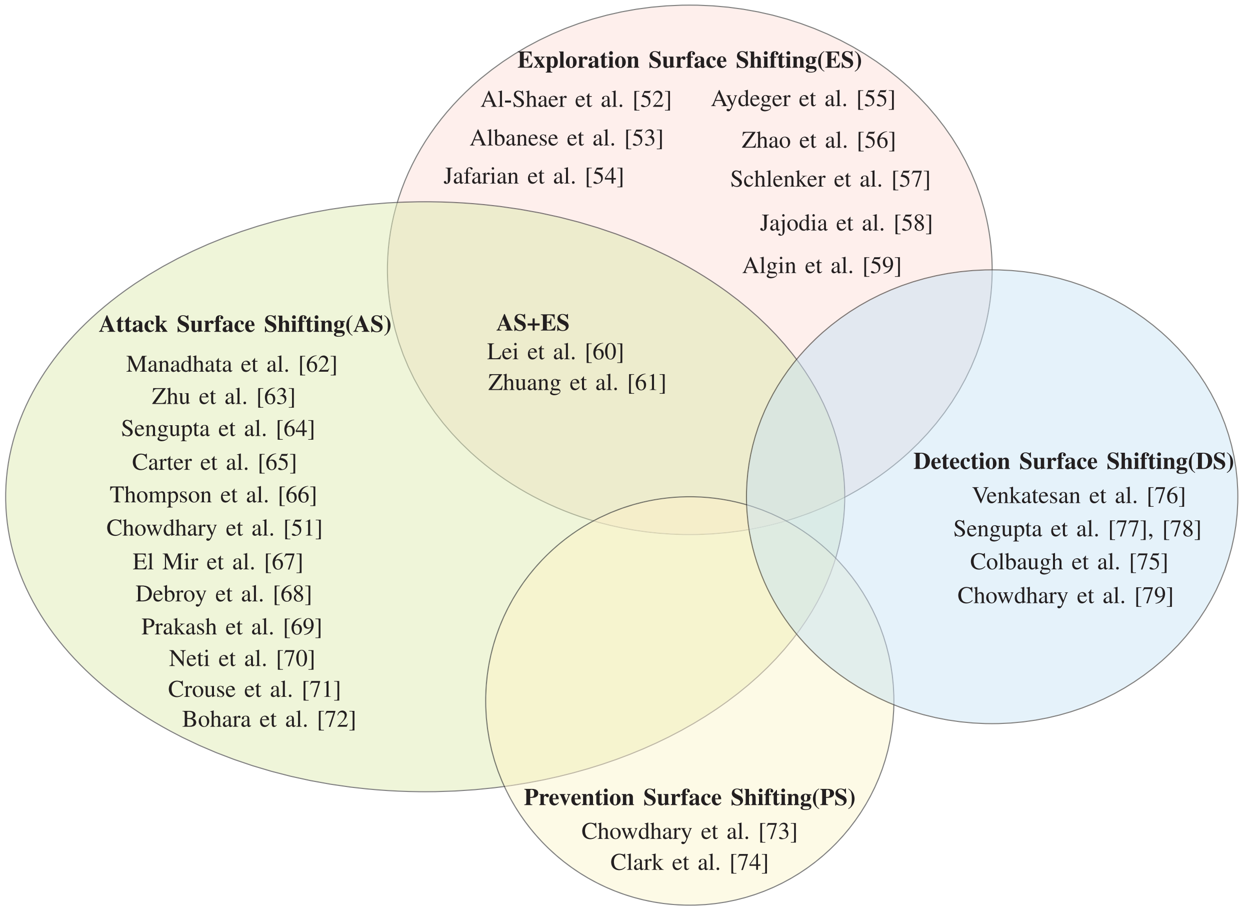


Fig. 4. The different surfaces that can be moved by a particular Moving Target Defense (MTD). Moving the exploration surface makes it harder for the attacker to figure out the exact system configuration by making the sensing actions noisy or erroneous. Moving the attack surface makes a particular attack inapplicable. Moving the detection surface, similar to 'patrolling methods', helps in providing efficient detection in budget-constrained cyber environments. Moving the prevention surface makes it harder for an attacker to ex-filtrate data even when they have a strong foot-hold inside the system.

In [58], Jajodia *et al.* looks at how they can deploy *honey-nets* in a strategic fashion and thereby, minimize the expense of deploying all possible honey-nets at once. They show that deploying honey-nets introduces deception in the network against *noisy-rich* (NR) cyber-attackers (i.e., adversaries who try to exploit all the vulnerabilities present in order to compromise the target network). In this case, if we represent the set of all honey-nets that can be placed in a network as $X$, then the configuration space $C$ consists of all budget-limited subsets of honey-nets that can be deployed by the administrator.

*2) Attack Surface Shifting:* Most of the work from the research community has focused on movement of the attack surface. The main aim of switching between attack surfaces is to render invalid an attack action that the attacker chooses after some exportation. For example, an attack to exploit a *Linux-based OS* will be useless if it is executed on a machine running a *Windows OS*. We first discuss some MTD methods that are, for a networking environment, defined at an abstract level. We will then, focus on ones that consider moving more specific elements of the network.

In one of the earlier works [62], the authors have a set of systems, which forms their space of MTD configurations $C$, and each configuration $c \in C$ can be represented by an attack surface characterized by three properties– (1) the set of entry and exit points into the system, (2) the set of *channels*, and (3) the set of un-trusted items (data/system/network link). The defender aims to switch between the different configurations to minimize the similarity of the attack surface in consecutive configurations and at the same time, have a minimum impact on performance. As we will soon see, this multi-objective trade-off is a common theme in many of the works on MTD.

In [63], Zhu and Başar break down a full-stack system into several layers (denoted by $l$). For each layer, they have a set of technologies that can be used to provide the functionality that the layer is responsible for. Now, when all layers (or stacks) come together to form the full-stack system, all possible combinations of technologies cannot be used to provide proper functionality to the end-user. Thus, from all these possible combinations, they handpick a few combinations of technologies that meet the use-case demands for the full-stack software among which they switch, which defines the configuration space $C$. On similar lines, Sengupta *et al.* [64] also assume a full-stack Web-application and thus, has similar action sets where the technologies for each layer are relevant to Web-application development. The two papers differ in the way they decide *how to switch* (to be discussed later).

Carter *et al.* design an MTD system that switches between different Operating Systems (OSs) (which are called 'platforms') [65]. In their case, the configuration set $C$ is the set of all OSs that they can shift between. They mention a notion of similarity (or diversity) between two configurations $\in C$ based on code syntax and semantics of the functionality provided. On similar lines, authors in [66] move away from MTD systems that consider a set of abstract configurations and implement an MTD that can perform the OS rotation for machines deployed in the systems hosted on a network using a centralized mechanism. We now shift our attention to MTDs that move elements solely relating to networks.

In [51], Chowdhary *et al.* describe an MTD that leverages port hopping to thwart known attacks on a cloud-based system. In their work, the states of the system are composed of variables, each of which indicates if a certain vulnerability in the system has been exploited (or not) and based on it, decides when and how to move. This fits well with our earlier discussion on how various surfaces are inter-related – the attack surface shifting comes into play when an attacker can successfully evade the detection surface. Along similar lines, authors in [67] move a deployed Virtual Machine (VM) to a different physical server if the impact of known vulnerabilities (measured using heuristic measures) on the physical server exceeds a certain threshold. In [68], the authors implement MTD at a different layer of the system abstraction where they move services deployed on a particular VM to another VM. A natural extension could be to use both the layers for designing a hybrid MTD that shifts (1) software deployed on a VM and (2) individual VMs deployed on actual physical servers in the cloud network. This is similar to the concept of multi-layer MTD, as discussed in [63], [64].

In [69], authors talk about a range of configurations (precisely, 72 of them) and analyze the effect of using various game-theoretic strategies for switching between them. They shed some light on the amount of security gain that can be achieved in various settings. In [70], the authors point out a fundamental assumption that is often inherent in cybersecurity in general and MTD systems in particular– the different configurations of the MTD are not vulnerable to the same attack (similar to the notion of diversity described in [65]). They create a bipartite graph that maps hosts to vulnerabilities and show that MTD is more effective when the diversity of the configuration $\in C$ is higher. Similar conclusions have also been drawn in regards to an ensemble of Deep Neural Networks (DNN) where higher *differential immunity* to attacks leads to higher gains in robustness [80].

In rare cases, the number of configurations in $C$ may become large. To address scalability challenges, a solution could be to first reduce the configuration set $C$ by partitioning them into disjoint subsets [78] or choose a reasonably-sized subset and perform MTD on them separately. In [71], the authors take the latter route and use *genetic algorithm* search methods where the fitness function measures the diversity of the subset. In [72], constraining the movement of software between containers both reduces the size of the set $C$ and helps to minimize performance impact.

*3) Detection Surface Shifting:* The concept of detecting attacks by inspecting traffic on the wire or the behavior of a request on a host machine has been the cornerstone of cybersecurity. Unfortunately, placing all possible Intrusion Detection Systems (IDS) on a system, especially in the case of a large network system, can lead to a degradation in the performance of the system. Thus, to minimize the impact on performance while keeping the attacker guessing about whether their attack will be detected or not, researchers have looked at intelligent ways in which a limited number of detection systems can be placed on the underlying network. These methods are similar to patrolling methods that try to identify security breaches in physical security domains like airports, wildlife sanctuaries, *etc.* with limited resources [81].

In [76] and [77], authors show that when faced with stealthy botnets and external adversaries, who are powerful enough to attack any internal node of a deployed system, shifting the detection surface helps to maintain system performance, while being effective in detecting attacks. In both of these cases, they have a set of nodes $N$ that are deployed in the network. The configuration set $C$ is composed of all $k$-sized subsets of $N$ (where $k(< |N|)\}$). In [78], [79], the authors argue that in many real-world multi-layered networks, the assumption that an attacker can attack from any place in the network is too strong and split the configuration set $C$ into disjoint sets based their position in on an attack graph.

In contrast to MTD for detection surface shifting, whose goal is to maximize security while adhering to performance constraints of the underlying network, some works investigate detection surface shifting for the sole purpose of enhancing security. In [75], the authors use an ensemble of classifiers that can distinguish a mail as spam and switch between them to make it more difficult for the attacker to fool the system. In general, the use of machine learning for anomaly detection [82] coupled with the paradigm of MTD for stochastic ensembles [80], [83] can lead to interesting future research, especially the introduction of a new attack surface (authors in [84] highlight several challenges in the use of supervised machine learning in cloud network settings). In these cases, each classifier is a configuration in $C$.

*4) Prevention Surface Shifting:* The goal of an MTD in shifting the Prevention Surface is to make the attacker uncertain about the defense mechanism that is in place; this forces the attacker to spend more resources and come up with sophisticated methods to exfiltrate the data. It also adds a layer of reasoning on the adversaries part; for example, distinguishing between whether their attack was undetected and went through to the actual system, or was detected and their behavior is presently being monitored becomes difficult.

Investigation of MTD techniques for shifting the prevention surface has been scarce, especially in the context of computer networks. We think this is mostly because an administrator can only use these defenses when they can identify an attack with high accuracy, which is often too strong an assumption. Yet, in some cases, authors make this assumption– in [73], an MTD mechanism that modifies the bandwidth of the network in response to malicious activity is proposed. The configuration space $C$ consists of infinite actions as the bandwidth can

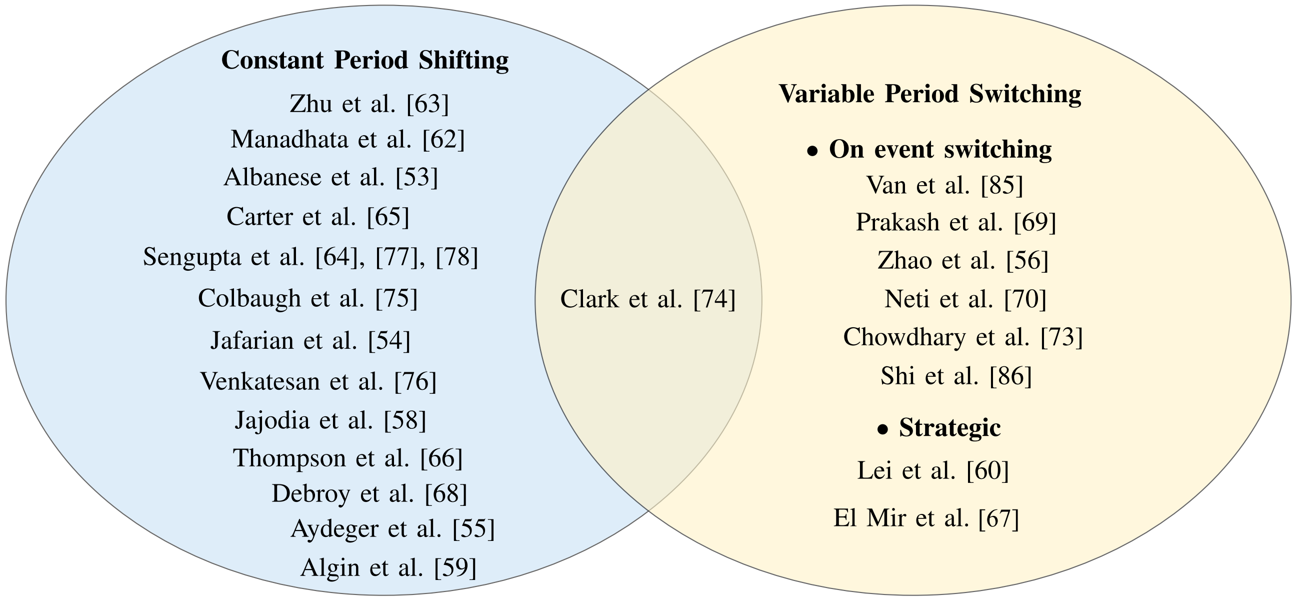


Fig. 5. The time period between two move events is either a fixed interval or decided based on some form of reasoning in the various Moving Target Defenses proposed in the literature.

take any real value. Similarly, researchers have also considered shifting the latency period when replying to a query that is, with high probability, malicious [74]. The latter work also considers the deployment of decoy nodes and switches among them to hide from an adversary seeking to actively uncover the prevention surface, i.e., the decoy nodes.

*5) Multi-Surface Shifting:* By choosing the shift multiple technologies that belong to different surfaces of a system, one can design an MTD that moves more than one surface. Yet, research on hybrid MTDs has been rare. In this regard, we discuss two works that try to shift both the exploration and the attack surface of a system.

In [76], authors show that constructing a simple proxy-based switching approach is ineffective by introducing a new attack known as the proxy harvesting attack that collects a set of IPs over time and then, performs a DDoS attack against all of them. To protect against such attacks, they propose an MTD approach that replaces entire proxies followed by the task of remapping the clients to these new proxies. This renders the exploration effort of the attacker useless and at the same time, reduced the attack efficiency of the attacker. As opposed to looking at a particular network-motivated setting, authors in [61] and [60] formalize the concept of MTD in which they highlight that the set $C = ES \times AS$, i.e., the elements represents configuration tuples that have technologies from both the attack and the exploration surface.

An interesting future direction could be to leverage existing security research on amalgamation of multiple surfaces and extend it to setting where MTD can be used. For example, NICE combines the techniques involving the detection of attacks to countermeasure (or prevention surface) selection in a single framework [87]. This work can be the first step to propose MTD solutions that move both the detection and prevention surface.

We believe the implementation of systems that integrate multiple surface shifting mechanisms under a single MTD mechanism has several challenges. For example, some of the surfaces might lend themselves easy to management by a centralized controller like SDN, while others are better suited for movement in a decentralized fashion. Characterizing these challenges that deter researchers from considering multi-surface shifting and suggesting methods to overcome that is an interesting research direction.

### *B. The Timing Function T– When to Switch?*

Having defined the possible configurations that a defender can switch between, the next question to ask is *at what point in time does a defender choose to execute a switch action that changes their present configuration c to another configuration* $c'$? To answer this question, we divide the works on MTD systems into two broad categories based on how the time-period between multiple switches is determined–*Constant Period Switching* (CPS) and *Variable Period Switching* (VPS). We first describe the primary characteristics of these categories and then categorize the different MTDs (see Fig 5).

In CPS, the timing function enforces an MTD system to move, in a randomized fashion, after a constant time-period. This time is dependent on the composition of the set $C$ because attacks on different compositions have different investments in regards to time. On the contrary, in VPS, the timing period between one switch action can vary from the time of another switch. The works in this category can be further subdivided into two distinct categories based on whether the timing function $T$ is reflexive or strategic based on opponent modeling. While the two classes have the exact opposite definition, there is no restriction to not use both in certain scenarios. For example, when multiple surfaces being shifted, it is reasonable to have one surface shift using CPS while the other is shifted via

VPS. We conclude his section with a brief discussion in this regard and later, elaborate on it in Section VI.

*1) Constant Period Switching (CPS):* As mentioned, the key idea of these methods is to move the MTD system after a constant amount of time. In existing works, this is stated as an implicit design choice (for example, in frameworks that model the MTD as a game, the actions of the defender are played after a constant time-period unless they explicitly discretize the time and consider it in the state information), or an explicit one (for example, to ensure security, we move the MTD system after every 100 seconds).

A lot of the research works in MTD literature such as [58], [62], [64], [77], [88], do not explicitly bring up the topic of a time-period but inherently assume that the system moves after a constant time. In these works, authors (1) model the problem as a single-step game and generalize the solution of this game to multiple stages (not necessarily repeated games), and (2) showcase the problem of *when to switch* as a different one than *how to switch* and only address the latter, forcing the system admin (or defender) to consider a constant time-period when implementing MTD solutions.

