# A Survey of Moving Target Defenses for Network Security

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

***Abstract*—Network defense techniques based on traditional tools, techniques, and procedures (TTP) fail to account for the attacker's inherent advantage present due to static nature of network services and configurations. Moving Target Defense (MTD), on the other hand, provides an intelligent counter-measure by dynamically re-configuring the underlying systems, thereby reducing the effectiveness of cyber attacks. In this survey, we analyze the recent advancements made in the development of MTD tools, techniques and procedures (TTP) and highlight how these defenses can be made more effective with the use of artificial intelligence techniques for decision making. We first define a unified formal notation for MTDs that can capture different aspects of such defenses. We then categorize these defenses into different sub-classes depending on how they answer the three questions– what to move, when to move and how to move– showcasing how game theoretic strategies can effectively answer the latter question. To understand the usefulness of these defense methods, we study the implementation of such MTD techniques. We find that (relatively) new networking technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) provide effective means for implementing these dynamic defense methods. To encourage researchers and industry experts in using such defenses, we highlight industry use-cases and discuss the practicality and maturity of these defenses. To aid readers who want to test or deploy MTD techniques, we highlight existing MTD test-beds. Our survey then performs both a qualitative and quantitative analysis to better understand the effectiveness of these MTDs in terms of security and performance. To that extent, we use well-defined metrics such as risk analysis, performance costs for qualitative evaluation and metrics based on Confidentiality, Integrity, Availability (CIA), attack representation, QoS impact, targeted threat models and defense cost for quantitative evaluation. Finally, we conclude by summarizing research opportunities that our survey elucidates.**

***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**

## I. Introduction

The network and cloud infrastructure have become both ubiquitous and more complex in the past few years. Gartner predicts that by the year 2020, 60-70% of the entire IT infrastructure which includes deployed applications, technologies and services will be cloud-based [1]. Other important trends that are expected in a cloud-based IT infrastructure include serverless computing solutions, increased storage demands, growing demand for cloud-based container solutions, increased internet speeds with sub-millisecond service requirements, more competitive Service Level Agreements, e.g., 99.9999% promised up-time on certain cloud computing platforms such as Amazon Web Service (AWS).

While the storage capacity, the networking efficiency, and the hardware have received due attention and evolved with business demands, the various aspects that govern the security of a 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?*, i.e., whether, monitoring network traffic for malicious patterns, patching known vulnerabilities (even if done regularly), and relying on the network and perimeter defense such as Firewalls, Intrusion Detection Systems, Anti-virus, and Anti-malware tools enough to detect and thwart attacks?

There are multiple inherent limitations of these traditional defense mechanisms. First, the attacker, with time on their side, can spend reconnaissance effort in modeling the cloud system 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 year 2020, 99% of the vulnerabilities exploited will be known for the security and IT professionals since a year ago [2]. A major reason for this is the time and complexity associated with routine maintenance and security patching of the cloud and networking infrastructure. A degradation in the Quality of Service (QoS) provided to customers deters the Cloud Service Providers (CSPs) from making changes to the existing configuration. Third, *zero-day* attacks developed using the information the attacker gathers during the reconnaissance phase render traditional defenses useless.

To address the shortcomings of existing defense methods, *Moving Target Defense* (MTD) has emerged as a solution that provides a *proactive defense* against adaptive adversaries. The goal of MTD is to constantly shift between multiple configurations in a cyber-system (such as changing the open network ports, network configuration, software versions, etc.) in order to increase the uncertainty for the attacker; in effect, diminishing the advantage of reconnaissance that an attacker inherently has against traditional defense mechanisms. Note that this is

*These two authors have equally contributed to this work.

The authors Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Dijiang Huang, and Subbarao Kambhampati are with the School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University, Tempe, AZ, USA (e-mail: ssengu15@asu.edu, achaud16@asu.edu, asabur@asu.edu, dijiang@asu.edu, rao@asu.edu). Adel Alshamrani is with University of Jeddah, Saudi Arabia (e-mail: asalshamrani@uj.edu.sa)

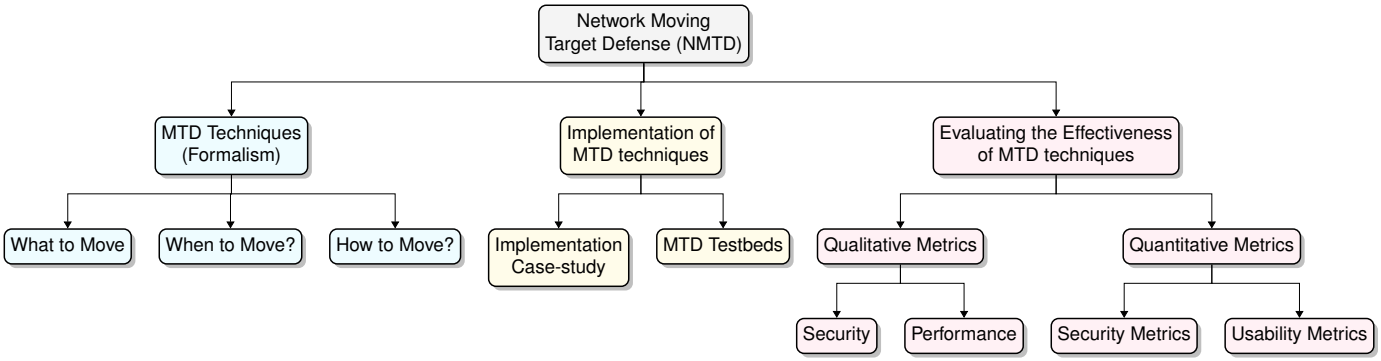



Fig. 1: We highlight a multitude of defense techniques based on the paradigm of Moving Target Defense (MTD) for cyber-systems and showcase new research opportunities that emerge from the categorization in our survey. We first discuss how all the MTDs surveyed can be viewed categorized based on the *what*, the *when*, and *how* that move. Then we showcase implementation examples and options that can be leveraged to deploy these defense techniques in practice. Finally, we provide a set of metrics that we believe are essential to properly evaluate the effectiveness of any MTD technique.

not true if the shifting mechanism is deterministic because the attacker, with time on their side, will eventually understand this movement and design their attacks accordingly, thereby rendering the moving target defense ineffective. Thus, MTD techniques should always have implicit randomness built into them. Note that this randomness increases the cost for an attacker to succeed in using a zero-day attacks because it does not necessarily know which configuration of the system is in place at any particular moment.

Advancements in networking technology provide a great opportunity for system administrators to implement MTDs in practice. *Network Functions Virtualization* (NFV) [3] has emerged as a technology to provide a virtualized implementation of hardware-based equipment such as firewall, routers, etc. *Virtual Network Functions* (VNFs) can be realized through *Virtual Machines* (VMs) or containers running on top of the physical server of cloud computing infrastructure. *Software-Defined Networking* (SDN) [4] acts as enabling technology for NFV. The SDN controller can work as a centralized security policy enforcer helping to design optimal movement policies. The programmable interfaces afforded by SDN can be leveraged to achieve a dynamic defensive strategy based MTD [5]; thereby providing a systematic solution by selecting countermeasures to prevent or mitigate attacks in an SDN enabled data center networking environment [6], [7]. Furthermore, the ease of implementing MTD mechanisms can help us evaluate the effectiveness of these defenses in practical settings, beyond theoretical analysis.

The key contribution of the survey are as follows: (1) provides an umbrella under which we can categorize the array of MTD techniques proposed for securing cloud environments, (2) introduces a new language that can be used to describe (and understand) new MTDs that makes clear what assumptions were made and what aspects were considered in developing the MTD, (3) gives an overview of how these ideas have been or can be implemented in practice, and how these methods can be evaluated from a qualitative and a quantitative standpoint, in effect, shedding light on how effective these tools and techniques really are, in terms of security and performance (see Figure 1).

The rest of the survey is organized in the following manner. In Sec. II, we introduce the reader to some background knowledge about the various stages of an attack in cloud systems, popular mechanisms for detection and defense of malicious traffic, and frameworks for modeling known attacks in cloud systems such as attack graphs, attack trees, and their inherent scalability limitations. In the next section, we describe a game formulation that captures the formalism proposed by all the MTD works we look at and use it to investigate and categorize the works depending on how they answer the questions (1) *what to move*, (2) *when to move* and (3) *how to move* them. Moreover, we 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 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/NFV in enabling them. We then showcase examples of existing MTD testbeds that can be used by researchers to evaluate their defenses and industry adoption of MTD solutions in production-grade networks. We categorize different implementation methods based on the kind of technologies they use, and their level of maturity. In Sec. V, we elaborate on various qualitative and quantitative metrics that have been discussed in literature so far, and whether they can help a cloud admin qualitatively decide if a defense mechanism is secure enough. In Section VI, we discuss the research directions that we believe are promising and need investigation. Finally, we conclude the survey in Section VII.

## II. Background and Related Work

Moving Target Defense (MTD) techniques for the network seek to move different parts of the network infrastructure that an attacker may leverage to launch an attack. Furthermore, many of the MTD mechanisms leverage some of the traditional defense methods but add dynamics to it to make it harder for an attacker to fool the overall defense. Thus, in this section, we first discuss related surveys in the field of MTD that

| Research Work | Cyber Surface Shift Analysis | | | Relation to APTs | MTD Implementation | MTD Evaluation | | Research Directions |
|---|---|---|---|---|---|---|---|---|
| | What? | When? | How? | | | Qualitative | Quantitative | |
| [8] | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| [9] | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [10] | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| [11] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| [12] | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |
| [13] | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Our Survey | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE I: A comparison of our survey with existing surveys on Moving Target Defenses (MTD). Note that our categorization looks at all the super-set of aspects that others works consider. Furthermore, the analysis of MTDs in the presence of Advanced Persistent Threats and a case study of how exisitng MTDs have been implemented are missing in all the previous works.

try to provide a categorization for such defenses and then, motivate the need for this survey. Second, we elaborate an attack process that will help us later understand which step(s) of the 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 an attack. These serve as building blocks of MTDs, which we discuss in the latter sections. Finally, we provide an overview of existing databases (NVD, OVSDB, CVSS) [14], [15] that are used to obtain domain knowledge about known vulnerabilities along with their impacts, and popular attack representation methods like attack graphs and attack trees.