Some research works explicitly state that the time-period is uniform and the equilibrium strategy is played at the start of each stage [59], [63], [65]. Note that, some of these works, such as [53], [59], experiment different timing functions for switching, i.e., see the effectiveness of an MTD if the constant time-period was set to different predefined values. This helps them empirically figure out the constant timing function that works best. Some other works that employ a constant timing function, use different terminology. For example, authors in [76] address DDoS attacks use the notion of a pre-defined fixed frequency of switching while research in [54] refers to *T* as a constant mutation interval. Other works that have syntactic generalizability (by using notations such as $t_i$ that allow a user to use different time intervals for configuration *i*) default to a constant timing function.

An interesting question is *how long should this constant time-period be in practice*? In [66], the authors vary the time-period from $60-300$ seconds and evaluate the effectiveness of OS rotation in a network-system based on (1) the likelihood of thwarting a successful exploit, (2) the magnitude of impact for an exploit, and (3) the availability of applications. They show that a smaller $t = 60s$ was often good enough to thwart Network Mapping (Nmap) [89] attacks, but may allow accurate OS fingerprinting when $t = 60s$. The latter does not help the attacker because, during the execution of the attack, the fingerprinted OS might simply have shifted. For $t = 300s$, these results look less promising against automated attack systems , but work well when evaluated against human experts. To allow for faster rotations, the authors set up machines with different OS and after every time interval, simply redirect traffic to a new OS. While this makes faster switching less painful, it incurs added cost to maintain multiple machines. To reduce this, we believe that having at least two systems is necessary to allow for such small switching periods, especially when shifting between OSs. With two systems, one VMs sets up an environment while the other handles traffic and the role switches in the next time step.

In [68], researchers use the knowledge obtained from historical attack data to obtain a cyber attack inter-arrival (CAIA) rate. Then, they setup an optimization problem to maximize the time-period of switching, constrained upon the fact that the constant timing function has a value less than CAIA. Along similar lines, authors in [55] have looked at *traceroute* [90] data between possible source-destination pairs to decide on a reasonable time-period for obfuscating links or mutating routes of ICMP packets on the network. This helps to mitigate crossfire attacks.

An interesting case arises in [52] because the states of this MTD represent a bipartite mapping between hosts and virtual IPs (vIP) and the authors let each edge in the mapping have a separate, but constant, mutation time. Some hosts belong to a set of High-Frequency Mutation (HFM) set while others belong to a Low-Frequency Mutation (LFM) set. To ensure that availability of a host is not impacted, the hosts that are in the HFM set map to two vIPs (one that it was mapped to in the previous round and one to which it is mapped in the present round) for a small time duration compared to the time-period of switching. In [53], the authors also have a local timer maintained by each host; its expiry triggers the change in their vIP. They use *ad hoc* messaging to communicate this change to other nodes in the network to facilitate routing.

*2) Variable Period Switching (VPS):* The idea behind this switching strategy, as evident from its name, is to have a way to come up with a timing function that varies the switching time based on the present condition of the system. For example, if the system transitions through a series of configurations $\langle \ldots, c, c' \ldots \rangle$ and the time spent in $c$ and $c'$ is denoted by $t$ and $t'$ respectively, then $t \neq t'$. We believe that doing a VPS along with an MTD mechanism is similar to doing a two-layer MTD where the first layer deals with a meta MTD for shifting the time-surface and then, the second layer MTD is responsible for executing the actual cyber-surface switching. We will now categorize the MTD works under the following sub-classes.

*a) On-event switching:* Most works that have a general timing function, which isn't constant, fall under this sub-category. The main idea, evident from the nomenclature, is that when a particular event occurs, such as detection of an attack, unavailability of a link or a server, the timing function specifies a time-period to switch. In most cases, the switch action is triggered immediately.

A well-known case study is that of the *FlipIt* games [85]. The true state of the system is represented using a boolean variable 1/0 that indicates whether the defender or the attacker has control of an underlying software system. The value is unknown to both the players, and a defender strategy is based on its belief that the true value is 0 (which represents the attacker having access to the server). Thus, the timing function is dependent on this belief. On similar lines, in [56], authors update the belief about the sender type (are they malicious?) upon detection of suspicious packets on the network. This belief, in turn, influences the timing function *T*. Authors in [70] have a belief variable indicating whether a vulnerability was exploited or not. The timing function orders a switch action if the belief value exceeds a certain threshold.

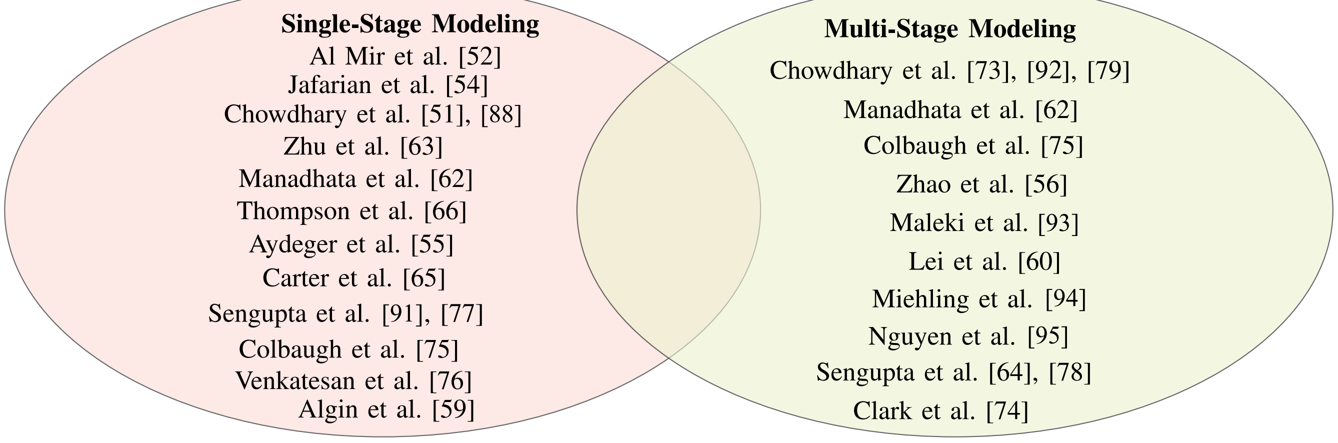


Fig. 6. Game-theoretic modeling of Moving Target Defenses (MTDs), that is necessary for obtaining the movement strategy *M*, can be categorized into either single-stage or multi-stage game modeling. This choice reflects the type of threat model being considered and the characteristic of the particular system for which the MTD is implemented.

Other works argue that obtaining belief parameters (such as *with what probability is the sender an attacker vs. a normal user?* or *with what probability is the system compromised?*) that are accurate is difficult in cybersecurity settings because they are based on analysis of historical datasets that are both scarce and might represent a different distribution. These works seek to use the intuitive knowledge of security experts in designing the timing function. In [73], if the admin detects a spike in bandwidth usage of a particular link or sub-net, they change the maximum bandwidth value allocated to that link or the network. Authors in [86] scan open connections routinely and upon detection of unexpected connections, move between MTD configurations to protect the system against port-scanning attacks.

*b) Strategic:* To understand these methods, we design a timing function that produced integer values in the range $[0, t_{\max}]$. These represent discrete time intervals and a value *t* represents the time at which a move occurs.
In [60], the authors model the MTD setting as a game-theoretic setting. While we discuss this in much more detail in the upcoming section on the design of the movement function *M*, we briefly highlight it here. The basic idea involves representing a game-state that considers the currently deployed configuration *c* and an integral-valued time variable *t*. In each state that can be represented by the tuple (*c*, *t*), it recommends between the actions *switch* or *stay*. The stay action takes them to the state (*c*, *t* + 1), while the switch actions move them to the state $(c', 0)$. Note that the stochastic switching or movement policy described later may, with some non-zero probability predict, $c' = c$.

In general, the idea of incorporating the time variable as a part of the state can result in an explosion in the number of states and thus, computing a policy for all states time intensive. Authors in [67] show that they can model the problem similarly but resort to a simple strategy for defining the timing function. While it may be sub-optimal, scalability is no longer a concern. The timing function considers the impact of (known) vulnerabilities in the currently deployed configuration and based on it, recommends a switching time $t \in \{0, 1, \ldots, \infty\}$. If there are no vulnerabilities whose impact is greater than a predefined threshold, then the system remains static until a new vulnerability is discovered.

*c) Hybrid switching:* Although the idea of having both a CPS and VPS in an MTD mechanism seems counter-intuitive at first, authors in [74] look at MTD mechanisms where one layer shifts the time window of replying to a strategic adversary using an event-based timing function while the other layer that moves the decoy nodes to hide from an active adversary is done using a constant timing function. This acts as a stepping stone for an investigation on the effectiveness of CPS *vs.* VPS for a particular surface and finally when a combination of both should be considered.

## *C. The Movement Function M – How to Switch?*

In this section, we look at how different MTDs come up with a policy for movement. While some surveys (e.g., [5]) have been solely devoted to analyzing how this question, we propose a unifying framework that maps any MTD system to a Markov Game. This gives us a better sense of the implicit assumptions made by the various MTDs when coming up with *M*. Before getting into the categorization shown in Figure 6, we first briefly introduce Markov games [96].

An MTD system can be mapped to a Markov game defined by the tuple $\langle S, A^{\mathcal{D}}, A^{\mathcal{A}}, T, R^{\mathcal{D}}, R^{\mathcal{A}} \rangle$ where $S$ denotes the state of the MTD system (often represented by deployed configuration *c*), $A^{\mathcal{D}}$ denotes the set of pure strategies that map to the set of configuration set *C* and the mixed strategy over this set maps to the movement strategy *M*. The other variables are specific to the domain for which the MTD system is designed and help in coming up with good *M*; $A^{\mathcal{A}}$ represents the set of attack actions, *T* represents the transition from one configuration to another depending on the joint action taken by the defender and the attacker, and $R^i$ represents the rewards for the player *i* (i.e., the defender $\mathcal{D}$ or the attacker $\mathcal{A}$). We will use the superscripts $\mathcal{D}$ and $\mathcal{A}$ to represent the functions for the defender and the attacker respectively.

*1) Single-Stage Modeling:* In this section, we investigate works that have modeled the interaction between an attacker and a defender as a single-stage game. In these settings, the goal is to come up with a single mixed-strategy that can be played in all configurations (or states). Thus, the movement function $M : H \rightarrow C$ is independent of any history or $H = \emptyset$. As we shall see, the different works consider various notions of equilibrium to come up with this mixed-strategy.

In [62], the configuration of the system is modified by a pair of actions, one by the defender and one by the attacker. The defender's action, in this case, maps a value of system features represented by the set $F$ to one of the possible actions in the set {*enabled*, *disabled*, *modified*, *unchanged*}. The attacker's action set $A^{\mathcal{A}}$ is comprised of conceptually opposite actions that can *dis-enable* a port, *disable* a functionality, *etc.* and the rewards, $R^{\mathcal{A}}$ and $R^{\mathcal{D}}$ for the attacker and the defender, are given by weighing the change in system's features $\Delta F$, i.e., the change in the attack surface $\Delta AS$ and the attack surface measurement $\Delta ASM$, as follows.

$$R^{\mathcal{D}}\left(s, a^{\mathcal{D}}, a^{\mathcal{A}}\right) = B_1(\Delta F) + B_2(\Delta AS) - D_1(\Delta ASM)$$
$$R^{\mathcal{A}}\left(s, a^{\mathcal{D}}, a^{\mathcal{A}}\right) = B_3(\Delta ASM) - D_2(\Delta AS)$$

where the $B_i$ and $D_j$ are weights defined by security experts of the system. Note that the *cost of a defense* action ($D_1$) and the cost of attack action ($D_2$) are incorporated in the reward. Thus, solving for an equilibrium of this game results in strategies that account for the cost-reward trade-off. The authors use the notion of a Subgame Perfect Equilibrium (SPE).[1]

In [52] and [54], the defender played a predefined strategy that is either uniform random (choose *k* random vIPs at random) or based on intuitive heuristics (select top-k scanned vIPs). The latter is based on the assumption that the most scanned vIPs will not be scanned as frequently in the new state. The chosen vIPs are then assigned to the *k* hosts using a purely random matching function.

In [51], [88], the authors explain the game modeling based on a real-world example and compare the effectiveness of a reactive switching strategy to the Uniform Random Strategy (URS). URS, which is found to be more effective, picks all pure strategies $a \in A^{\mathcal{D}}$ with equal probability $|1/A^{\mathcal{D}}|$. An array of works that simply use URS as the defender's policy are [55], [59], [66]. This makes an implicit assumption that the game is a constant sum and the normal form game matrix has symmetric rewards. When no information is available about the attacks and the defender has no preference over the MTD configurations, this may be a reasonable assumption. Fortunately, defenders often have an idea about an attacker's threat model and thus, such assumptions (resulting in URS) result in highly sub-optimal movement strategies. Works such as [66] closely mirror URS strategies but make a minute modification– given the present deployed OS configuration *c*, they consider a URS over remaining OS configurations, i.e., $C \setminus \{c\}$ as the movement function *M*. In [55], given all networking paths from source to destination, one is picked at random while in [59], the authors use to Fisher-Yates shuffling to select a transmission schedule at the start of every round.

In [63], [65], [76], [77], [91], and [98] the authors design a single-stage normal form game. Thus, if the defender chooses to deploy a particular configuration $c \in A^{\mathcal{D}}$ and the attacker chooses a particular exploit $e \in A^{\mathcal{A}}$, then the rewards for the players can be read from the normal-form game matrix. At equilibrium, a mixed strategy over the defender's actions turns out to be the optimal movement policy. Thus, *M* is a stochastic function based on the mixed strategy, chooses the next configuration at the end of each switching period. In [98], the authors consider an MTD for the Internet of Things (IoT) and show that Zero-Determinant strategy results in a dominant strategy for the single-stage game. In [63], the authors assume (1) a zero-sum reward structure, and (2) a threat model in which the attacker is irrational. The opponent model for the attacker is determined based on historical traces of the attacker's response to defense strategies and eventually updates the utilities of the game to reflect the rationality of the attacker. After each update, the resulting Nash equilibrium (NE) is played. In [65], the authors assume that if the present system is working with the operating system *c* and the system is made to switch to the operating system $c'$, lower the similarity between *c* and $c'$, more secure the move action. To model this, the defender's rewards are defined w.r.t. switch actions and, over time, fine-tuned by simulating the behavior of an adversary. Similar to [63], the NE strategy is chosen to be the defender's policy.

The other works assume that an attacker will eventually, with reconnaissance on its side, figure out the mixed strategy of the defender (using maximum likelihood estimates). Thus, authors of [77], [91] concentrate on finding the Strong Stackelberg equilibrium (SSE) strategy for the defender. The rewards for their game follow a general-sum structure and are obtained using the CVSS metrics of known attacks that can successfully exploit one of the defender's actions, i.e., the MTD configurations that can be deployed. On similar lines, authors in [76] use the notion of SE to obtain defender's strategy to thwart DDoS attacks.

*2) Multi-Stage Modeling:* Works in these sections reason about (1) the history, i.e., paths that the system has taken, which may be a result of the actions taken by the players, to reach the present state, (2) the future, i.e., how the decision in the present state affects the rewards in the future or both.

In [73], similar to the work by [62] in single-stage modeling, the authors discretize the continuous action space of the defender, which represents bandwidth values, to two levels– high and low. Then they find a meaningful defender strategy over them by ensuring that in a repeated game setting, the advantage that an attacker gained by packet flooding attacks (at some stage in the past) is neutralized by reducing their bandwidth to the low state for $1 \leq x < \infty$ number of game stages. On similar lines, authors in [75] consider a repeated game setting and analyze the defender's policy against self-learning attackers. They infer that in their case, the optimal policy converges to the URS. Other works such as [56] also update their belief about an attacker based on observations

[1]In sequential games, a strategy profile is an SPE if it is a Nash equilibrium in every subgame of the original game [97].

drawn from multiple plays of the game. This belief is then used to come up with an optimal strategy for the defender.

A set of works in MTD leverage the attack graph of an underlying system to better reason about the sequential process of an attack. In [94], the authors model the problem of network intrusion via a Bayesian Attack Graph. The action set for the attacker $A^{\mathcal{A}}$ includes the (probabilistic) paths an attacker can take in this graph. Then, the authors map these paths with the defender's imperfect sensing capabilities to form a Partially Observable Markov Decision Process (POMDP) [99]. Thus, the state and transition of this POMDP are designed to model the attacker's behavior and the optimal policy becomes the defender's strategy in a particular belief state. A shortcoming of this work is that this strategy may be sub-optimal if the *attacker* deviates away (intentionally or not) from the assumptions that informs the POMDP modeling. On the other hand, modeling the scenario as a Partially Observable Stochastic Game (POSG) [100] results in scalability concerns for any real-world system.

Authors in [60], [79], [93], and [101] relax the assumption about partial observability and formalize MTD in a Markov Game framework. In [93], authors consider policies over the defender's action set that comprises of either single or multiple IP hops. Each action results in the defender uniformly selecting the next state given that an attacker samples actions randomly from $A^{\mathcal{A}}$. They can show that multi-element IP hopping actions increase the defender's reward by a greater magnitude compared to static defense strategies. As we will discuss later, the authors do not model the cost of performing a hop action or the downtime associated with switches in the defender's reward function. Thus, the optimal defender policy might be sub-optimal for performance in real-world multi-layered network attack scenarios.