### A. Related Works and Need for this Survey

We present a comparison of our survey to existing surveys in Table I. Firstly, we observed that most existing surveys [9], [12] 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 [16], [17]. 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 in-depth analysis of MTD and its adoption using advanced networking technologies like SDN and NFV. Moreover, we categorize the MTDs based on where they lie in the spectrum of implementation– ranging from simuation to research testbed implementations to production grade industry 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 were was evaluated (analytic/simulation/emulation/commercial). To the best of our knowledge, this categorization has not been considered by previous surveys.

Similar to our survey, many existing works [11], [12] have taken a look at categorizing the evaluation metrics for understanding the effectiveness of MTDs but unfortunately, 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 individual system configuration and with the ensemble, enabling us to highlight directions that can help improve existing MTDs.

Beyond just providing a categorization of existing work that most other surveys seek to do, the goal of our survey is also to establish a language for researchers to present their new MTDs. This will help in making evident the aspects that have been considered and those that have been assumed away in the development of the particular MTD. Furthermore, our categorization helps to identify promising directions for future research such as *policy conflict* analysis post-MTD, prevention, and hybrid surface shifting, modeling opportunities against APTs, etc. (see Figure 15).

### B. Attack Modeling Techniques

Knowledge relating to the phases of an attack executed by an adversary in order to achieve a desired objective is quite critical for the deployment of effective and intelligent cyber-defense. Organizations utilize advanced infrastructure management tools and follow best practices such as software patching, hardening, log analysis for attack surface reduction. 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.

We believe that an intelligence-driven, threat focused approach for studying attacks from attacker's perspective is required for detection and mitigation of attacks, especially against sophisticated attacks that are categorized as *Advanced Persistent Threats* (APTs) [17], [16]. Lockheed Martin defines the *Cyber Kill Chain* (CKC) [18] for collection, correlation, and categorization of the data related to a cyber attack. The evidence-based knowledge derived from the study, can help us in understanding, and thus, deploying an appropriate defense mechanism against an adversary. Thus, we now describe the different phases of the CKC against an APT scenario and later, leverage this to see how MTD can be employed during different phases to CKC.

*1) Reconnaissance:* The attacker gathers information about the target in this phase. The attacker can perform passive monitoring using automated tools such as trace-route and nmap to perform network probes. These probes look like normal traffic but are actually meant to identify valuable information such as the Operating System (OS) or the service version of the target machine, the network topology, the routing information, etc.

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

*3) Delivery:* The transmission of infected payload takes place during this stage. For example, the attacker may leave 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. 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 vulnerability and the attacker gaining elevated privileges on the victim's resources by utilizing specially crafted payload corresponding to each known/unknown 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 ($C\&C$):* Once the installation is complete, the attacker contacts the $C\&C$ in order 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 that involves identifying similar target nodes that have already been exploited, and *lateral movement* that involves identifying new systems that can be exploited in the network.

### C. 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 the 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 *et. al.* [19], 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. One such data collection and analytic system is CACTUS [20] that provides automated analytic capabilities that can be leveraged by prediction frameworks.

| **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 |

TABLE II: Highlights how existing defense mechanisms are related to the different phases of the Cyber Kill Chain.

*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:* The proxy 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 the Section II-C target known attacks with easy to detect signature patterns. We discuss a special category of cyber attacks known as Advanced Persistent Threats (APTs), which are harder to detect using reactive security mechanism like IDS and Firewall in the next subsection.

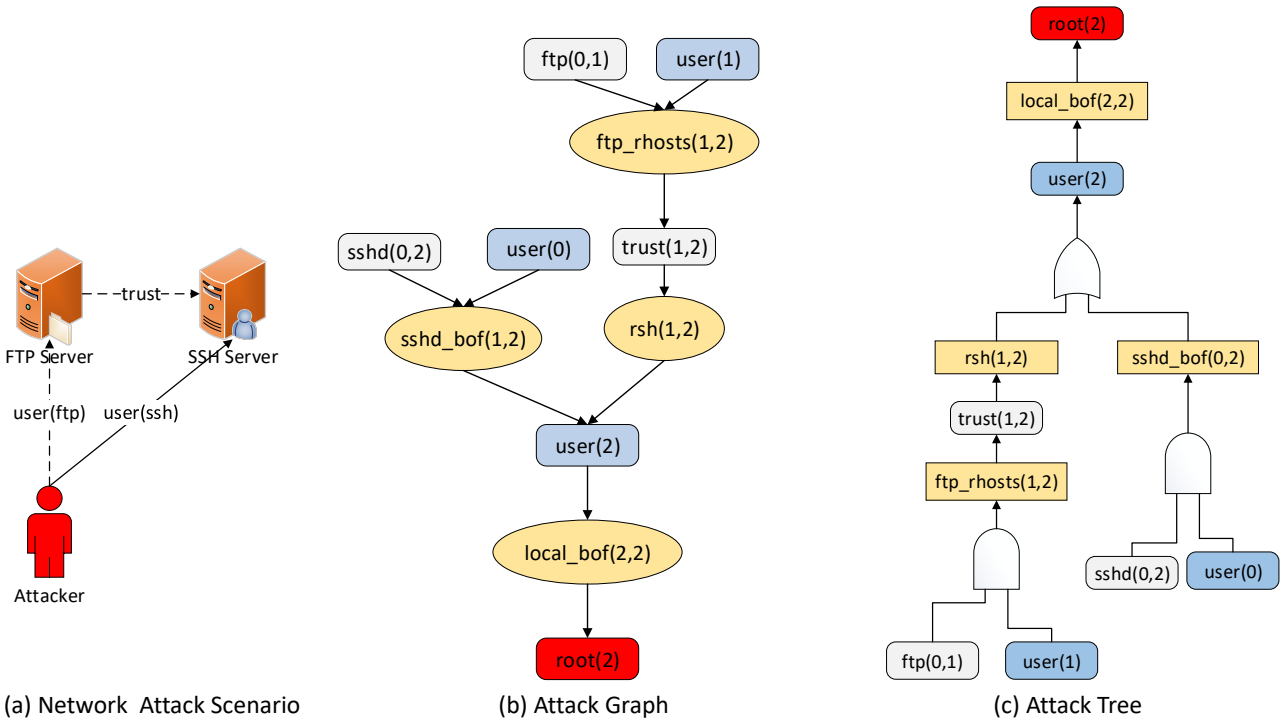
### D. Advanced Persistent Threats (APTs)

*Advanced Persistent Threats* (APTs) are different from normal cyber attacks. APTs are achieved by a group of advanced attackers, that is well funded either campaign against a high-value target organization [16]. An APT attacker i) pursues the objectives repeatedly over an extended period of time ii) adapts to defender's effort used for resisting against attacks iii) determined to maintain a level of access required to achieve his/her objectives. According to *Mandiant Report* [33], APTs such as *Operation Aurora, Shady Rat and Red October* [34], [35] have been used on a global scale for industrial espionage. The attackers mounting APTs often work closely with government organizations. The objectives are information ex-filtration or undermining the key services in the network using multiple attack vectors [36].

*1) Attack Representation Methods:* Cyber deception can serve as one major technique to defend against advanced and experienced adversaries. Shu *et al.* [37] proposed a cyber deception solution to protect *FTP* services against APT attackers. The research work is to ensure cyber deception consistency, in which the attacker must remain believing their attack is still successful. In other words, the defender may reroute the attacker connects to another host (or honeypot for instance) where the attacker is not able to notice a connection difference between the real IP address, or the Honeypot trap. The FTP services were hardened by observing the attacker's attempt to exploit vulnerabilities. Next, the attacker attempt is transferred to a *deceptive environment* [37]. *Bftpd* and *ProFTPd* services were set up in order to attract the APT attacker to the deceptive area. The author calls the process of transforming the attacker from the target to the deceptive area a context switch. It is vital to ensure the exact communication setup remains during the context switch process, otherwise, the attacker will know they are being trapped. This process is achieved by tracking all of the attacker traces before they enter the deceptive environment. These traces include process ID (PID), the IP address, the raw FTP command, argument of the command, the returned value after command execution, and the output of the command. The APT attacker's knowledge is recorded and represented in a logical tree structure [37]. The process of the context switch involves the following steps:

- Attacker's IP address and PID are obtained by the real FTP server and transferred to a defense agent running on the host machine.
- The defense agent dumps all the processes related to the PID to an image file.
- An IP table rule is added immediately on the host machine to prevent the actual FTP server from communicating with the attacker.
- An agent inside the VM handles the process from now on by communicating with the attacker, also, another IP table rule is added to prevent the VM from communicating with the attacker with its real IP address. Otherwise, the attacker will become suspicious of this activity.
- The VM changes it's IP address to the host machine one, which is also the one that the attacker is assuming they are connecting to.

Changing the IP address of the VM to the host one will result in different scenarios [37]. To handle this procedure, one possible option is to use the OS kernel to deliver packets coming to either, a local area network, or outside network wherein the first case the *ARP protocol* is used to deliver messages, and in the latter case the packets are delivered according to the routing table or to the default gateway. The proposed approach was evaluated by setting up two experiments that involve security experienced university students. The participants were asked to attack the FTP server, and then determine if they were able to figure out whether they were deceived or not. Also, the latency and overhead of migrating the connection to the




(a) Network Attack Scenario (b) Attack Graph (c) Attack Tree

Fig. 2: Attack representations such as an attack graph and an attack tree for a simple network attack scenario 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).

VM are measured and evaluated, where it showed an increase in the execution time due to creating many file systems and directories inside the VM.

### *E. APT and Cyber Kill Chain*

Phases of APT can be mapped to different phased of Cyber Kill Chain discussed in Section II-B. According to Alshamrani *et. al.* [16], different phases of APT can be defined as a) Reconnaissance b) Foothold Establishment c) Lateral Movement d) Data Ex-filtration e) Post Ex-filtration.

The *reconnaissance phase* has been described in Section II-B. The *foothold establishment* phase comprises of *Weaponization* and *Delivery* phases from Cyber Kill Chain. The attackers use the reconnaissance information in order to prepare an attack plan. The information collected can be used to exploit the vulnerabilities found in the target organization's web application, databases, and other software.

Once the attacker has gained a foothold into the target environment, he/she 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 the data ex-filtration phase of APT, the attacker tries to ex-filtrate the data collected 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 ex-filtration often goes 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 organization/government are achieved.

### *F. Domain Information for Modeling Cyber-attacks*

Defenders almost always have information about the system they want to protect. This can help the model the system and thereby the problem better for increasing 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 now discuss a few popular models and data sources that are leveraged by researchers, as we will later see.