In contrast, the works [60], [78], [79] incorporate this trade-off in the rewards ($R^{\mathcal{D}}$ and $R^{\mathcal{A}}$) of the Markov Game, and can thus generate policies for the defender that trade-off security corresponding to an attacker's actions and performance at each step of the game. In [60], the inclusion of time variables and attack strategies in the state allows it to formulate the problem as an MDP but, becomes vulnerable when an attacker is aware of this modeling. It also makes it less scalable for large networks. In all these works, the impact on performance $C^{\mathcal{D}}$ is a part of $R^{\mathcal{D}}$ and often, just a *heuristic idea* as opposed to simulation in a real system that informs these values. The more accurate these measures become, the better is the strategy. Authors in [64] model the performance and security concerns in a different way than the above methods. The performance cost represents the switching cost while the rewards of the state represent the security costs of deploying a particular configuration. Then, they formulate an optimization problem that produces a defender strategy that reduces the cost of switching over two steps (one switch) and maximizes the security over a single step. Although the authors in [64] do not discuss how sub-optimal their switching strategies are, recent work by [102] extends their approach to a Markov Game setting with the Stackelberg Equilibrium concept.

Lastly, authors in [74] look at an MTD defense mechanism for deception (obfuscation of the links to send attacker to decoy nodes) and model attackers who are actively trying to uncover this deception over a repeated stage interaction. For one problem, they use the Nash equilibrium (NE) while for the other (identifying decoy nodes) they consider the Stackelberg equilibrium (SE) as the defender's strategy. We believe that it is necessary to highlight a shortcoming of the different modeling choices, especially in light of multi-stage attacks and APT scenarios, however optimizing the model and ensuring the scalability is a difficult task, thus we highlight these as possible research opportunities in Section VI.

TABLE IV
COMMON CAUSES OF MIDDLEBOX FAILURES. MISCONFIGURATION IS A DOMINANT CAUSE OF FAILURE BECAUSE OF DIFFERENT UNDERLAY AND OVERLAY NETWORK

| Middlebox | Misconfiguration | Overload | Physical/Electric |
|---|---|---|---|
| Firewall | 67.3% | 16.3% | 16.3% |
| Proxies | 63.2% | 15.7% | 21.1% |
| IDS | 54.5% | 11.4% | 34% |

## IV. IMPLEMENTATION OF MOVING TARGET DEFENSES

In this section, we discuss how the various Moving Target Defenses (MTDs) have been implemented using research test-beds and industrial products. First, we briefly discuss the tools for MTD implementation. In this regard, we highlight that traditional networking technologies like middleboxes have a set of disadvantages. To overcome this, users can leverage the advancement in networking technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV). We briefly emphasize their role in making MTD a pragmatic solution for real-world systems. Second, we highlight how the existing MTDs, discussed in the survey as shown in Table V, have been evaluated. We select a subset of works and conduct a detailed case-study to highlight how movement strategies can be implemented in practice. Third, the Section IV-D provides details of the test-beds used for academic research and industry products.

### A. Middleboxes for Enabling Moving Target Defenses

Middleboxes are the devices used by network operators to perform network functions along the packets data-path from source to destination, e.g., Web Proxy, Firewall, IDS. This provides a decentralized framework that can be used alongside existing network technology to implement strategies for MTDs. Furthermore, researchers have focused significant effort on several issues associated with middleboxes, such as ease of use, ease of management and deployment, the design of general-purpose middleboxes for different network functions *etc.* This makes them seem like a good choice for enabling the practical implementation of MTDs.

Unfortunately, a survey of various middlebox deployments conducted by Sherry *et al.* [103] reveals some drawbacks such as increased operating costs caused by misconfiguration and overloads that affect their normal functioning. As shown in the Table IV, based on the results in the survey [103] of 57 enterprise network administrators from *NANOG* network operators group, the misconfigured and overloaded middleboxes are the major reasons for middlebox failure. Furthermore, about 9% of administrators reported about six and ten hours per week dealing with middlebox failures. Also, the adoption of new and secured middlebox technologies is slow in the industry based on the survey results. In the median case, enterprises update their middleboxes every four years. The use of traditional networks for incorporation of MTD defense can increase chances of network misconfiguraion and outages.

Moreover, this static nature of the middleboxes themselves, in contrast to the dynamic nature of the system they enable, provides an assymetric advantage to the attackers as noted by Zhuang *et al.* [61]. The attackers can perform necessary network reconnaissance, identify the services and configuration of the applications, and operating systems by leveraging the middleboxes. This information helps the attacker in choosing best-fit attack methods against the static configuration and select the best time to attack the system. The attackers can use the compromised resource to maintain the foothold in the network for a long period of time and try to exploit other high-value targets in the same network.

## *B. SDN and NFV for Enabling Moving Target Defenses*

*Software Defined Networking* (SDN) [4] is defined as a networking paradigm that decouples the control plane from the data plane, allowing a global view of the network, and centralized control capabilities. SDN deals with packet headers, from layers 2-4 of OSI model and other protocols such as MPLS [139]. SDN and NFV have emerged as a state-of-the-art network architecture for data centers and backbone networks. Google's *B4 project* [140] shows the feasibility of SDN for handling real-world network challenges such as traffic engineering and Quality of Service (QoS).

The decoupling allows a network architecture where switches act as forwarding devices, maintaining flow tables populated with flow rules. The new architecture allows considerably more flexible and effective network management solutions, and a unified programmable interface that can be utilized by software developers for application deployment [141]. The SDN architecture can be vertically split into three layers described below:

- *Application Plane:* It consists of end-user business applications that consume SDN communication and network services. The examples include network security or virtualization applications.
- *Control Plane:* The control plane consists of SDN controllers such as ONOS [105], OpenDaylight [110] providing open Application program interfaces (APIs) to monitor network forwarding behavior. The communication interface between the application and control plane is known as northbound interface. The interface between control and data plane is known as southbound interface.
- *Data Plane:* The forwarding elements such as OpenFlow switches are present in the data plane. The forwarding devices can be physical or virtual switches. Control plane installs flow rules in order to govern the forwarding behavior of data plane devices.

Network Functions Virtualization (NFV) [3] has emerged as a technology that provides a virtualized implementation of hardware-based equipment such as firewall, routers, and Intrusion Detection Systems (IDS). Virtual Network Functions (VNFs) can be realized through virtual machines (VMs) or containers running on top of the physical server of cloud computing infrastructure. SDN acts as an enabling technology for NFV and help in centralized management, making it easier to implement MTD. Despite the great benefits offered by SDN and NFV, its use to implement MTDs has been limited to a few research works (as can be seen in Table V).

## *C. SDN-Based MTD Applications and Case Study*

We categorize the MTD industrial, and research implementations from the perspective of networking technologies used, e.g., traditional network or SDN/NFV as shown in the Table V, as highlighted in column 2. It is noteworthy that SDN/NFV has been a dominant technology for MTD research. Around 34% of the research works in Table V, utilize SDN/NFV for implementation of MTD or cyber-deception. Column 3 describes the layer in network protocol stack where the describe MTD solution has been implemented. As on may notice, most research works are implemented at the network or the application layer. In the fourth column of the table, we identify the key technologies used by these research works; if they utilize a particular implementation test-bed (for example, GENI [124], then the name of the test-bed is mentioned in this column. Column 5 of the table shows the level of *maturity* of the MTD research work. We categorized the research works into four levels - *analytic* - if only numerical results or thought-based experiments have been presented in the said paper. We put the research work under the category *simulation* - if the research work describes some simulated network, e.g., *Mininet*. The *emulation* category consists research works that use close to real world networks, e.g., multiple VMs deployed in a cloud testbed, e.g., GENI [124]. Lastly, if the research work has been deployed in some commercial product dealing with live attacks in a production grade network, we consider these research works under category *commercial*.

We observed that more than 50% of the research works use simulated networks or applications when experimenting with MTD techniques, whereas $\sim$ 34% of research works have used emulated environments for performing MTD based analysis research. Only few MTD solutions, e.g., $\sim$ 13% have implemented MTD at commercial level (in production grade networks for dealing with live attacks). This shows that though MTD has been well accepted in research community, and benefits of MTD can help in dealing with different threat models, the industry adoption of MTD is rather slow. The key reason behind this is the adverse impact that MTD can induce on Quality of Service (QoS), and the additional cost-overhead associated with deployment of MTD. We discuss these factors under *Qualitative* and *Quantitative* metrics in Section V.

*1) SDN-Based Network Mapping and Reconnaissance Protection:* The first step of the Cyber Kill Chain (CKC) is the identification of vulnerable software and OS versions. Most

TABLE V
A TAXONOMY OF MTD IMPLEMENTATION CATEGORIZED DEFENSES BASED ON THE USE OF SDN/NFV, THE LAYER OF NETWORK PROTOCOL STACK THEY PROTECT, KEY TECHNOLOGIES THEY USE, AND THE LEVEL OF MATURITY AT WHICH THEY HAVE BEEN IMPLEMENTED

| **Research Work** | **SDN/NFV** | **Implementation Layer** | **Technologies/ Testbed** | **Maturity Level** |
|---|---|---|---|---|
| [79], 2019 | ✓ | network and application | Sample Use-Case | simulation |
| [104], 2018 | ✓ | network and application | ONOS [105] | emulation |
| [57], 2018 | | network | Personalized use cases | simulation |
| [58], 2018 | | abstract | Personalized use cases | simulation |
| [92], 2018 | ✓ | network and application | Personalized use-cases | simulation |
| [88], 2018 | ✓ | network | Personalized test-bed with VMs | emulation |
| [106], 2018 | | application | Polyverse | commercial |
| [33], 2018 | | network and application | Personalized Test-bed with VMs | simulation |
| [95], 2018 | | network | personalized use-cases | simulation |
| [107], 2018 | | network and application | Deception Grid [107], Crypto Trap [108] | commercial |
| [77], 2018 | | network and application | Mininet | both |
| [109], 2017 | ✓ | network | Opendaylight Helium [110] | simulation |
| [111], 2017 | ✓ | network | Ryu SDN [112] | simulation |
| [113], 2017 | | application | LLVM compiler [114] | emulation |
| [64], 2017 | | application | personalized use-cases | simulation |
| [115], 2017 | | application | CMS application [116] | simulation |
| [117], 2017 | | application | Simpy [118] | simulation |
| [73], 2017 | ✓ | network | network simulator + ODL controller | emulation |
| [60], 2017 | | abstract | Personalized Test-bed with VMs | simulation |
| [56], 2017 | ✓ | application | Mininet with POX controller | emulation |
| [86], 2017 | ✓ | network | CloudLab [119] | emulation |
| [120], 2016 | | network | Smart grid, energy management system (EMS) | emulation |
| [64], 2017 | | application | personalized test-bed | simulation |
| [121], 2017 | | network and application | Morphisec | commercial |
| [59], 2017 | | network | Personalized Simulation Environment | simulation |
| [55], 2016 | ✓ | network | Mininet with Floodlight controller | emulation |
| [93], 2016 | | network | Personalized use-cases | simulation |
| [51], 2016 | ✓ | abstract | OpenStack [122] with ODL [110] controller | emulation |
| [123], 2016 | | network | OpenStack [122] | emulation |
| [68], 2016 | ✓ | network | GENI [124] | emulation |
| [76], 2016 | | network | Rocketfuel Dataset | simulation |
| [125], 2016 | | network | ARCSYNE [126] & SDNA [127], [128] | emulation |
| [87], 2016 | ✓ | network and application | Mininet with POX controller | simulation |
| [129], 2016 | ✓ | network | CyberVAN | commercial |
| [68], 2016 | ✓ | application | GENI [124] | emulation |
| [69], 2015 | | abstract | Personalized simulation environment | simulation |
| [130], 2015 | | network and application | Cyber Quantification Framework | emulation |
| [74], 2015 | | network | matlab | simulation |
| [94], 2015 | | network | personalized use-cases | simulation |
| [131], 2015 | | network and application | RPAH framework | simulation |
| [66], 2014 | | network and application | CORE Impact Pro | commercial |
| [65], 2014 | | application | personalized test-bed | simulation |
| [132], 2014 | ✓ | network | Personalized Cloud System | emulation |
| [133], 2013 | | network and application | PlanetLab [134] | emulation |
| [87], 2013 | | network and application | Personalized virtual cloud system | emulation |
| [85], 2013 | | abstract | Personalized test-bed | simulation |
| [63], 2013 | | application | game-based theoretical framework | analytic |
| [54], 2012 | ✓ | network | NOX in Mininet | emulation |
| [70] , 2012 | | abstract | Proposes to use a modified CTF environment | analytic |
| [71] , 2012 | | network and application | Personalized test-bed | emulation |
| [75], 2012 | | network | KDD dataset [135] | simulation |
| [52], 2012 | ✓ | network | Personalized use cases | simulation |
| [136], 2011 | | network and application | SLAAC [137] | simulation |
| [138], 2011 | | network | MUTE | simulation |

scanning tools make use of ICMP, TCP or UDP scans to identify the connectivity and reachability of the potential targets. The replies to the scans can also reveal the firewall configuration details, i.e., what traffic is allowed or denied. The time to live (TTL) information can also help in the identification of a number of hops to the attack target [132].

SDN-enabled devices can help in delaying the network attack propagation by hiding the real response and replying back with a random response in order to confuse the attacker. As a result, the attacker will believe that random ports are open in the target environment. The attack cost will be increased since the attacker will need to distinguish the real reply from the fake reply. SDN-enabled devices can also introduce random delays in TCP handshake request that will disrupt the reconnaissance process utilized by the attacker for identification of TCP services. The cost-benefit analysis of MTD adaptations against network mapping attempts has been discussed by Kampankis *et al.* [132]. The survey considers cost-benefit aspects of MTD in Section V, under the quantitative metrics of MTD.

*2) Service Version and OS Hiding:* The attacker needs to identify the version of OS or vulnerable service in order to mount an attack. For instance, the attacker can send *HTTP GET* request to Apache Web Server, and the response can help in identification of vulnerability associated with a particular version of the Apache software. If the attacker gets a reply 404 Not Found, he can identify some obfuscation happening at the target software. A careful attacker can thus change the attack vector to exploit the vulnerability at the target.

An SDN-enabled solution can override the actual service version with a bogus version of the Web server. Some application proxies leverage this technique to prevent service discovery attempts by a scanning tool. Another attack method is known as Operating System (OS) Fingerprinting, where the attacker tries to discover the version of the OS which is vulnerable. Although modern OS can generate a random response to TCP and UDP requests, the way in which TCP sequence numbers are generated can help an attacker in the identification of OS version.

In an SDN-enabled solution, the OS version can be obfuscated by the generation of a random response to the probes from a scanning tool. SDN can introduce a layer of true randomization for the transit traffic to the target. The SDN controller can manage a list of OS profiles and send a reply resembling TCP sequence of a bogus OS, thus misguiding the attacker.

*3) Protection Against Multi-Stage Attacks and Service Disruption:* Once the attackers have obtained necessary information, they proceed towards targeted attacks, with the aim of stealing information - SQL Injection or service disruption - Distributed Denial of Service (DDoS). The SDN-based MTD can introduce various countermeasures at network-level such as network shuffling [109], route modification [104], IP, and port obfuscation as discussed by Shi *et al.* [86].

We now discuss some case studies which show use of SDN/NFV capabilities for implementation of MTD techniques - what, when, and how to switch? Jafarian *et al.* [54] use OpenFlow based random host mutation technique to switch virtual IP (what to switch) targeted by reconnaisance attempts. Debroy *et al.* [68] use SDN framework to identify optimal rate of migration (when to switch) and ideal migration location for VMs under DoS attack. Chowdhary *et al.* [73] use SDN-environment to create a analyze the hosts mounting DDoS attacks on critical network services. The SDN-controller downgrades network bandwidth (how to switch) using a Nash-Folk theorem based punishment mechanism.

► **Case Study (What to Switch)– OpenFlow Random Host Mutation.** SDN makes use of OpenFlow protocol for control plane traffic. Jafarian *et al.* [54] propose OpenFlow enabled MTD architecture as shown in Figure 7, can be used to mutate IP address with a high degree of unpredictability while keeping a stable network configuration and minimal operational overhead.

The mutated IP address is transparent to the end host. The actual IP address of the host called real IP (rIP), which is kept unchanged, but it is linked with a short-lived virtual IP address (vIP) at regular interval. The vIP is translated before the host. The translation of rIP-vIP happens at the gateway of the network, and a centralized SDN controller performs the mutation across the network. A Constraint Satisfaction Problem (CSP) is formulated in order to maintain mutation rate and unpredictability constraints. The CSP is solved using Satisfiability Modulo Theories (SMT) solver.

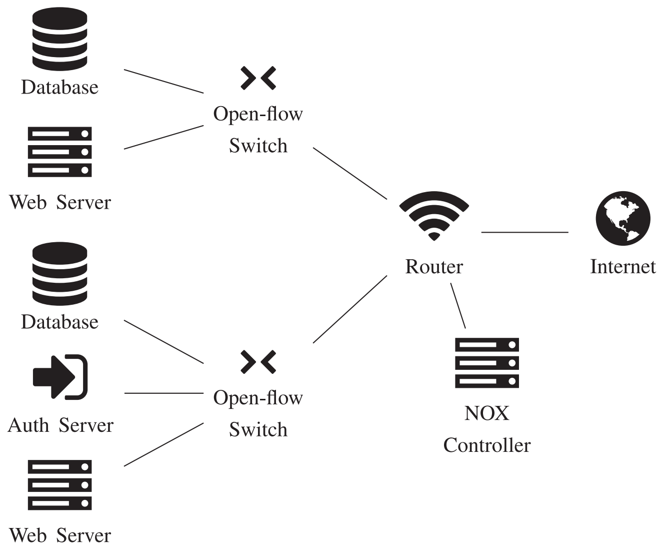


Fig. 7. OpenFlow-based Random Host Mutation mutates the virtual IPs of hosts based on a pool of available IP addresses.