*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 on a regular basis to 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 [46]. For (most) CVEs listed in the NVD database, we have a six-dimensional *CVSS v2* vector [15], which can be decomposed into multiple

| Category | Details | Complexity |
|---|---|---|
| Automated Attack Analysis [38] | 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 [39] | 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 [40] | 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 [41] | 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 [42] | Scalable attack graph for risk assessment using divide and conquer approach | $O(r(n+c)^k)$, where r is small coefficient. |
| Attack Cost Modeling [43] | Attack Graph cost reduction and security problem solving framework Min. Cost SAT Solving. | NP-Hard problem, SAT solving methods employed. |
| Ranking Attack Graphs [44] | Asset Ranking algorithm for ranking attack graphs to identify attacks. *Page Rank* based algorithm | Similar to complexity of page rank algorithm. |
| Attack Cost Modeling [45] | Identifying critical portions of attack graph. Min. Cost SAT solving, Counter-example guided abstraction refinement (CEGAR) | NP-Hard problem, SAT solving methods used. |

TABLE III: Complexity of the various Attack Representation methods.

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 in order 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 which allows the attacker to obtain reverse shell (rsh) on the system. 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** [47] 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 in order 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 [48].

**Definition 1.** *An Attack Tree [48] 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 [49]. 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.*
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 scalability issue of attack representation methods (ARMs) is defined by the exponential rise in complexity and size of the graph as the number of hosts or number of known vulnerabilities grow in the network [50]. There are many research works, which explain and provide some solutions the

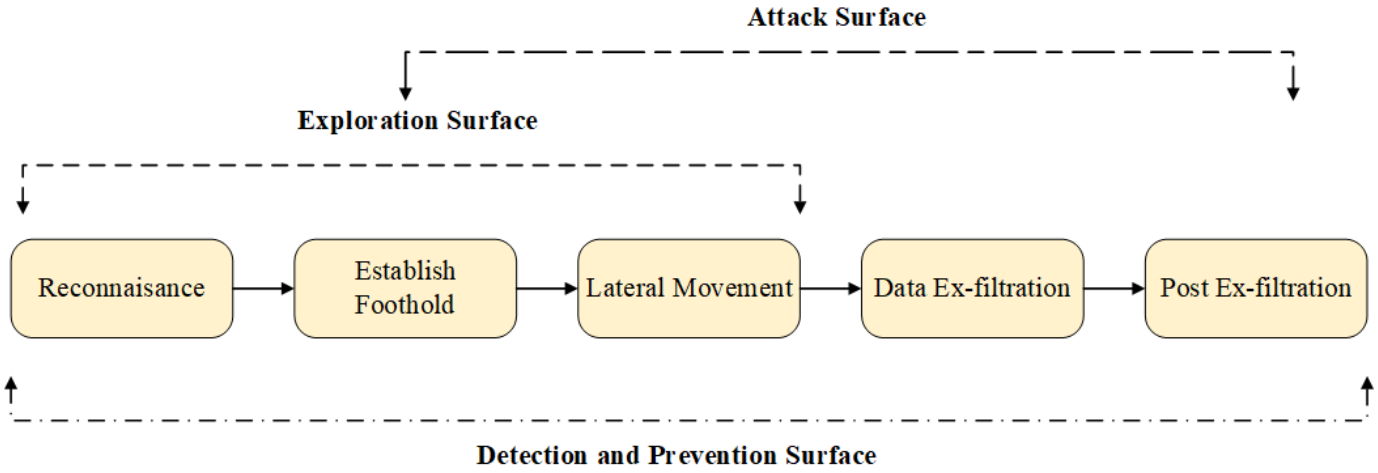



Fig. 3: A mapping between the different phases of Advanced Persistent Threat (APT) and various surfaces of a cyber system that various MTDs seek to move. Shifting of the exploration surface and the attack surface are effective only against some phases on an APT, whereas shifting the detection and the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

help in overcoming the scalability issues in creating ARMs. Amman *et al.* [51] presented a scalable solution in comparison with prior modules [52] by assuming monotonicity. This assumption allowed them to achieve scalability of $O(N^6)$ [53]. To mitigate the state explosion problem, most of the existing solutions either tried to reduce the dependency among vulnerabilities by using logical representation [54] or applying a hierarchical strategy to reduce the computing and analysis complexity of constructing and using ARMs by grouping and dividing the connectivity of the system into hierarchical architecture such as the work conducted by Hong *et. al.* [55]. The authors proposed a two-layer AG generation approach. The goal of this work was to provide a low complexity security analysis solution by considering network reachability and vulnerability information at different layers. The two-layered graphs they construct have vulnerability information of the individual hosts in the lower graph while the topological information of the network is in the upper layer. Authors expect this approach to learning dynamic changes in the environment. The research work, however, has no practical implementation to support the theoretical model.

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 [56], 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. Moving Target Defense Techniques

The goal of *Moving Target Defense* (MTD) is to move the components of a system in a continuous and randomized fashion. This increases the uncertainty for the attacker, making it harder (or costlier) for them to successfully exploit the system. For example, the information gathered by the attacker in the *reconnaissance phase* might become *stale* during the *attack phase* if the defender has moved to a new configuration in that time. Furthermore, we believe that MTD can be used as a meta-defense technique that can be used in conjunction with traditional defense methods to provide better security while ensuring the minimum impact on performance.

Let us now discuss the formalism of Markov Games [84]. This is a generalization of the formalism defined for MTDs in [65] and helps us capture the various aspects of the various MTDs surveyed. An MTD system can be specified 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 (or a deployed configuration), $A$ denotes the set of actions or pure strategies the defender ($A^{\mathcal{D}}$) or the attacker ($A^{\mathcal{A}}$) can take, $T : S \times A \rightarrow S$ represents a transition function over he joint space of actions (i.e. $A : A^{\mathcal{D}} \times A^{\mathcal{A}}$), and $R : T \rightarrow \mathbb{R}$ represents the reward/loss on performing a transition ($R^{\mathcal{D}}$) or an attack ($R^{\mathcal{A}}$). We will use the superscripts $\mathcal{D}$ and $\mathcal{A}$ to represent the functions for the defender and the attacker respectively.

Note that most of the elements in the tuple such as $S$ and $A^{\mathcal{D}}$ are highly problem-specific and can only be concretely defined in the context of a particular moving target defense. Although we do this later, we believe that an overview of how the different defenses model the various elements of the tuple help us to (1) get a holistic view of the various MTD mechanisms, giving us a sense of what kind of modeling makes a particular defense more effectively, and (2) highlight opportunities on how modeling of a particular element can be improved or combined with other MTDs.

Having defined the formalism that we will use throughout the survey, we now describe our plan to categorize of the various MTDs. We ask three questions that are relevant for any moving target defense– (1) *what to switch?*, i.e. the kind of surface that is being shifted by the MTD system, (2) *when to switch?*, i.e. finding an event after which a defender should shift, and (3) *how to switch?*, i.e. finding a mechanism that,

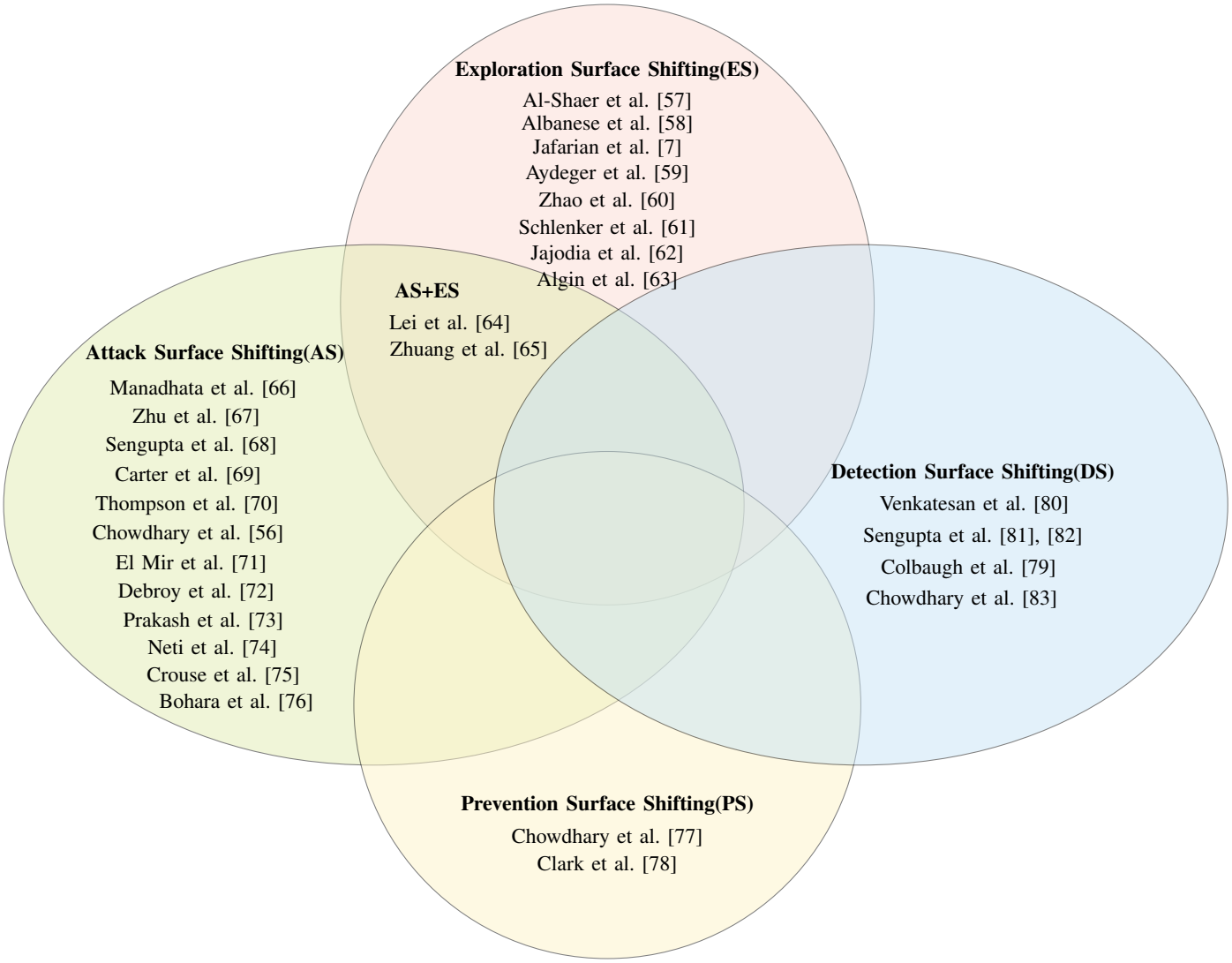



Fig. 4: The different surfaces that can be moved by a particular Moving Target Defense (MTD). Moving the exploration surface makes it harder for the attacker to figure out the exact system configuration by making the sensing actions noisy or untrue. Moving the attack surface makes a particular attack 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-flitrate data even when they have a strong foot-hold insider the system.

given the present state of the system, calculates a reasonable action to take. We then show how answers to these questions can help us in coming up with useful categorizations.