Sensitive hosts have a higher mutation rate compared to the regular hosts in this scheme. The OF-RHM is implemented using Mininet network simulator and NOX SDN controller. OF-RHM is able to thwart about 99% of information gathering and external scanning attempts. The framework is also highly effective against worms and zero-day exploits.

► **Case Study (How to Switch)– Dynamic Game-based Security in SDN-enabled Cloud Networks**. DDoS attacks are a major security threat affecting networks. Attacker's leverage sophisticated bots to generate a huge volume of network traffic, and overwhelm critical network services as discussed by Chowdhary *et al.* [73]. The case study presented in this research work, focused on the MTD decision *how to move?* The framework presented in Figure 8 (a) to communicate with OpenFlow devices using southbound REST API, whereas any application plane decision and network analytic is performed using northbound REST API.

The Snort IDS [142] detects malicious DDoS patterns, and uses Nash Folk Theorem [143] to analyze the behavior of a malicious node, e.g., red color VM shown in Figure 8 (a). The scenario is represented as an extensive-form game spanning multiple rounds. If the node sending network traffic P2, the Defect - Figure 8 (b), the normal bandwidth B is reduced to $\frac{B}{2}$. The bandwidth is reduced in each subsequent round by network admin P1, till average bandwidth over a period of time $t = \{0, 1, 2, \ldots\} \in T$ is B.

The SDN controller utilizes Instruction field present inside the flow table in order to change the band rate of the node flagged as malicious, as shown in Figure 8 (c). Thus

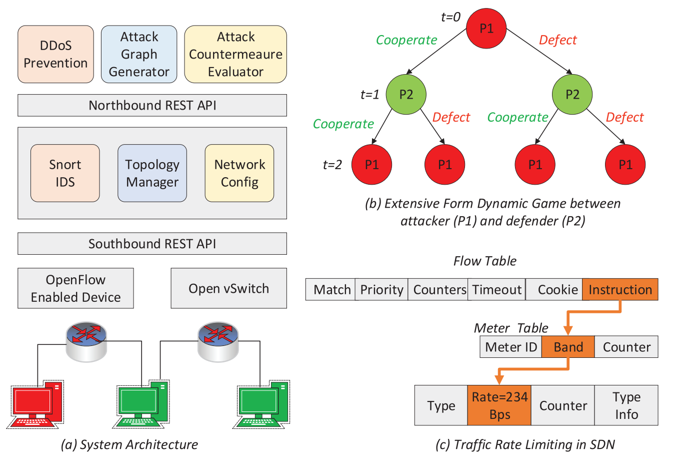


Fig. 8. SDN/NFV based MTD that models the interaction between an attacker and the defender as a game to select an optimal countermeasure strategy. The work imposes limiting on bandwidth and views the MTD as an adaptation problem.

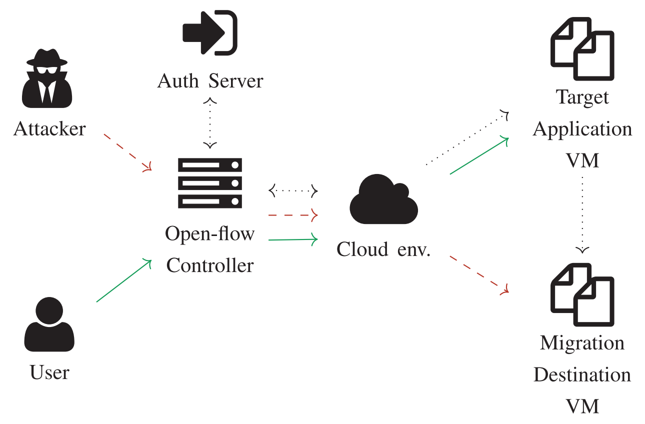


Fig. 9. Virtual Machine migration using frequency minimal MTD seeks to migrate applications to new VMs and redirect an attacker's DDoS attacks without effecting availability to regular users. The goal of this work was to find an optimal timing function $T$, termed as the migration frequency.

SDN-based rate-limiting serves as an effective countermeasure against flooding/loss of availability attacks like DDoS.

► **Case Study (When to Switch): Frequency Minimal MTD using SDN.** *Frequency minimal MTD* [68] approach considers resource availability, QoS and exploits probability for performing MTD in an SDN-enabled cloud infrastructure. The design goals of this research work are as follows:

1) Identification of optimal frequency of VM migration, ensuring minimal wastage of cloud resources.
2) Finding the preferred location of VM migration without impacting the application performance.

As shown in Figure 9, the normal clients can access the services hosted in the cloud network via regular path, and the attack path represents the path along which the attacker tries to exploit the target application. The VMs periodically share their resource information such as storage and compute capacity with the controller along the control path. Depending upon the level of threat, the migration of VM can be proactive or reactive. The real IP address of the VM hosting cloud application is hidden from the clients. The data-path shows the path along which VM application and its back-end database are migrated. VM migration is based on the following factors:

- *VM Capacity:* This parameter considers the capacity of migration target in terms of computing resources available to store files and database of current and future clients.
- *Network Bandwidth:* The lower the network bandwidth between the source and target VM, the slower will be the migration process and the longer will be the exposure period (in case of active attacks) for the VM being migrated. This parameter considers bandwidth between source and target while performing VM migration.
- *VM Reputation:* This is the objective indicator of VM robustness to deter future cyber attacks. It is the history of VM in terms of cyber attacks launched against the VM. This parameter is considered in order to ensure VMs suitability for migration.

This research work estimates the optimal migration frequency and ideal migration location based on the parameters described above. The VM migration mechanism is highly effective in dealing with denial-of-service (DoS) attacks.

## D. *MTD Testbeds: Research and Prototyping*

The practicality of MTD can be established by the deployment of MTD techniques and tactics over an underlying network. In this section, we identify some platforms which can assist MTD researchers in conducting experiments or serve as a guideline when creating MTD case studies.

*1) GENI Platform:* Global Environment for Networking Innovation (GENI) [124] provides a distributed virtual environment for at-scale networking and security experimentation. Individual experiments can be conducted in isolation using network sliceability (ability to support virtualization and network isolation). Each experimental slice is provided with network and computing resources. The users can request resources for an allotted time period, in which they can conduct experiments and release the resources once they have finished the experiment.

Debroy *et al.* [68] utilize GENI framework for implementation of a VM migration MTD solution. The authors utilized the InstaGENI platform at the Illinois site to host a news feed application targeted by DoS attacks. The setup also involved four non-malicious users, one remote SDN controller, and attacking VMs, all hosted at physically distributed locations. The attacker VMs were utilized for sending a large number of HTTP GET requests to news feed site in order to achieve a DoS attack. Proactive and reactive frequency minimal (FM) MTD countermeasures were deployed in response to targeted DoS attacks. The research work shows the capability of GENI platform to host similar MTD experiments where users can study the impact on network bandwidth, VM performance, attack success rate when MTD security countermeasures are implemented at scale on a cloud platform.

*2) OpenStack Cloud:* OpenStack [122] is a cloud operating system that consists of compute, storage, and networking resources. The users can log in through GUI to provision isolated virtual networks or utilize OpenStack APIs to create

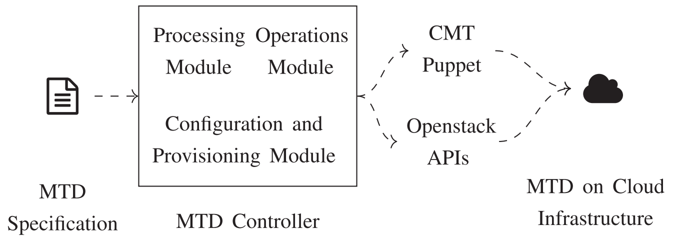


Fig. 10. A platform that takes an abstract specification of a cloud system as input and outputs the corresponding system on a cloud infrastructure. Advantages of cloud automation using Automated Enterprise Network Compiler (ANCOR) include performing live instance migration.

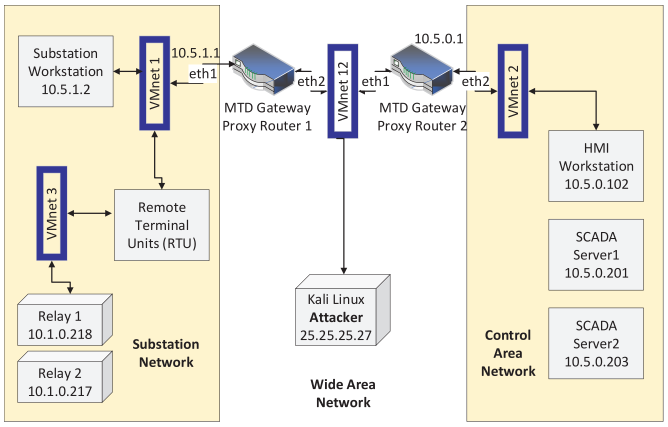


Fig. 11. An MTD-based architectural platform for IP hopping. The solution has been targeted at Supervisory Control and Data Access Systems (SCADA). The MTD gateway routers act as dynamic proxies, periodically mutating the IP address of the external interfaces exposed on the public wide-area network.

and configure the network. OpenStack is compatible with the existing virtual solutions such as VMWare ESX, Microsoft's Hyper-V, KVM, LXC, QEMU, UML, Xen, and XenServer.

**Mayflies** [123] utilized OpenStack for designing a fault-tolerant MTD system. The research work utilizes VM introspection to checkpoint the current state of the live node/VM, and reincarnation - node commission/decommission based on attacks against the introspected node. The strategy allows the Mayflies framework to deal with attacks in a short interval of time, avoiding the attack progress. Zhuang *et al.* [144] conduct MTD experiments to simulate a pure-random MTD strategy and an intelligent MTD approach based on intelligent attack indicators. The experiments were conducted using *NeSSi2* [145], an open-source, discrete event-based network security simulator. The authors proposed implementation on an OpenStack based MTD testbed as future work.

**MTD CBITS** MTD-platform for *Cloud-Based IT Systems* (MTD CBITS) [146] as shown in Figure 10 evaluates the practicality and performs detailed analysis of security benefits when performing MTD on a cloud system such as OpenStack.

The platform makes automated changes to the IT systems running on the cloud infrastructure. One adaptation performed in MTD CBITS is replacing the running components of the system with new ones. The system parameters are stored in the operational model and can be viewed using the MTD system inventory - CMT (Puppet) APIs. Any MTD adaptation is also recorded in an operational model for future reference. OpenStack API utilizes Puppet agents to communicate with the live instances of the cloud system. MTD controller communicates with agents over a private network.

*3) CyberVAN Testbed:* Reference [129] provides a testing and experimentation platform for cybersecurity research. The platform supports high fidelity network, by representing the network in a discrete event simulator. CyberVAN allows hosts represented by VMs to communicate over the simulated network. The testbed utilizes network simulator *ns-3* [147] to simulate network effects such as end-to-end network latency, link capacities, routing protocols, *etc.* The platform also supports wireless networks by modeling mobility, interference, propagation effects, as well as the details of different waveforms. The Scenario Management GUI allows the experimenter to access and manage the elements of an attack scenario, including network traffic and logging, running analytic tools on the VM, saving results of the experiment, pausing and restarting the experiments, *etc.*

*4) MTD SCADA Testbed for Smart Grid Network:* In the context of smart-grid environments, Pappa *et al.* has proposed an MTD for IP hopping [120] (see Figure 11). The network consists of two SCADA servers running Siemens Spectrum Power TG Energy Management Systems (EMS) software. The software periodically polls Remote Terminal Units (RTUs) for network traffic measurements. The substation networks are connected to the Wireless Area Network (WAN) using MTD gateway routers. The remote attacker can utilize the publicly exposed service over the WAN to identify security vulnerabilities. The services can be exploited using traditional means or using APT techniques. The gateway routes in the network act as dynamic proxies, mutating IP address of the external interfaces, while providing end-to-end SCADA communication. The IP generation algorithm employs a random initial seed to initiate the IP generation at the network boot up. Both routers are assigned set $(j, k)$ of $n$ random IP addresses at the start, in combination of initial seed vector. The initial seed vector ensures order between the two gateway routers is synchronized; this helps to prevent IP collision and network outages. Additionally, the MTD algorithm used creates a dynamic network technology which makes the job of network mapping difficult for the attacker.

*5) MTD Commercial Domain Solutions:* **TrapX:** The production-grade decoy network technologies such as *DeceptionGrid* [107] and *CryptoTrap* [108] deceive the attackers by imitating the true assets. DeceptionGrid provides agent-less visibility and control appliance, that dynamically identifies and evaluates network endpoints and applications when they connect to the main network. Assuta Medical Center incorporated DeceptionGrid as a part of their network security suite. The use of this framework helps to counter not only known attacks but also APTs and zero-day attack scenarios. The DeceptionGrid created a network of traps (decoys) that camouflaged real medical devices such as MRI & CT scanners, X-ray machines, and ultrasound equipment (PACs). The solution has been deployed on many VLANs across Assuta Medical Center and provided better visibility into the lateral movement of the attackers.

*CryptoTrap* on the other hand is utilized for countering the ransomware early in their exploitation lifecycle. This helps in countering malware propagation while protecting valuable network assets. The traps (decoys) are masked in the form of SMB network shares across the network. The fake data of the company is replicated across the traps. Once the attacker touches these traps, the targeted computer is disconnected from the network and security administrator is alerted about the incident. In effect, the attacker is held captive in the trap and attacker actions are logged for further threat intelligence.

**Polyverse:** *Zero-day* vulnerabilities, e.g., *Heartbleed* [148] (vulnerability discovered in *OpenSSL* software in 2014), can cause a significant damage in an underlying network. There is no known attack signature to identify these vulnerabilities, hence they can bypass security monitoring software unnoticed. The attackers trying to target software or OS memory-based zero-day vulnerabilities start with the assumption that the gadgets they are trying to access are located at a certain address or within a specific offset from the absolute base address. *Polyverse* [106] employs MTD strategy to defeat this assumption of the attackers. The polymorphic version of Linux is utilized by Polyverse in order to create a high-level of entropy in the software system in such a way that the entire memory structure is diversified. With the polymorphic versions of Linux, the entire Linux software stack is roughly randomized. The resulting program is semantically identical to the original program (functionally equivalent), however, nearly every machine instruction is changed.

**MorphiSec:** Some threat vectors classified as Advanced Evasive Attacks cloak the malicious intent in order to deceive the security monitoring tools. Some of these techniques include Polymorphism (changing malware signature), Metamorphism (changing malware code on the fly), Obfuscation (hiding malicious activity), behavior changes (waiting for normal user activity before executing). *Morphisec* [121] uses MTD at network-level (route changes, random addresses and ports), firewall level (policy changes), host-level (OS version, memory structure changes), and application level (randomizing storage format addresses, application migration, multilingual code generation) in order to deceive and detect attacks using evasive attack techniques.

## V. Evaluating the Effectiveness of Moving Target Defenses

In this section, we present a list of qualitative and quantitative metrics that can help in determining the effectiveness of MTD. First, we discuss the quality of the defender actions $C$, i.e., the various system configurations it can choose to deploy, in terms of performance and security. This discussion helps in identifying a set of qualitative metrics. As we will see, most works either consider this metric implicitly when choosing the configuration set or simply ignore it. Second, we devote a sub-section on qualitative evaluation metrics. These are mathematical functions to help a defender capture measures such as the cost-benefit, the risk analysis, *etc.* of an MTD. As we shall see, they can be measured by simulation on test-beds or by using metrics based on security domain knowledge such as CVSS, attack graphs representations *etc.*

### A. Qualitative Evaluation

When using a Moving Target Defense (MTD) that moves between multiple system configurations, we would want to believe that it increases the security of the deployed system without negatively impacting the performance for legitimate users. Thus, we look at how different MTDs consider these aspects either in the modeling of the defense or its evaluation. In particular, we look at two major aspects– security considerations and performance considerations– to evaluate the quality of an MTD. Under each of these sub-headings, we consider the quality of each individual defense and then, the overall ensemble of constituent defenses. This helps us identify non-trivial heuristics to understand when an MTD might succeed or fail. For example, an MTD that lacks *differential immunity*, i.e., all the constituent system configurations ($\in C$) are all vulnerable to the same attack, can never be secure regardless how well it is implemented in practice.

In regards to quality metrics, we categorize the various MTDs in Figure 12. An MTD is part of a set if they explicitly consider the particular qualitative metric when designing either the configuration set $C$ (i.e., *what to move?*), the timing function $T$ (i.e., *when to move?*), or the movement function $M$ (i.e., *how to move?*). Note that only a subset of works discussed under security considerations (i.e., shown in the blue and/or yellow boxes on the left of Fig. 12) are featured under performance considerations (i.e., shown in the pink and/or green boxes on the right of Fig. 12). We discovered that majority of the defenses proposed only consider (and therefore highlight) the improvement with regards to security and ignore the impact on performance. We will now discuss the details about how the different works capture the various qualitative aspects that we put forth in this survey.

*1) Security Considerations:* We first look at works that reason about the security risks of individual actions followed by works that reason about the security risks associated with an ensemble and lastly, discuss works that consider both.

*a) Considers only security of individual defenses:* Most MTDs that consider some form of game-theoretic modeling, consider the security of individual defenses in the ensemble by representing them as a part of the defender's utility value [58], [60], [65], [77], [79], [91], [150]. For most of these works, the security metrics considered for each action are obtained using scoring metrics designed by experts such as the Common Vulnerability Scoring Services (CVSS) discussed earlier. These works are able to come up with an intelligent movement strategy $M$ based on reasoning over known vulnerabilities (more specifically, common vulnerabilities and exposures (CVE)) for which CVSS scores are readily available. This can result in highly sub-optimal behavior when faced with zero-day vulnerabilities. In that regard, authors in [65] model the security of a configuration as inversely proportional to the probability with which an adversary can come up with a new attack given the attacks it performed in the earlier time steps. In [150], the authors try to model zero-day

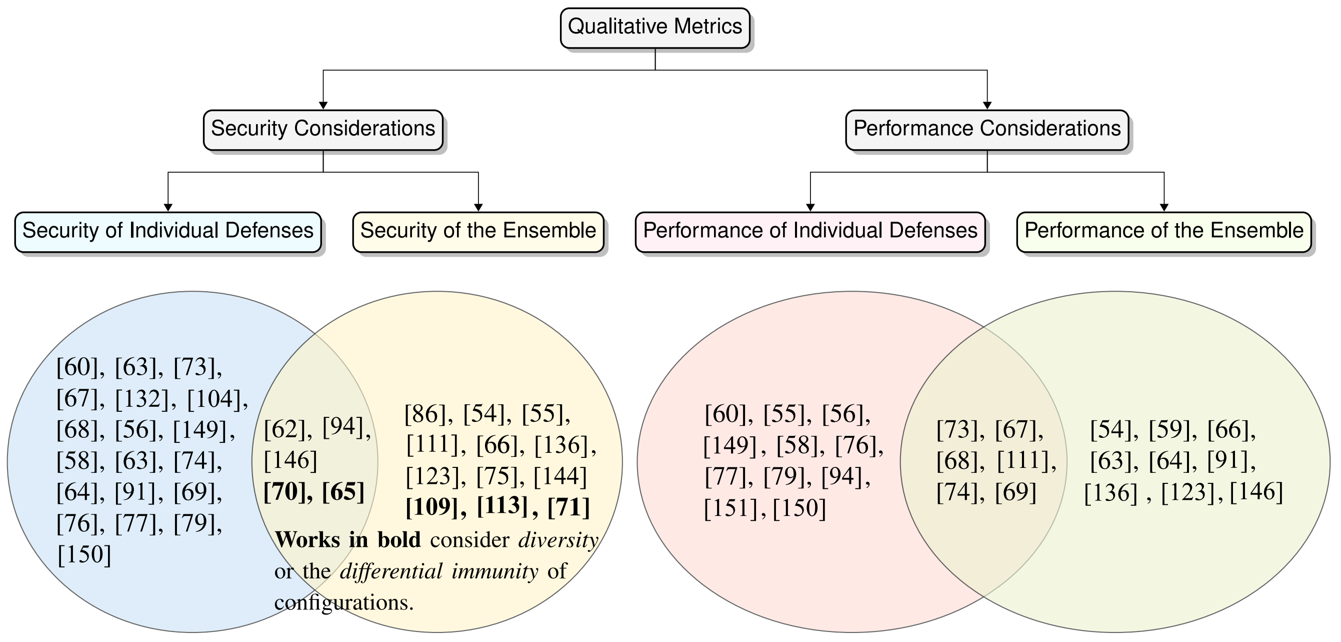


Fig. 12. A Moving Target Defense is more effective if it considers the security and performance impacts of the system configurations, both in unison and also when used as part of an ensemble that the MTD leverages. We categorize the works which explicitly consider these metrics either while modeling the defense or during its evaluation. Note that there is no work which considers all four kinds of metrics, i.e., falls in the intersection zone both on the left and on the right.

attacks by asking security experts to annotate how effective they think a particular countermeasure or defense action will be against zero-day vulnerabilities. As the annotations might be inaccurate (due to the lack of actual black-hat hackers who invent zero-day attacks in the set of security experts who annotate the defense methods), the effectiveness of their MTD cannot be accurately determined. As to how, if possible, utility values can capture the security risks associated with zero-day vulnerabilities is an interesting question and remains to be explored by future works.

Other works that also only capture the security of individual constituent system configurations are more domain or problem specific in nature. In [73], each defense action, before being played, is weighed based on the penalty it imposes on an attacker (who tried to do a DDoS attack) over a repeated game setting. In [67], authors choose an action (to migrate a VM or keep running) after reasoning about the security risks associated with the present vulnerable state. In [68], the security risk is based on the reputation of the current state, which in turn looks at the type and number of attacks that were done in previous time steps when the particular system was deployed. On similar lines, researchers in [63] find a more compact way to model and continuously update the risk associated with deploying a particular defense action. Some works that consider the longitudinal effects of a movement strategy $M$ model the epistemic effects of movement on an attacker's knowledge about the network. For example, [56] considers the fingerprint of the overall network when a particular defense action is selected, while [149] reasons about the topology information a particular defense action leaks to an attacker. In contrast to all these works, [132] models the security risk associated with a particular defense action as inversely proportional to the cost (it believes) an attacker would spend to compromising it.

*b) Considers only security of the ensemble:* Works that measure only the security of an ensemble as, opposed to security of actions, mostly showcase the security benefits of the MTD ensemble by comparing them to the security benefits provided by a static system configuration [54], [55], [66], [86], [144]. In essence, these works consider the static system as the control case but unfortunately, do not ensure that this static system is the most secure constituent defense configuration in the ensemble. This way, they might overestimate and thus, over-promise the security guarantees of the designed MTD.

On the other hand, works that consider the security metrics associated with the ensemble (as a whole) at modeling time look at metrics that are similar to *entropy* or *diversity* of the ensemble. In [136], the authors try to use the large address-space of IPv6 to their advantage and are able to create more uncertainty for an attacker who is trying to pinpoint the address to use for a successful attack. On similar lines, authors in [123] argue that fast reincarnations of a functional system (i.e., bringing the service down and starting it up on a new location/container, *etc.*) increases the entropy and makes it more costly for an attacker to continuously keep attacking such a system. On a different note, researchers in [109] try to select a defense configuration from the ensemble based on how diverse it is with respect to the current configuration that might have been attacked. In order to do this, they use a topological distance measure, which in their case is the symmetric difference between the edge sets of the current and the consecutive defense configuration. Although they do not explicitly recognize it as a diversity metric like [71], [113], they bring to light an interesting issue that most MTD papers seem to either miss or assume by default. If there was an attack, that with extremely little modification, could exploit all the defender's configuration that is a part of the MTD, an MTD would not be an effective defense strategy. For example, MTD work that randomly selects a classifier for malware detection assumes by default that each classifier is not vulnerable to the same attack [75] which seems to be an incorrect assumption to make given current research works [30], [80]. We believe that it would be easier to convince practitioners about the effectiveness of an MTD if researchers can show that diverse defender actions can indeed be generated at a low cost. This is the goal of [113], who create an approach at the compiler level to increase software diversity and [71], who use a genetic algorithm to draw a pool of configurations that maximize diversity.

*c) Considers both:* Only a few works take a holistic view in regards to security and consider both the security of each individual defense and the ensemble as a whole. In [62], the general level at which they define the utility values, one can capture the security risks associated with an individual defense action in terms of attack surface features and attack surface measurements while the security risks associated with an ensemble can be captured via the utility variable for attack surface shifting. In [146], the security for individual defense actions is trivial for their settings, where defending a node that is not a stepping stone is not beneficial at all for the defender. The security risks associated with an ensemble are evaluated through experimentation with a static defense. Particularly, the try to increase the number of interruptions for an attacker to start from one point in the network and reach a goal node. In [94], the selection of the entire ensemble is based on the fact that the set of possible defenses have a global property of covering each of the leaves that an attacker might want to reach and each individual defense action has some utility in terms of security associated with it. Lastly, the works [65] and [70] model both the diversity of constituent defenses present in the MTD ensemble and also model the security of each individual defense. In [70], each configuration is mapped to a set of vulnerabilities and thus, a diverse ensemble is composed of system configurations that do not have a lot of overlapping vulnerabilities. In [65], since they use Linux based operating systems, the diversity is modeled in terms of lines of code that is different between two Operating Systems.

*2) Performance Considerations:* As mentioned above, a large set of works for developing MTDs focus on showcasing the security benefits and sweep under the rug the performance costs. Note that in the case of MTDs, the impact on performance may arise due to a variety of reasons. First, each system configuration (individual defense action), that is a part of the MTD ensemble, has a performance cost associated with it and moving to a high-cost configuration impacts the performance. These concerns are termed as performance considerations for individual defense. Second, the switching from one configuration to another may need a defender to deal with (1) downtime, (2) legitimate requests on the wire that were meant for the previous configuration in a graceful manner and/or (3) keep all the different configurations running (at least two of them) to facilitate a faster switch. All these costs can be termed as shuffling costs and are categorized as performance considerations of the ensemble because they only arise when there is an MTD ensemble.

We will now describe how the various MTD systems consider the performance costs associated with each individual configuration followed by the performance cost associated with the ensemble. We also discuss works that reason about both these factors together and analyze the affect on the overall Quality of Service (QoS) when such defenses are is deployed.

*a) Considers only performance of individual defense actions:* Similar to the case of security metrics, when we look at performance considerations for an individual defense, most game-theoretic works model these as a part of the defender's utility function [60], [76], [77], [79], [95]. Various equilibrium concepts in these games yield movement strategies for the defender that gives priority to constituent defenses that have low performance costs while ensuring that the security is not impacted by a lot. In [60] and [95], the reward functions are defined at an abstract level and the authors point out that they can be used to consider the performance cost of constituent defenses. In [76] and [77], the authors consider the impact of placing Network-based Intrusion Detection Systems (NIDS) on the latency of the network and use centrality based measures as heuristic guidance for it.

A large number of works also look at problem-specific instances. Authors in [55] perform experiments to evaluate their MTD against crossfire attacks and find that the average time for packet transfer from one point to another increases because a particular path selected at random can be highly sub-optimal. On similar lines, in [56], the authors consider the performance cost of doing a *defend* action on legitimate user traffic. On a different note, [149] tries to solve a multi-faceted problem where the MTD tries to obfuscate the network topology to an attacker and, at the same time, ensures that it does not negatively impact a defender's ability to debug network issues. This is done by leveraging the knowledge asymmetry about the network topology that a defender and an attacker has. In [58], the performance costs of a *honeynet* configuration, which is the defense action for this MTD, represents the number and *quality of resources* (or honey) necessary for developing a credible honeynet that fools an attacker. Lastly, authors in [150] combine both the costs of setting up a good defense and the usability of that defense for legitimate traffic as the performance cost for a particular placement of countermeasures.

*b) Considers only performance of the ensemble:* Multiple works try to capture the performance costs that result because of the dynamics involved is shuffling between the different configuration but do not look at the *performance impacts* resulting because of a particular *bad constituent configuration*. In [63] and [64], the authors consider the *one-step cost* of switching from one defense action to another one and seek to find a strategy that minimizes this.

Other works, instead of accounting for the performance impact of the ensemble, compare an MTD defense to a static system configuration by measuring usability metrics such as latency, availability to legitimate users, *etc.* In [136], authors notice an overhead of 40 bytes for the IPv6 header and latency of 12ms during address change as opposed to 3ms when MTD is not implemented on the network packets. They point out that a more efficient implementation might help in reducing this gap. On similar lines, authors in [123] highlight that creating an entire file system replica takes two more minutes in the case of an MTD system in comparison to a non-MTD enabled system refresh. On the contrary, [146] does not notice negligible performance overhead for instance replacements in a 14 node system (where one is a controller node and the others are compute nodes). Although, they do notice a larger number of HTTP error packets on the wire when using the MTD. To ensure that the usability to legitimate users is not impacted at all, authors in [66] keep multiple systems running with the different configurations. At every switch, they simply pick the system that serves the request. On the downside, they incur the cost of maintaining multiple services (at least two). An interesting side effect of using MTD in [59], mainly done to thwart selective jamming attacks, is the reduction of delay in transmitting packets over the network. Existing approaches that schedule packet delivery in a deterministic way land up in packet collision scenarios. The random schedule selection in each round is shown to reduce the number of collisions, improving end-to-end (ETE) packet delivery time.

*c) Considers both:* A small section of works either consider or evaluate their MTD in regards to both performance of single constituent defenses and the ensemble. In [73], the authors model the performance costs associated with each defense action as a part of the utility values and consider the shuffle cost as a part of the rewards the attacker and a legitimate user gets in a repeated game setting. On similar lines, in the empirical analysis of MTDs in the context of Flip-it games [69], the authors model the cost of each defender action as a part of defender's utility value while the state of the system after a move action considers the number of servers being controlled by the defender– an indirect way of measuring system performance. Authors in [67] consider the availability of each defense action over a *bounded time horizon* and for *shuffling costs*, consider the downtime or unavailability that results when the system is migrating from one system configuration to another. On similar lines, the authors of [68] model the *host capacity* and *network bandwidth* associated with each system configuration and also determine the next configuration based on the performance of other defenses in the previous time steps. For some works, the performance costs associated with the ensemble do not arise due to the shuffle but represent the cost of ensuring exact same performance across defense configurations [111] or cost of implementing a system that can support the various configuration [74]. More specifically, authors in [111] create a system that ensures that the logical virtual identifier distances between any two nodes remain the same, while in [74], the authors assume an extra cost of creating an ensemble that is not necessary when not using MTD solution.

## B. Quantitative Metrics

Although we realize that the quality of MTD is key is determining the quality of defense being offered, quantifiable measures about these qualitative terms are essential for effective evaluation. For example, shuffling $x$ % number of hosts leads to $y$ % reduction in attack success probability, and $z$ % increase in overhead/quality of service (QoS) for the normal user. In this sub-section, we present the quantitative analysis of MTD research works discussed in the survey. We categorize quantitative analysis into *Security Metrics* and *Usability Metrics* as shown in Figure 13.

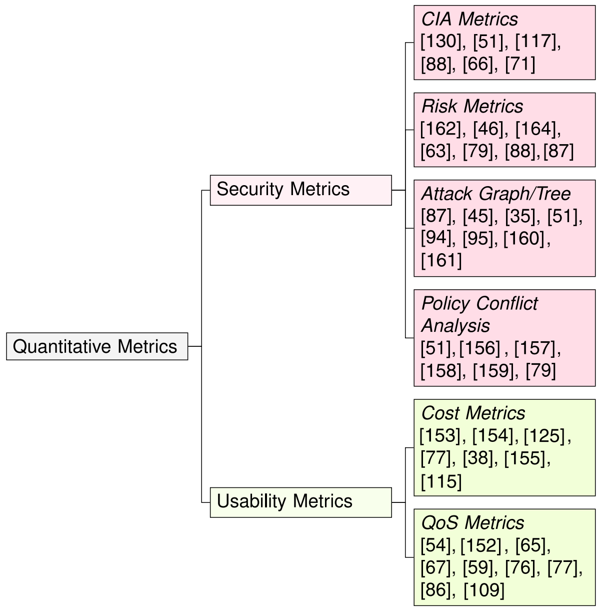

Quantitative Metrics
Security Metrics
Usability Metrics
CIA Metrics [130], [51], [117], [88], [66], [71]
Risk Metrics [162], [46], [164], [63], [79], [88], [87]
Attack Graph/Tree [87], [45], [35], [51], [94], [95], [160], [161]
Policy Conflict Analysis [51], [156], [157], [158], [159], [79]
Cost Metrics [153], [154], [125], [77], [38], [155], [115]
QoS Metrics [54], [152], [65], [67], [59], [76], [77], [86], [109]

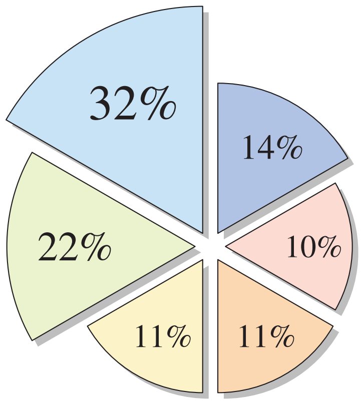
Fig. 13. Categorization of MTDs based on the fine-grained quantitative metrics, under the broader umbrella of security and usability metrics, they use for evaluation.

*1) Security Metrics:* An important aspect of network defense is representation and visualization of network attacks. Enterprise networks are becoming large and complex with different network overlay and underlay technologies. The adage *what can't be measured cannot be effectively managed* applies aptly here. Security metrics such as the ones shown in the Figure 13, covers attack quantification using CVSS metrics– Confidentiality, Integrity, and Availability (CIA). This metric also considers attack representation methods (ARMs)– Attack Graphs and Attack Trees. We now discuss the MTD research works from this perspective of security metrics.

We surveyed ∼70 MTD research articles and analyzed them from the perspective of different threats models they targeted. A summary of the findings has been provided in Table VI. The key observations are as follows a) 32% of research works focus on defense against reconnaissance attempts, b) 22% of these research works target vulnerability exploitation, c) 11% research works use MTD defense against DoS/DDoS attacks. There has been a limited focus on using MTD defense for dealing with stealthy attacks like APT and data-exfiltration (7/68 ∼10% research works). Also, some papers used other types of attacks such as timing-based attacks [74], and network intrusions [75], [77], [87], [88].

**CIA Metrics:** *Confidentiality*, *Integrity*, and *Availability* (CIA) are used as quantitative metrics for measurement of impact on system under attack. Zaffarano *et al.* [130] use

TABLE VI
THREAT MODELS CONSIDERED IN THE MTDS PROPOSED IN THE SURVEYED PAPERS



| | |
|---|---|
| **Reconnaissance** | [54] [52] [118] [8] [9] [56] [154] [57] [62] [69] [71] [86] [89] [132] [139] [94] [95] [96] [133] [110] [112] [137] |
| **Vulnerability Exploitation** | [92] [64] [65] [67] [70] [66] [72] [96] [165] [116] [166] [124] [147] [167] [114] |
| **(D)DoS Attack** | [105] [55] [9] [68] [73] [132] [139] [154] |
| **Multi-Stage Attack** | [161] [79] [51] [60] [93] [166] [168] [145] |
| **APT and Data Exfiltration** | [85] [126] [33] [169] [151] [126] [76] |
| **Other types of Attack** | [74] [89] [88] [75] [77] [126] [76] [80] [53] |

confidentiality metric for measuring information exposed by modeling tasks. The mission *M* confidentiality valuation *v* is expressed as, $Conf(M, v) = \frac{1}{|T|}\sum_{t\in T}(t, unexposed)$. As the measurement equation suggests, the goal is to maintain information as unexposed over time ($t \in T$). Similarly *Integrity* valuation *v* for mission *M* is quantified as $Int(M, v) = \frac{1}{|T|}\sum_{t\in T}(t, intact)$. High integrity is ensured by keeping information intact over time.