### *A. What to Switch? ≈ What actions to play?*

At a high level, every software system vulnerable to attacks can be defined to have four surfaces. These are:

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

These surfaces interact with one another in complex ways, especially in the context of Advanced Persistent Threats (APT) (see Fig. 3). A strong adversary first tries to explore the target network, trying to figure out its topology, bandwidth, software deployed on the different nodes, etc. The exploration surface accounts for reconnaissance, verifying if it has actually gained access and established a foothold and a new vantage point to further explore the deployed system. All this knowledge helps an adversary to execute attacks that can exploit the system, helping them to move to different points in the network, gather and ex-filtrate important information. As both exploration and attack traffic tend to be malicious in nature and thus, different from legitimate user traffic, the detection and prevention surface can come into play at any point in time.

In Fig. 4, we show a Venn diagram that categorized existing MTDs based on what surfaces they shift. Although there exists overlap among the various research areas (especially when viewed from the perspective of APTs), most MTDs shift one surface at a time, hardly discussing what happens to the other surfaces. After discussing the various MTDs, we highlight some unexplored areas that can lead to development of new

defense methods which shift multiple surfaces.

*1) **Exploration Surface Shifting**:* The main motivation behind shifting the exploration surface is to ensure that the information an attacker can gather by scanning for open-ports, sending non-malicious traffic to uncover system topology, discover vulnerabilities, etc. is noisy or inaccurate. Thus, an adversary, with this faulty information from the reconnaissance, will have to shoot arrows (attacks) in the blind.

In [57], [58], the authors propose the concept of Random Host Mutation (RHM)– moving target hosts are assigned random virtual IP addresses (vIP) in an unpredictable and distributed way. Formally, the action space $A^{\mathcal{D}}$ of configurations is the set of mappings from the set of vIP addresses in the Virtual Address Range (VAR) to the set of hosts in the network. Each action $\in A^{\mathcal{D}}$ represents a particular one-to-one mapping in a bipartite graph that consists a set of VARs and a set of hosts. In [7], the authors seek to implement the same defense method using a centralized approach that uses *software-defined networking* (SDN) technologies like *OpenFlow*.

Another line of work tries to reduce the quality of information an attacker can retrieve through exploration. In [59], 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 action space, i.e. the set of defender configurations, $A^{\mathcal{D}}$ is the set of all paths from a source point to a destination point in the network under attack. The MTD paradigm comes into play because the administrator chooses one path, in a randomized fashion, so that the attacker is not able to get a piece of reliable and deterministic path information between the two points in the network. They are thus forced to use a lot of attack traffic in order to retrieve a good topology of the underlying network, thereby increasing their chances of getting caught or having to deal with an inaccurate (hopefully, not useful) topology. On similar lines, there have been works that either try to move the fingerprint of a protected host [60] or randomly alter the time schedule that guides when a host transmits, reducing the effectiveness against selective jamming attacks against smart meters [63]. In [61], 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 about the underlying system opens up the possibility for an MTD solution for shifting the exploration surface. 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 adversaries model of the underlying network is incorrect as opposed to simply incomplete or noisy.

In [62], Jajodia et al. looks at how they can deploy *honeynets* in a strategic fashion. They show that deploying honeynets introduce 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 can represent the set of honeynets by $H$, then the action space $A^{\mathcal{D}}$ of configuration consists of all the subsets of honey-nets. Each state $a \in A^{\mathcal{D}}$, which is a subset of $H$ ($a \subset H$) that are currently deployed.

*2) **Attack Surface Shifting**:* Most of the work from the research community, both in cyber-security and in artificial intelligence, has focused on this area. The main aim of switching between attack surfaces is to render an attack action, that the attacker chooses after some exportation, *invalid* (for example, an attack to exploit a *Linux-based OS* will be useless if it is run on a machine running a *Windows OS*. We first discuss some MTD methods that are defined at an abstract level to be applicable to various parts of a network environment and then, focus on more specific ones.

In one of the earlier works [66], the authors have a set of systems, which forms their space of MTD configurations, and each configuration $a \in A^{\mathcal{D}}$ 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 untrusted items (data/system/network link) in the particular state $a$. The aim of the defender is to switch between the different states (or systems) so as to maximize the shifting of the attack surface and at the same time, have minimum impact on performance. This multi-objective trade-off is a common theme in most of the other works as well.

In [67], Zhu and Bashar break down a full-stack system into a number of 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 combination of technologies cannot be used to provide proper functionality to the end user. Thus, from all these possible combinations, they handpick a few technologies that meet the use-case demands for the full-stack software among which they switch. Thus, this represents the defender's action space $A^{\mathcal{D}}$ for their system. On similar lines, Sengupta et al. [68] also assumes a full-stack web-application and thus, similar action sets where the technologies relate to web-stack development. The two papers differ in the way they decide *how to switch*, which we shall discuss in detail later.