Conell *et al.* [117] consider availability as an important metric for analyzing impact of MTD countermeasure. The system reconfiguration rate $\alpha$ is modeled as a function of system resources using Continuous Time Markov Chain (CTMC) modeling. The analysis of the effect of reconfiguration on the availability is considered for fine-tuning MTD decision. MASON [88] framework utilizes Intrusion Detection System (IDS) alerts and vulnerability score *CVSS* calculated on the basis of CIA values to identify critical services in the network. A *Page Rank* based threat scoring mechanism is utilized for combining static and dynamic information, and prioritizing network nodes for MTD countermeasure port hopping. It is noteworthy, that port-hopping for $40-50\%$ services can help reduce overall threats in the network by 97%. This research work, however, does not consider the usability impact induced by the MTD countermeasure.

*MORE* [66] framework shows reduction in reconnaissance attempts and exploits targeting software integrity violation on frequent rotation of Operating Systems (OS) of the network under attack. Experimental analysis shows that a rotation window of 60s makes *nmap fingerprinting* attempts ineffective. Temporal and spatial diversity have been used to introduce genetic algorithm based MTD by Crouse *et al.* [71]. The experiments on average vulnerability of different configurations show decaying vulnerability rates for evolved configurations selected from the chromosome pool of VM configurations.

*Attack Graph/ Attack Tree:* CVSS present only a piece of quantitative information about the vulnerabilities such as complexity of performing network attack, impact on confidentiality or integrity of the system if the attack is successful, *etc.* This information alone is not sufficient for taking MTD decisions. Attack representation methods (ARMs) such as *Attack Graph* [45] and *Attack Tree* [35] answer questions such as (a) What are the possible attack paths in the system (b) What attack paths can be taken by the attacker to reach a specific target node in the network. SDN based scalable MTD solution [51] makes use of attack graph-based approach to perform the security assessment of a large scale network. Based on the security state of the cloud network, MTD countermeasures are selected.

Bayesian attack graphs have been used by Miehling *et al.* [94] for defending the network against vulnerability exploitation attempts. The defender's problem is formulated as a Partially Observable Markov Decision Process (POMDP) and optimal defense policy for selecting countermeasures is identified as a solution for the POMDP.

The security analysis of a large-scale cloud network in real time is a challenging problem [169]. Attack Graphs help in identification of possible attack scenarios that can lead to exploitation of vulnerabilities in the cloud network. The attack graphs, however, suffer from scalability issues, beyond a few hundred nodes as we discussed in Section II.

*Risk Metrics:* Like any other system, MTD systems have an associated risk once an organization considers deploying whole or part of the MTD technique. According to the *National Institute of Standards and Technology* (NIST), there are several attacks, service disruptions, and errors caused by human or machines that may lead to breaking benefits and critical assets at the organization or national level. Risks assessment is a critical measure and has many ways to deploy and use. In this survey, we identify and highlight research work that has adopted and took into consideration risks associated with deploying MTD solution. Specifically, we emphasize on the research work that evaluates the cost of the adopted MTD solution, since system administrators need to take into account the cost of using the MTD solution.

Risk assessment in the MTD has a direct and strong relationship with the effectiveness of the deployed MTD technique. According to [162], the system administrator can determine how good the MTD solution is by examining the associated risk. Therefore, the authors [162] studied the effect of deploying each MTD technique alone (i.e., shuffle, diversity, and redundancy) by inspecting the Hierarchical Attack Representation Method (HARM). HARM is basically a *ARM* at the upper layer such as attack graph (AG), and another ARM in the lower layer such as attack tree (AT), where these to ARM have a one-to-one mapping between them. Moreover, the authors also studied the associated risk by computing the instance measure (IM) which uses vulnerability's base score, impact score, etc [46].

Feedback-driven multi-stage MTD has been proposed by Zhu and Başar [63] for dealing with multi-stage attacks like Stuxnet [170]. Author's quantify the damage or cost caused by an attacker at different stages of the network. The game between attacker-defender is modeled as a finite zero-sum matrix game with a bounded cost function, and a mixed-strategy Saddle Point Equilibrium (SPE). Players utilize cost-function learned online to update MTD strategies. The numerical results show that feedback mechanism allows network defense to respond to unexpected (exogenous) events and reduce unusual peak of risk imposed by different vulnerabilities.

In [125], the authors provide metrics for MTD evaluation and risk analysis. For risk metrics, they proposed statistical metrics to study the effect of how the attacker can quickly conduct and succeed in adversarial attacks. The authors assumed the system will always have a running task that can be measured. The validity of the metrics was studied by simulating the APT attack scenario, where they assumed the APT will always have some sort of overhead that can be measured and detected. Finally, the performance of the proposed metrics was measured also by examining the CPU utilization of the designed system.

Lei *et al.* [60] consider the game theoretical formulation of MTD systems; specifically, they model it as a Markov Game. The authors provided a theorem, subject to probabilistic constraints, to calculate the revenue for the defensive and offensive approaches in MTD systems. Their work depended on testing different defensive and offensive strategies and was tested using a networking setup that includes vulnerable services and a firewall component as well.

Another work considered the statistical approach to evaluate the likelihood of a successful attack is the work conducted by [76]. The authors proposed an approach to determine the minimum effort required a system to detect stealthy botnets. Moreover, the entropy was measured to determine how close an adversary is to the detection point, where high entropy indicates the attacker is far in distance from the detector. The evaluation of the proposed approach was conducted on a real ISP network obtained from [171]. The results of the proposed algorithm show that the detection strategy has a complexity of $O(N^3)$ although theoretical complexity analysis indicates the algorithm is $\sim O(N^6)$.

Chung *et al.* [87] provide a detailed and comprehensive evaluation for the optimal countermeasure selection over a set of vulnerable attack paths in the attack graph. By evaluating the CVSS score of vulnerabilities, the authors determined the countermeasure selection option, taking into account the ration of the Return of Investment (ROI). Countermeasure option that produces the smallest ROI is considered the optimal one. The system performance of *NICE* [87] is proven to be efficient in terms of network delay, CPU utilization, and the traffic load.

To study the effect of a number of intrusions on the system's threat, Chowdhary *et al.* [51] use a statistical approach also to evaluate how the number of intrusion in the system, with the number of vulnerabilities, affect the threat score. The threat scoring algorithm is similar to *Page Rank* algorithm. To evaluate the proposed work, two experiments were conducted. One experiment is to test the threat scoring engine on software vulnerabilities and IDS alerts. The second experiments study the effect of port hopping attack. The results show that as the number of services in the system increase, the service risk value remains unchanged between a specified interval. However, the first few services show an increase in the number of risk value. Finally, the port hopping attack showed a reduction in threat score.

*Policy Conflict Analysis:* The MTD countermeasures such as network address switching can dynamically and rapidly insert new type of traffic, or new flow rules (environment managed by SDN). Pisharody *et al.* [156], [157] show how different MTD countermeasures such as network address change, load-balancing, and intrusion detection can cause security policy violations. The research works discuss SDN-based MTD, but the policy conflicts can cause security violations [158], loops, and blackholes in the network as discussed by Khursid *et al.* [159]. The violations of network-wide in-variants and security policies must be analyzed before deployment of MTD countermeasure. This research challenge is an important quantitative metric, which has not been considered by a lot of research works. We discuss this as a possible research oppurtunity in Section VI.

*2) Usability Metrics:* This category as shown in the Figure 13, analyzes MTD research work from the aspects such as QoS (network bandwidth, delay), impact on existing mission metrics and the cost of deploying MTD defense.

*QoS Metrics:* MTD can induce some performance cost on the existing system resources. Jafarian *et al.* [54] identify the virtual IP (vIP) mutation, range allocation, and range distribution constraints in order to minimize the QoS impact which can be induced by vIP collisions, as well as, maintain optimal-level of unpredictability. Probabilistic performance analysis of MTD reconnaissance defenses has been conducted by Crouse *et al.* [152]. The research work analyzes quantifiable MTD metrics such as reconnaissance, deception performance, *attack success probability* vs *connection drop probability*, attacker's success probability under different conditions such as network-size, number of vulnerable computers.

Taylor *et al.* [125] use mission and attack metrics for analyzing the effectiveness of a network defense. The research work analyzes dynamic defenses such as *Active Re-positioning in Cyberspace for Synchronized Evasion* (ARCSYNE) and *Self-shielding Dynamic Network Architecture* (SDNA) using mission and adversary activity set. Mission *Success*, i.e., the rate at which mission tasks are completed, and Mission *Productivity*, i.e., how often are mission tasks successful are used as QoS measurement metrics are used for evaluations.

A statistical analysis of static *vs.* dynamic attacks against different MTD strategies– uniform, random, diversity-based, evolution-based, and optimal– has been conducted by Carter *et al.* [65]. Experimental results on performance *vs.* adaptability shows that diversity-based MTD is the optimal strategy against most attack scenarios. They also show that uncertainty about the adversary type– slow adversary or fast-evolving adversary– can adversely impact the effectiveness of an MTD.

El-Mir [67] models performance parameters such as availability, downtime, and downtime cost using a Continuous

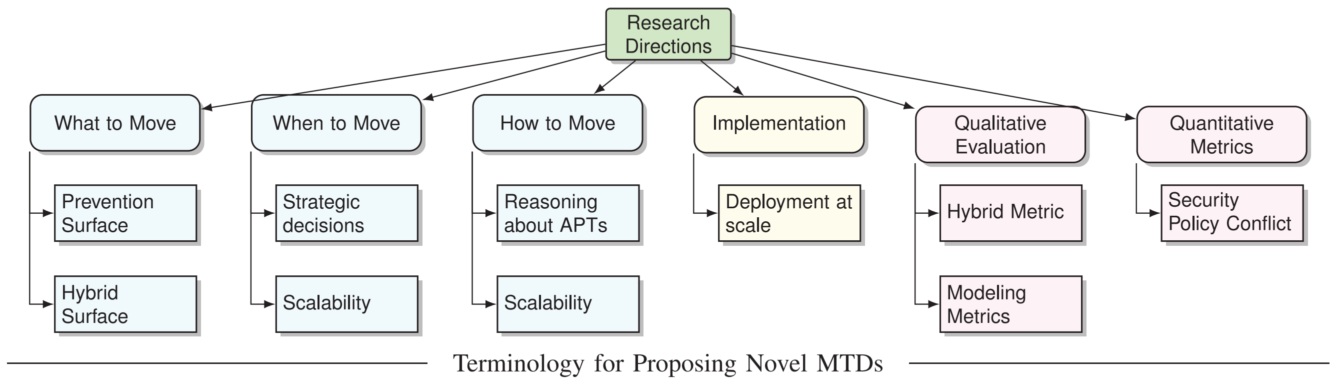


Fig. 14. Our categorizations help in identifying various directions that have been under explored during the development, implementation and evaluation of Moving Target Defenses (MTDs). In addition, it also provides a terminology that can help to underlying identify assumptions made by existing work and describe future MTDs.

Time Markov Chain (CTMC) model. The experimental results showed that cost-effective VM migration can be performed in an SDN-based network with limited impact on network performance. The research work utilizes normalized CVSS score as a key metric for initiating VM migration.

Sengupta *et al.* [77] analyze the performance impact of placing IDS (NIDS and HIDS) at all possible enforcement points in a cloud network. It is noteworthy that the placement of more than 15 detection agents in their simulated network fails to provide any additional intrusion detection benefit, whereas the network throughput decreases drastically from 16 Gbps in the case of a single detection agent to $\sim$6 Gbps when 15 detection agents are placed.

CHAOS [86] analyzes how the delay intentionally introduced by MTD impacts the packet count in a SDN-managed network. The SDN controller utilizes host-mutation and decoy-severs to deceive an adversary. The obfuscated network increases the cost and difficulty for an adversary targeting the network. The percentage of information disclosure reduces from 90% to 10% in a CHAOS protected network, with slight impact on packet delay (1s to 1.5s for 1800 packets).

*Cost Metrics:* Protection against Distributed Denial of Service (DDoS) attacks is one of the important priorities for many cyber systems. Wang *et al.* [153] present a cost-effective MTD solution against DDoS and *Covert Channel* attacks. Through MTD adaptation, their work [153] aims to answer two main questions: 1) what is the adaptation cost?, and 2) what is the cost incurred by a defender if an attacker succeeds in exploiting a particular vulnerability?. This solution does not rely on IDS-generated alerts while making the adaptation. The adaptation cost includes any cost related to purchasing required software or hardware that helps in the adaptation process. Lei *et al.* [154] utilize change-point analysis method for evaluation MTD cost-benefit for a multi-layer network resource graph. The proposed method analyzes mission productivity ($\Delta M$), and attack success productivity ($\Delta A$) on dynamic network address translation (DNAT). The evaluation results show reduced attack success probability using DNAT over a network under observation. The path enumeration mechanism used in this research work can, however, suffers from scalability challenges because of frequent path probability calculation and update operations. The cost and effectiveness evaluation of reactive and proactive network defense strategies, has been conducted by Taylor *et al.* [125] using Measurement of Effectiveness (MOE) metrics. The research work considers hop-delay for different attack success rates, and static defense policies. They show that an attacker's productivity, i.e., how quickly attacker can perform adversarial tasks increases against static defense, whereas attacker's confidentiality, i.e., ability to remain undetected is same for both the static and the dynamic defense case.

## VI. Research Opportunities

In this section, we highlight some lessons learnt and the under explored aspects of Moving Target Defenses (MTDs). These lead to a discussion on promising directions for future research (a brief summary is highlighted in Figure 14).

### A. The Configuration Set (What to Move?)

Works in MTD have concentrated mostly on the movement of the exploration, the detection and the attack surface. While moving the exploration and the attack surface helps a defender take away the advantage of reconnaissance that an attacker has, the goal of moving the detection surface is to improve the scalability and Quality of Service (QoS).

The movement of the prevention surface, which comprises of security modules such as Firewalls, IPS *etc.* has only been investigated by a couple of works (see discussion in Section III-A4). An MTD research can consider exploring Next-Generation Firewall (NGFW) architectures that combine security modules such as firewall, content filter, anti-virus *etc.* to provide a multi-layered defense-in-depth solution. Some current implementations of NGFW, that can be leveraged for testing the effectiveness of these defenses, are *Cisco ASA* [172] and *PAN Firewall* [173]. Further, with the rise of mobile technologies, MTDs can prove to be effective defenses. Although we discuss a few works on thwarting jamming attacks [59] and identity shifting in mobile ad-hoc networks [53], MTDs for different surfaces of in mobile infrastructure networks can be a promising direction for future research.

As previously stated, the movement of different surfaces in a single framework, although challenging, can provide greater security benefits than the movement of a single surface. Investigation in this area would require one to identify sets of configurations across the various surfaces that are compatible (in terms of performance) with one another. This prevents the number of strategies from multiplying uncontrollably by leveraging the expertise of system designers.

The categorization of the various software surfaces opens up the possibility of considering other logical surface level distinctions and thus, MTDs that shift these surfaces. For example, *Microsegmentation* [174] is a method of creating secure zones in data centers and cloud deployments to isolate workloads from one another and secure them individually. With MTD formalism, one can test the existing hypothesis and develop new ones for microsegmentation. We believe that formal modeling, in line with [79], one might discover that advanced services will be more effective when applied at a granular-level (as close to the application as possible in a distributed manner).

## B. The Timing Function (When to Move?)

The timing problem has mostly been ignored in previous works on MTDs. While some works perform empirical studies to test the best (constant) time-period for switching, they can be highly specific to threat model and the elements being shifted. For example, while 15 second time-periods are shown to be reasonable when protecting against jamming attacks [59], 60 seconds time periods are needed to defend against Network Mapping attacks [89] attacks. Also, the complexity of changing the virtual IP address vs. the underlying virtual machine impose different constraints on the lower bound on feasible time-periods.

Note that the handful of works that empirically determine time-periods are by no means complete. We need an extensive study just to come up with reasonable time periods for particular surfaces, how it effects that attack model and provide guidelines on efficient implementation methods.

On the other hand, a few existing approaches that do address the timing problem theoretically, suffer from scalability issues. In [60], the authors land up increasing number of states in their Markov Game by including time as a parameter. Inferring the defender's strategy in such Markov Games cripples the MTD to work beyond small networks. Effective solutions or improved modeling could both be interesting research directions for the future.

## C. The Movement Function (How to Move?)

Randomization in the movement is a necessary part of effective MTDs. Given the extensive use of randomization in cryptography, MTD defenses tend to use a Uniform Random Strategy (URS) an the movement strategy [61]. Several works have argued that often the defender has performance information about their system and known attacks that can be used to exploit their system. In such cases, a game-theoretic modeling can result in better strategies that offer higher gains in terms of both security and performance metrics. Unfortunately, the latter works suffer from scalability issues, encouraging practitioners to default to URS. Works that can leverage existing knowledge without suffering from scalability issues can fill an important gap in MTD research.

Recently, it has been shown that attacks on networks are better characterized by persistent approaches such as APTs. In such scenarios, attackers are known to demonstrate sequential attack behavior spread over a long time. A relatively new line of work investigates modeling MTDs to come up with strategies that are effective against APTs. Although these approaches leverage the use of attack graphs to bootstrap the modeling process [150], [168], they do not scale well. While real-world cloud services host thousands of nodes, researchers have only considered small-scale settings [78], [79]. Furthermore, works such as [94] and [95] that try to consider partial observability limit the scalability even further. Study on the design of approximation approaches and their effectiveness when compared to scalable baselines such as URS will be key for future research.

Current research often makes strong assumptions about the threat model. This makes results regarding the effects of such defenses questionable. In the future, we hope to see more studies that try to figure out realistic attack scenarios. Figuring out an attacker's behavioral model might also lead the modeling community to relax assumptions often made by rationality of an attacker.

## D. Qualitative and Quantitative Evaluation

As seen in Figure 12, none of the existing works demonstrate either empirically or model the impact of a proposed MTD on all four types of metrics. The lack of testing against real-world attackers also makes it difficult to prioritize which metrics a defender needs to care about and to what extent.

Beyond human studies to understand attack behavior, use of MTDs may introduce a new attack surface in cyber-systems. For example, firewall filtering rules need to be carefully analyzed to prevent conflicting security policies that may arise at the movement time because such conflicts might result in either dropping legitimate user packets or introducing new attack points. Although some works such as [51] have tried to address this issue of identifying security policy conflict for an SDN-managed cloud network, it is not immediately clear how it can be adapted in the context of other MTDs. A clear idea of when such scenarios arise and finding ways to address them would constitute an important line of research in the future.

A key idea is to have a continuous feedback cycle that verifies the security policy in place post MTD-countermeasure deployment. This can be done by ensuring end-to-end integration and regression test for the various use-cases pertaining to network traffic. Another solution could be to incorporate the policy conflicts that might arise into the modeling of the MTD. This would produce *safe* movement policies that for see the use of MTD as a new attack surface and avoid policy conflicts.

## E. Proposing Novel Moving Target Defenses

An important goal of this survey is to establish a common terminology for MTD researchers. Thus, we try to categorize an array of existing works in this terminology. For example, consider the work [79]. This MTD can be categorized as a moving target defense for the movement of detection surfaces with fixed interval switching formulated as a multi-stage game that performs simulation studies of simple use-cases and measures the security and performance of individual defenses in these settings.

An interesting idea would be to turn this goal of ours on its head and explore the design of new MTDs based on the permutations of the various categorization aspects designed in this survey. For example, a hybrid surface shifting MTD that (1) shifts the detection surface and then based on a stochastic environment, shifts the prevention surface, (2) models this problem as a two-step game, (3) considers rewards that incorporate performance of individual actions and security of the ensemble, and (4) showcases experimental results on an emulated testbed is a novel Moving Target Defense.

## VII. Conclusion

In this survey, we looked at various Moving Target Defenses (MTDs) that have been proposed for enhancing network security. First, we categorized them based on *what* surfaces these defenses move (the configuration set), *when* a move operation occurs (the timing function), and *how* they move between the different constituent system configurations (the movement function). In doing so, we highlight how the movement of particular software surfaces is linked to Advanced Persistent Threats; thereby, allowing us to understand how the various MTDs can help thwart real-world sophisticated attacks. The dearth of works that consider the simultaneous movement of different surfaces points to an exciting research direction and possibly, the invention of more effective MTDs. In answering *how to move*, we notice that many approaches leverage Artificial Intelligence (AI) methods in general and game-theoretic techniques in particular for crafting intelligent movement strategies.

Second, we discuss how these MTDs can be implemented in practice. We find that the use of centralized technologies like Software Defined Networking (SDN) helps in implementing the MTD countermeasures with limited impact on network performance. We showcase how the surveyed MTDs are implemented in the context of real-world systems ranging from simulation studies to use in commercial products. We highlight the key technologies leveraged by the various MTDs, the layers of the network protocol stack at which an MTD is effective and the level of maturity at which it is implemented. We briefly described a few test-beds that either have been leveraged by existing MTDs and encourage researchers to use them for evaluating the effectiveness of a proposed MTD. We conclude that (1) SDN/NFV is a dominant technology used by MTDs and (2) industrial adoption of MTD solutions is still limited to few application-security products.

Third, we discuss various metrics used for measuring the effectiveness of MTDs. We put forth two categorizations based on whether these metrics (1) consider security and performance impact of the designed system and (2) are used in the modeling phase to generate particular behavior *vs.* used for evaluation of the MTD. We notice that none of the proposed MTDs consider all the metrics we put forth. One wonders if a defense that models all the proposed metrics can be realized in practice and if so, prove to be better that existing defenses both in terms of performance and security.

Lastly, we highlight areas of network security where the scope of developing MTDs hasn't been investigated. We believe that coming up with defenses which can fill these gaps will improve security and/or performance aspects of various network systems. We conclude by showcasing how our categorization provides a common terminology for researchers to describe existing MTDs and develop future ones.

## Acknowledgment

A special thanks to Jim Kirby from NRL for the valuable feedback on the paper.

## References

[1] ESDS. (2018). *Gartner Identifies the Top 10 Trends Impacting Infrastructure and Operations for 2019*. Accessed: Mar. 3, 2020. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-12-04-gartner-identifies-the-top-10-trends-impacting-infras

[2] K. Panetta. (2016). *Gartner's Top 10 Security Predictions 2016*. [Online]. Available: https://www.gartner.com/smarterwithgartner/top-10-security-predictions-2016/

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[4] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[5] S. Roy, C. Ellis, S. G. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *Proc. IEEE 43rd Hawaii Int. Conf. Syst. Sci.*, 2010, pp. 1–10.

[6] H. Okhravi *et al.*, "Survey of cyber moving target techniques," Lexington Lincoln Lab., Massachusetts Inst. Technol., Cambridge, MA, USA, Rep., 2013.

[7] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein, "Finding focus in the blur of moving-target techniques," *IEEE Security Privacy*, vol. 12, no. 2, pp. 16–26, Mar./Apr. 2014.

[8] K. A. Farris and G. Cybenko, "Quantification of moving target cyber defenses," in *Proc. Sensors Command Control Commun, Intell. (C3I) Technol. Homeland Security Defense Law Enforcement XIV*, vol. 9456, 2015, p. 94560L.

[9] G.-L. Cai, B.-S. Wang, W. Hu, and T.-Z. Wang, "Moving target defense: State of the art and characteristics," *Front. Inf. Technol. Electron. Eng.*, vol. 17, no. 11, pp. 1122–1153, 2016.

[10] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Security Commun. Netw.*, vol. 2018, pp. 1–25, Jul. 2018.

[11] J. Kouns, "Open source vulnerability database (OSVDB)," in *The Open Source Business Resource*, Ottawa, ON, Canada: Talent First Network, 2008, p. 4.

[12] P. M. Mell and T. Grance, "Use of the common vulnerabilities and exposures (CVE) vulnerability naming scheme," Comput. Security Division, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Rep. 800-51, Sep. 2002.

[13] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.

[14] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *Proc. IFIP Int. Conf. Commun. Multimedia Security*, 2014, pp. 63–72.

[15] L. Martin. (2017). *Cyber Kill Chain (CKC)*. Accessed: Nov. 11, 2018. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[16] MI Center. (2013). *Apt1: Exposing One of China's Cyber Espionage Units*. [Online]. Available: http://mandian.com/

[17] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Comput. Security*, vol. 48, pp. 35–57, Feb. 2015.

[18] D. Moon, H. Im, J. Lee, and J. Park, "MLDS: Multi-layer defense system for preventing advanced persistent threats," *Symmetry*, vol. 6, no. 4, pp. 997–1010, 2014.

[19] R. Kissel, *Glossary of Key Information Security Terms*. Kidlington, U.K.: Diane, 2011.

[20] K. Glass and R. Colbaugh, "Web analytics for security informatics," in *Proc. IEEE Eur. Intell. Security Informat. Conf. (EISIC)*, 2011, pp. 214–219.

[21] P. Mehra, "A brief study and comparison of snort and bro open source network intrusion detection systems," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 1, no. 6, pp. 383–386, 2012.

[22] R. U. Rehman, *Intrusion Detection Systems With Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[23] P. Deshpande, S. Sharma, S. Peddoju, and S. Junaid, "HA HosIDS: T-based intrusion detection system for cloud computing environment," *Int. J. Syst. Assurance Eng. Manag.*, vol. 9, no. 3, pp. 567–576, 2018.

[24] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proc. 2nd ACM Conf. Comput. Commun. Security*, 1994, pp. 18–29.

[25] X. Zhang, C. Li, and W. Zheng, "Intrusion prevention system design," in *Proc. IEEE CIT*, 2004, pp. 386–390.

[26] H. Welte, "The NetFilter framework in Linux 2.4," in *Proc. Linux Kongress*, 2000.

[27] F. Pouget and M. Dacier, "Honeypot-based forensics," in *Proc. AusCERT Asia–Pac. Inf. Technol. Security Conf.*, 2004.

[28] M. Oosterhof, "Cowrie Honeypot," *Security Intell.*, 2014.

[29] SANS. (2013). *Intrusion Detection Evasion: How Attackers Get Past the Burglar Alarm*. Accessed: Nov. 11, 2018. [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-evasion-attackers-burglar-alarm-1284

[30] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013. [Online]. Available: arXiv:1312.6199.

[31] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. IEEE Security Privacy Workshops (SPW)*, 2018, pp. 76–82.

[32] N. Stojanovski, M. Gusev, D. Gligoroski, and S. J. Knapskog, "Bypassing data execution prevention on microsoftwindows XP SP2," in *Proc. IEEE 2nd Int. Conf. Availability Rel. Security (ARES)*, 2007, pp. 1222–1226.

[33] Z. Shu and G. Yan, "Ensuring deception consistency for FTP services hardened against advanced persistent threats," in *Proc. 5th ACM Workshop Moving Target Defense*, 2018, pp. 69–79.

[34] S. H. Houmb, V. N. Franqueira, and E. A. Engum, "Quantifying security risk level from CVSS estimates of frequency and impact," *J. Syst. Softw.*, vol. 83, no. 9, pp. 1622–1634, 2010.

[35] B. Schneier, "Attack trees," *Dr. Dobb's J.*, vol. 24, no. 12, pp. 21–29, 1999.

[36] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *Proc. ICISC*, vol. 3935, 2005, pp. 186–198.

[37] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proc. IEEE 22nd Annu. Comput. Security Appl. Conf. (ACSAC)*, 2006, pp. 121–130.

[38] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Proc. 42nd Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2012, pp. 1–12.

[39] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proc. 15th IEEE Comput. Security Found. Workshop*, 2002, pp. 49–63.

[40] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 336–345.

[41] J. Lee, H. Lee, and H. P. In, "Scalable attack graph for risk assessment," in *Proc. IEEE Int. Conf. Inf. Netw. (ICOIN)*, 2009, pp. 1–5.

[42] J. Homer, X. Ou, and M. A. McQueen, "From attack graphs to automated configuration management—An iterative approach," Kansas State Univ., Manhattan, KS, USA, Rep., 2008.

[43] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Proc. Eur. Symp. Res. Comput. Security*, 2008, pp. 18–34.

[44] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost SAT solving," in *Proc. ACM 27th Annu. Comput. Security Appl. Conf.*, 2011, pp. 31–40.

[45] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in *Proc. Int. Symp. Formal Methods Compon. Objects*, 2003, pp. 344–371.

[46] J. B. Hong and D. S. Kim, "Scalable security model generation and analysis using $k-$importance measures," in *Proc. Int. Conf. Security Privacy Commun. Syst.*, 2013, pp. 270–287.

[47] J. B. Hong, D. S. Kim, C.-J. Chung, and D. Huang, "A survey on the usability and practical applications of graphical security models," *Comput. Sci. Rev.*, vol. 26, pp. 1–16, Nov. 2017.

[48] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proc. 9th ACM Conf. Comput. Commun. Security*, 2002, pp. 217–224.

[49] X. Ou, S. Govindavajhala, and A. W. Appel, "MULVAL: A logic-based network security analyzer." in *Proc. USENIX Security Symp.*, 2005, p. 8.

[50] J. B. Hong and D. S. Kim, "Performance analysis of scalable attack representation models," in *Proc. IFIP Int. Inf. Security Conf.*, 2013, pp. 330–343.

[51] A. Chowdhary, S. Pisharody, and D. Huang, "SDN based scalable MTD solution in cloud network," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 27–36.

[52] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *Proc. Int. Conf. Security Privacy Commun. Syst.*, 2012, pp. 310–327.

[53] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun, "A moving target defense mechanism for MANETs based on identity virtualization," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2013, pp. 278–286.

[54] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," in *Proc. ACM 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 127–132.

[55] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using SDN-based moving target defense," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, 2016, pp. 627–630.

[56] Z. Zhao, F. Liu, and D. Gong, "An SDN-based fingerprint hopping method to prevent fingerprinting attacks," *Security Commun. Netw.*, vol. 2017, Feb. 2017, Art. no. 1560594.

[57] A. Schlenker *et al.*, "Deceiving cyber adversaries: A game theoretic approach," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 892–900.

[58] S. Jajodia, N. Park, E. Serra, and V. Subrahmanian, "Share: A Stackelberg honey-based adversarial reasoning engine," *ACM Trans. Internet Technol.*, vol. 18, no. 3, p. 30, 2018.

[59] R. Algin, H. O. Tan, and K. Akkaya, "Mitigating selective jamming attacks in smart meter data collection using moving target defense," in *Proc. 13th ACM Symp. QoS Security Wireless Mobile Netw.*, 2017, pp. 1–8.

[60] C. Lei, D.-H. Ma, and H.-Q. Zhang, "Optimal strategy selection for moving target defense based on Markov game," *IEEE Access*, vol. 5, pp. 156–169, 2017.

[61] R. Zhuang, S. A. DeLoach, and X. Ou, "Towards a theory of moving target defense," in *Proc. 1st ACM Workshop Moving Target Defense*, 2014, pp. 31–40.

[62] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," in *Moving Target Defense II*. New York, NY, USA: Springer, 2013, pp. 1–13.

[63] Q. Zhu and T. Başar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *Proc. Int. Conf. Decis. Game Theory Security*, 2013, pp. 246–263.

[64] S. Sengupta *et al.*, "A game theoretic approach to strategy generation for moving target defense in Web applications," in *Proc. 16th Conf. Auton. Agents MultiAgent Syst.*, 2017, pp. 178–186.

[65] K. M. Carter, J. F. Riordan, and H. Okhravi, "A game theoretic approach to strategy determination for dynamic platform defenses," in *Proc. 1st ACM Workshop Moving Target Defense*, 2014, pp. 21–30.

[66] M. Thompson, N. Evans, and V. Kisekka, "Multiple OS rotational environment an implemented moving target defense," in *Proc. 7th Int. Symp. Resilient Control Syst. (ISRCS)*, 2014, pp. 1–6.

[67] I. El Mir, A. Chowdhary, D. Huang, S. Pisharody, D. S. Kim, and A. Haqiq, "Software defined stochastic model for moving target defense," in *Proc. Int. Afro Eur. Conf. Ind. Adv.*, 2016, pp. 188–197.

[68] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev, "Frequency-minimal moving target defense using software-defined networking," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2016, pp. 1–6.

[69] A. Prakash and M. P. Wellman, "Empirical game-theoretic analysis for moving target defense," in *Proc. 2nd ACM Workshop Moving Target Defense*, 2015, pp. 57–65.

[70] S. Neti, A. Somayaji, and M. E. Locasto, "Software diversity: Security, entropy and game theory," in *Proc. 7th USENIX Conf. Hot Topics Security*, 2012, p. 5.

[71] M. Crouse, E. W. Fulp, and D. Canas, "Improving the diversity defense of genetic algorithm-based moving target approaches," in *Proc. Nat. Symp. Moving Target Res.*, 2012.

[72] B. Bohara, "Moving target defense using live migration of Docker containers," Ph.D. dissertation, Dept. Comput. Sci., Arizona State Univ., Tempe, AZ, USA, 2017.

[73] A. Chowdhary, S. Pisharody, A. Alshamrani, and D. Huang, "Dynamic game based security framework in SDN-enabled cloud networking environments," in *Proc. ACM Int. Workshop Security Softw. Defined Netw. Function Virtualization*, 2017, pp. 53–58.

[74] A. Clark, K. Sun, L. Bushnell, and R. Poovendran, "A game-theoretic approach to IP address randomization in decoy-based cyber defense," in *Proc. Int. Conf. Decis. Game Theory Security*, 2015, pp. 3–21.

[75] R. Colbaugh and K. Glass, "Predictability-oriented defense against adaptive adversaries," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, 2012, pp. 2721–2727.

[76] S. Venkatesan, M. Albanese, G. Cybenko, and S. Jajodia, "A moving target defense approach to disrupting stealthy BotNets," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 37–46.

[77] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "Moving target defense for the placement of intrusion detection systems in the cloud," in *Proc. Int. Conf. Decis. Game Theory Security*, 2018, pp. 326–345.

[78] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "General sum Markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks," in *Proc. Int. Conf. Decis. Game Theory Security*, 2019, pp. 492–512.

[79] A. Chowdhary, S. Sengupta, D. Huang, and S. Kambhampati, "Markov game modeling of moving target defense for strategic detection of threats in cloud networks," in *Proc. AAAI Workshop Artif. Intell. Cyber Security (AICS)*, 2019. [Online]. Available: https://arxiv.org/abs/1812.09660

[80] S. Sengupta, T. Chakraborti, and S. Kambhampati, "MTDeep: Boosting the security of deep neural nets against adversarial attacks with moving target defense," in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 479–491.

[81] A. Sinha, T. H. Nguyen, D. Kar, M. Brown, M. Tambe, and A. X. Jiang, "From physical security to cybersecurity," *J. Cybersecurity*, vol. 1, no. 1, pp. 19–35, 2015.

[82] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Proc. IEEE 4th Int. Conf. Cyber Security Cloud Comput. (CSCloud)*, 2017, pp. 97–103.

[83] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2014, pp. 485–492.

[84] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," in *Proc. Int. Conf. Inf. Sci. Security (ICISS)*, 2016, pp. 1–5.

[85] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FLIPIT: The game of 'stealthy takeover,'" *J. Cryptol.*, vol. 26, no. 4, pp. 655–713, 2013.

[86] Y. Shi *et al.*, "Chaos: An SDN-based moving target defense system," *Security Commun. Netw.*, vol. 2017, p. 11, Oct. 2017.

[87] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Trans. Depend. Secure Comput.*, vol. 10, no. 4, pp. 198–211, Jul./Aug. 2013.

[88] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "MTD analysis and evaluation framework in software defined network (MASON)," in *Proc. ACM Int. Workshop Security Softw. Defined Netw. Function Virtualization*, 2018, pp. 43–48.

[89] G. F. Lyon, *NMAP Network Scanning: The Official NMAP Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2009.

[90] V. N. Padmanabhan and D. R. Simon, "Secure traceroute to detect faulty or malicious routing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 77–82, 2003.

[91] S. G. Vadlamudi *et al.*, "Moving target defense for Web applications using Bayesian Stackelberg games," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2016, pp. 1377–1378.

[92] A. Chowdhary, S. Sengupta, A. Alshamrani, D. Huang, and A. Sabur, "Adaptive MTD security using Markov game modeling," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2019, pp. 577–581.

[93] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. van Dijk, "Markov modeling of moving target defense games," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 81–92.

[94] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on Bayesian attack graphs," in *Proc. 2nd ACM Workshop Moving Target Defense*, 2015, pp. 67–76.

[95] T. H. Nguyen, M. Wright, M. P. Wellman, and S. Singh, "Multistage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis," *Security Commun. Netw.*, vol. 2018, pp. 1–28, Dec. 2018.

[96] L. S. Shapley, "Stochastic games," *Proc. Nat. Acad. Sci. USA*, vol. 39, no. 10, pp. 1095–1100, 1953.

[97] M. J. Osborne *et al.*, *An Introduction to Game Theory*, vol. 3. New York, NY, USA: Oxford Univ. Press, 2004.

[98] S. Wang, H. Shi, Q. Hu, B. Lin, and X. Cheng, "Moving target defense for Internet of Things based on the zero-determinant theory," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 661–668, Jan. 2020.

[99] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Harlow, U.K.: Pearson Educ., 2016.

[100] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *Proc. AAAI*, vol. 4, 2004, pp. 709–715.

[101] S. Valizadeh and M. van Dijk, "Toward a theory of cyber attacks," 2019. [Online]. Available: arXiv:1901.01598.

[102] H. Li, W. Shen, and Z. Zheng, "Spatial–temporal moving target defense: A Markov Stackelberg game model," 2020. [Online]. Available: arXiv:2002.10390.

[103] J. Sherry, S. Ratnasamy, and J. S. At, "A survey of enterprise middlebox deployments," 2012.

[104] J. Steinberger *et al.*, "DDoS defense using MTD and SDN," in *Proc. NOMS*, 2018, pp. 1–9.

[105] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. ACM 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.

[106] Polyverse. (2018). *Moving Target Defense: Redefining the Power of Defense*. Accessed: Feb. 11, 2019. [Online]. Available: https://view.attach.io/ByfWW3KGf

[107] TrapX. (2018). *Deceptiongrid TRAPX*. Accessed: Feb. 11, 2019. [Online]. Available: http://trapx.com/wp-content/uploads/2018/03/

[108] CryptoTrap. (2018). *Cryptotrap TRAPX*. Accessed: Feb. 11, 2019. [Online]. Available: http://trapx.com/landing/wp-content/uploads/2018/12/Product_Brief_TrapX_CryptoTrap.pdf

[109] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online MTD," in *Proc. IEEE 36th Symp. Rel. Distrib. Syst. (SRDS)*, 2017, pp. 234–243.

[110] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDayLight: Towards a model-driven SDN controller architecture," in *Proc. IEEE 15th Int. Symp.*, 2014, pp. 1–6.

[111] Y. Wang, Q. Chen, J. Yi, and J. Guo, "U-TRI: Unlinkability through random identifier for SDN network," in *Proc. ACM Workshop Moving Target Defense*, 2017, pp. 3–15.

[112] OSRG. (2017). *RYU SDN Controller*. [Online]. Available: https://osrg.github.io/ryu/

[113] A. Homescu, T. Jackson, S. Crane, S. Brunthaler, P. Larsen, and M. Franz, "Large-scale automated software diversity—Program evolution redux," *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 2, pp. 158–171, Apr. 2017.

[114] C. Lattner and V. Adve, "The LLVM compiler framework and infrastructure tutorial," in *Proc. Int. Workshop Lang. Compilers Parallel Comput.*, 2004, pp. 15–16.

[115] X. Han, N. Kheir, and D. Balzarotti, "Evaluation of deception-based Web attacks detection." in *Proc. MTD@CCS*, 2017, pp. 65–73.

[116] S. Baxter and L. C. Vogt, "Content management system," U.S. Patent 6 356 903, Mar. 2002.

[117] W. Connell, D. A. Menascé, and M. Albanese, "Performance modeling of moving target defenses," in *Proc. Workshop Moving Target Defense*, 2017, pp. 53–63.

[118] S. Scherfke and O. Lönsdorf. *Simpy Testbed*. Accessed: Feb. 26, 2019. [Online]. Available: https://simpy.readthedocs.io/en/3.0/

[119] R. Ricci, E. Eide, and C. Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," in *Proc. Mag. USENIX SAGE*, vol. 39, 2014, pp. 36–38.

[120] A. C. Pappa, "Moving target defense for securing smart grid communications: Architectural design, implementation and evaluation," Ph.D. dissertation, Elect. Comput. Eng., Iowa State Univ., Ames, IA, USA, 2016.

[121] Morphisec. (2018). *Moving Target Defense*. Accessed: Feb. 12, 2019. [Online]. Available: https://www.morphisec.com/resources

[122] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, pp. 38–42, 2012.

[123] N. O. Ahmed and B. Bhargava, "MayFlies: A moving target defense framework for distributed systems," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 59–64.

[124] M. Berman *et al.*, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.

[125] J. Taylor, K. Zaffarano, B. Koller, C. Bancroft, and J. Syversen, "Automated effectiveness evaluation of moving target defenses: Metrics for missions and attacks," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 129–134.

[126] J. Yackoski, H. Bullen, X. Yu, and J. Li, "Applying self-shielding dynamics to the network architecture," in *Moving Target Defense II*. New York, NY, USA: Springer, 2013, pp. 97–115.

[127] J. Yackoski, J. Li, S. A. DeLoach, and X. Ou, "Mission-oriented moving target defense based on cryptographically strong network dynamics," in *Proc. ACM 8th Annu. Cyber Security Inf. Intell. Res. Workshop*, 2013, p. 57.

[128] J. Yackoski, P. Xie, H. Bullen, J. Li, and K. Sun, "A self-shielding dynamic network architecture," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2011, pp. 1381–1386.

[129] R. Chadha *et al.*, "CyberVAN: A cyber security virtual assured network testbed," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2016, pp. 1125–1130.

[130] K. Zaffarano, J. Taylor, and S. Hamilton, "A quantitative framework for moving target defense effectiveness evaluation," in *Proc. 2nd ACM Workshop Moving Target Defense*, 2015, pp. 3–10.

[131] Y.-B. Luo, B.-S. Wang, X.-F. Wang, X.-F. Hu, G.-L. Cai, and H. Sun, "RPAH: Random port and address hopping for thwarting internal and external adversaries," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 263–270.

[132] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for moving target defense network protection," in *Proc. IEEE 15th Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2014, pp. 1–6.

[133] Q. Jia, K. Sun, and A. Stavrou, "MOTAG: Moving target defense against Internet denial of service attacks," in *Proc. IEEE 22nd Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2013, pp. 1–9.

[134] B. Chun *et al.*, "PlanetLab: An overlay testbed for broad-coverage services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[135] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.

[136] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, "MT6D: A moving target IPV6 defense," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2011, pp. 1321–1326.

[137] S. Thomson, T. Narten, and T. Jinmei, "IPV6 stateless address autoconfiguration," IETF, RFC 4862, 2007.

[138] E. Al-Shaer, "Toward network configuration randomization for moving target defense," in *Moving Target Defense*. New York, NY, USA: Springer, 2011, pp. 153–159.

[139] J. H. Cox *et al.*, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25487–25526, 2017.

[140] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[141] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1086–1097, Sep. 2015.

[142] M. Roesch, "Snort: Lightweight intrusion detection for networks." in *Proc. 13th Syst. Admin. Conf. Lisa*, 1999, pp. 229–238.

[143] D. Fudenberg, D. Levine, and E. Maskin, "The Folk theorem with imperfect public information," in *A Long-Run Collaboration on Long-Run Games*. Singapore: World Sci., 2009, pp. 231–273.

[144] R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou, and A. Singhal, "Investigating the application of moving target defenses to network security," in *Proc. IEEE 6th Int. Symp. Resilient Control Syst. (ISRCS)*, 2013, pp. 162–169.

[145] A. Kosowski and V. Mosorov, "NESSi2 simulator for large-scale DDoS attack analysis," in *Proc. IEEE Perspective Technol. Methods MEMS Design*, 2011, pp. 157–159.

[146] A. G. Bardas, S. C. Sundaramurthy, X. Ou, and S. A. DeLoach, "MTD CBITs: Moving target defense for cloud-based it systems," in *Proc. Eur. Symp. Res. Comput. Security*, 2017, pp. 167–186.

[147] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM Demonstration*, vol. 14, no. 14, p. 527, 2008.

[148] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, "HeartBleed 101," *IEEE Security Privacy*, vol. 12, no. 4, pp. 63–67, Apr. 2014.

[149] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and practical network topology obfuscation," in *Proc. 27th USENIX Security Symp. (USENIX Security)*, 2018, pp. 693–709.

[150] S. Rass, S. König, and S. Schauer, "Defending against advanced persistent threats using game-theory," *PLoS ONE*, vol. 12, no. 1, 2017, Art. no. e0168675.

[151] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[152] M. Crouse, B. Prosser, and E. W. Fulp, "Probabilistic performance analysis of moving target and deception reconnaissance defenses," in *Proc. 2nd ACM Workshop Moving Target Defense*, 2015, pp. 21–29.

[153] H. Wang, F. Li, and S. Chen, "Towards cost-effective moving target defense against DDoS and covert channel attacks," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 15–25.

[154] C. Lei, D.-H. Ma, H.-Q. Zhang, and L.-M. Wang, "Moving target network defense effectiveness evaluation based on change-point detection," *Math. Problems Eng.*, vol. 2016, Jun. 2016, Art. no. 6391502.

[155] H. Alavizadeh, J. Jang-Jaccard, and D. S. Kim, "Evaluation for combination of shuffle and diversity on moving target defense strategy for cloud computing," in *Proc. 17th IEEE Int. Conf. Trust Security Privacy Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, 2018, pp. 573–578.

[156] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "BREW: A security policy analysis framework for distributed sdn-based cloud environments," *IEEE Trans. Depend. Secure Comput.*, vol. 16, no. 6, pp. 1011–1025, Dec. 2019.

[157] S. Pisharody, A. Chowdhary, and D. Huang, "Security policy checking in distributed SDN based clouds," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2016, pp. 19–27.

[158] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Commun. Mag.*, vol. 44, no. 3, pp. 134–141, Mar. 2006.

[159] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," presented at the 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2013, pp. 15–27.

[160] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Proc. IFIP Annu. Conf. Data Appl. Security Privacy*. 2008, pp. 283–296.

[161] A. Roy, D. S. Kim, and K. S. Trivedi, "Attack countermeasure trees (ACT): Towards unifying the constructs of attack and defense trees," *Security Commun. Netw.*, vol. 5, no. 8, pp. 929–943, 2012.

[162] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 2, pp. 163–177, Mar./Apr. 2016.

[163] P. Mell, K. Scarfone, and S. Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0.*, vol. 1, Forum Incident Response Security Teams, 2007.

[164] M. Christodorescu, M. Fredrikson, S. Jha, and J. Giffin, "End-to-end software diversification of internet services," in *Moving Target Defense*. New York, NY, USA: Springer, 2011, pp. 117–130.

[165] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *Proc. Nat. Symp. Moving Target Res.*, 2012, p. 246.

[166] J. Xu, P. Guo, M. Zhao, R. F. Erbacher, M. Zhu, and P. Liu, "Comparing different moving target defense techniques," in *Proc. 1st ACM Workshop Moving Target Defense*, 2014, pp. 97–107.

[167] J. B. Hong and D. S. Kim, "Scalable security models for assessing effectiveness of moving target defenses," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2014, pp. 515–526.

[168] S. Rass and Q. Zhu, "GADAPT: A sequential game-theoretic framework for designing defense-in-depth strategies against advanced persistent threats," in *Proc. Int. Conf. Decis. Game Theory Security*, 2016, pp. 314–326.

[169] A. Sabur, A. Chowdhary, D. Huang, M. Kang, A. Kim, and A. Velazquez, "S3: A DFW-based scalable security state analysis framework for large-scale data center networks," in *Proc. 22nd Int. Symp. Res. Attacks Intrusions Defenses (RAID)*, 2019, pp. 473–485.

[170] R. Langner, "StuxNet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May/Jun. 2011.

[171] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 133–145, 2002.

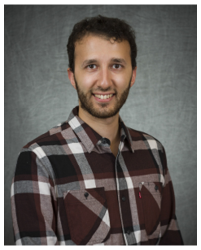
[172] J. Frahim, O. Santos, and A. Ossipov, *Cisco ASA: All-in-One Next-Generation Firewall, IPS, and VPN Services*. San Jose, CA, USA: Cisco, 2014.

[173] *Palo Alto Next Generation Firewall*, Palo Alto, Santa Clara, CA, USA, 2018.

[174] O. Mämmelä, J. Hiltunen, J. Suomalainen, K. Ahola, P. Mannersalo, and J. Vehkaperä, "Towards micro-segmentation in 5G network security," in *Proc. Eur. Conf. Netw. Commun. Workshop Netw. Manag. Qual. Service Security 5G Netw. (EuCNC)*, 2016.

**Sailik Sengupta** received the B.E. degree in computer science and engineering from Jadavpur University in 2013. He is currently pursuing the Ph.D. degree in computer science with Arizona State University. Then he worked for Amazon, where he became an Application-Security Certifier. He is interested in solving problems that arise in multiagent settings where the participating agents are non-cooperative or adversarial. He has looked at challenges in cyber-security, adversarial machine learning, and human-AI interaction. He was awarded the IBM Ph.D. Fellowship in 2018.

**Ankur Chowdhary** received the B.Tech. degree in information technology from GGSIPU in 2011, and the M.S. degree in computer science from ASU in 2015. He is currently pursuing the Ph.D. degree in computer science with Arizona State University, Tempe, AZ, USA. He has worked as Information Security Researcher for Blackberry Ltd., RSG, and an Application Developer for CSC Private Ltd. His research interests include SDN, Web security, network security, and application of AI and machine learning in the field of security.



**Abdulhakim Sabur** received the B.S. degree (Hons.) in computer science and engineering from King Saud University, Saudi Arabia, in 2015, and the master's degree in computer engineering from Arizona State University, Tempe, AZ, USA, in 2018, where he is currently pursuing the Ph.D. degree in computer engineering. He worked as a Researcher with the King Abdulaziz City for Science and Technology and a Teaching Assistant with Taibah University. His research interest include network and information security, vulnerability analysis and management, automated policy, and security checking in software-defined networking systems.

**Adel Alshamrani** received the B.S. degree in computer science from Umm Al-Qura University, Saudi Arabia, in 2007, the M.S. degree in computer science from La Trobe University Melbourne, Australia, in 2010, and the Ph.D. degree in computer science from Arizona State University in 2018. He is an Assistant Professor with the Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia, where he is the Chief Information Security Officer with the University of Jeddah. He has eight years of work experience in information security, network engineering, and teaching while working with the Faculty of Computing and Information Technology, King Abdul Aziz University, and the University of Jeddah. His research interests include information security, intrusion detection, and software-defined networking.

**Dijiang Huang** received the B.S. degree from the Beijing University of Posts and Telecommunications, China, and the M.S. and Ph.D. degrees from the University of Missouri, Kansas, in 1995, 2001, and 2004, respectively. He is an Associate Professor with the School of Computing Informatics and Decision System Engineering, Arizona State University. He has also served as the chair at multiple international conferences and workshops. His research was supported by the NSF, ONR, ARO, NATO, and Consortium of Embedded System. His research interests include computer networking, security, and privacy. He was the recipient of the ONR Young Investigator Program Award. He is an Associate Editor of the *Journal of Network and System Management* and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.

**Subbarao (Rao) Kambhampati** received the B.Tech. degree in electrical engineering (electronics) from the Indian Institute of Technology Madras, Chennai, in 1983, and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, in 1985 and 1989, respectively. He is a Professor of computer science with Arizona State University. He studies fundamental problems in planning, decision making, and game theory. He was the Program Chair for IJCAI 2016, ICAPS 2013, AAAI 2005, and AIPS 2000, and served on the Board of Directors of Partnership on AI. His research, as well as his views on the progress and societal impacts of AI, have been featured in multiple national and international media outlets. He received multiple teaching awards, including a University Last Lecture Recognition. He is the Past President of AAAI and was a Trustee of IJCAI. He is a fellow of AAAI, AAAS, and ACM, and was an NSF Young Investigator.