Carter et al. in [69], designs an MTD system that switches between different Operating System (OS) (they call these 'platforms'). Thus, in their case, the defender's action set $A^{\mathcal{D}}$ is a set of all OSs that they can move between and $a \in A^{\mathcal{D}}$ is a single operating system. They also mention a notion of similarity (or diversity) between two systems in $A^{\mathcal{D}}$ on the basis of code structure and logic. Authors in [70] move away from MTD techniques that are abstract and expected to work in general for all cyber systems. They implement an MTD that can perform the OS rotation for machines deployed in the underlying network using a centralized mechanism. We now shift our attention to MTDs that move elements solely relating to networks.

In [56], Chowdhary et al. describes an MTD that leverages port hoping to thwart exploits of known attack 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 discussion as to how attack surface shifting may happen after the detection surface comes into play. Along similar lines, authors in [71] move a deployed VM to a different physical server if the impact of known vulnerabilities (measured using particular metrics) on the physical server exceeds a threshold. In [72], the authors implement MTD at a different level where they move the 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 both services between the VMs and the VMs between the actual physical servers in the cloud network resulting in a multi-layer MTD, as discussed before [68], [67].

In [73], 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 [74], 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 [69]). They create a bipartite graph that maps hosts to vulnerabilities and show that MTD is more effective when the diversity is higher. This result is in lines with work in the security of Deep Neural Networks (DNN) where higher differential immunity to attacks leads to higher gains in robustness for ensembles based on MTD [85].

In rare cases, the number of playable actions in $A^{\mathcal{D}}$ may become huge. Due to computation and implementation challenges, a solution could be to first reduce $A^{\mathcal{D}}$ to $\hat{A}^{\mathcal{D}}$ such that $|\hat{A}^{\mathcal{D}}| \ll |A^{\mathcal{D}}|$ and then, implement an MTD with $A^{\mathcal{D}} = \hat{A}^{\mathcal{D}}$. In [75], researchers solve the first problem of finding a diverse $S'$ using a *genetic algorithm* search methods. On the other hand, authors in [76] showcase an MTD system that moves between containers and highlight that migration between containers, i.e., transition costs between the different states hardly results in a performance drop. Unfortunately, they are not able to generalize that such guarantees will hold for all MTD methods.

*3) **Detection Surface Shifting**:* The concept of detecting attacks from traffic on the wire or behavior or a request on a host machine has been a cornerstone of cybersecurity. Unfortunately, placing all possible Intrusion Detection Systems (IDS) on a system, especially in the case of large network leads to a degradation in the performance of the system. Thus, to minimize the impact of 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 in the underlying network. These methods are similar to patrolling methods that try to identify security violation in physical security domains like airports, wildlife sanctuaries, etc. [86].

In [80] and [81], authors show that when faced with stealthy botnets or 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 an ongoing attack. In both of these cases, they have a set of nodes $N$ that is deployed in the network. The action set of the defender is composed of all $k$-sized subsets of $N$, i.e., $A^{\mathcal{D}} = \{N_k : N_k \subset N \cap |N_k| = k(< |N|)\}$. In [83], the authors argue that in many real-world multi-layered networks. The author's assumption of an attacker being able to attack from any point is too strong. Thus, they use a Markov Game modeling and investigate the advantages of relaxing it and reasoning about the MTD mechanism over multiple stage attacks.

In contrast to the MTDs for detection surface shifting that are motivated by performance constraints of the underlying network, there is some work that investigates detection surface shifting for the sole purpose of enhancing security. In [79], 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 [87] coupled with the paradigm of MTD for stochastic ensembles [88], [85] can lead to interesting future research[1]. In these cases, each classifier is a possible action in $A^{\mathcal{D}}$.

*4) **Prevention Surface Shifting**:* The goal of an MTD that shifts the Prevention Surface is to make the attacker's process costly by introducing uncertainty in their mind about the defense mechanism in place. For example, it becomes difficult for an adversary to realize whether (1) their attack was undetected and thus, went through to the actual system, or (2) was detected and is presently being monitored (in a honeynet setting for example). Investigation on MTD techniques for shifting the prevention surface has been scarce, especially in the context of computer networks. We think this is mostly because (1) an administrator can only use these defenses when they are able to identify an attack with high accuracy and (2) identification of attacks itself is a strong assumption. In [77], the authors make this assumption and suggest an MTD mechanism that modifies the bandwidth of the network in response to malicious activity. Thus, the configuration space $A^{\mathcal{D}}$ consists of infinite actions as the bandwidth can take any real value. Similarly, researchers have also considered shifting the time window when replying to a query that may be malicious [78]. They also consider the deployment of decoy nodes and switching among them in order to hide from an adversary seeking to actively uncover the prevention surface, i.e. the decoy nodes.

*5) **Multi-Surface Shifting**:* MTD methods that move multiple surfaces of a underlying network can simply be done by choosing one technology from each of the above sections and combining them together. Yet, there is rarely any work in constructing such hybrid MTDs, i.e. ones that model the shifting of multiple surfaces at the same time. We discuss a few works that try to shift both the exploration surface and the attack surface of a system.

In [80], authors show that constructing a simple proxy-based switching approach is an ineffective defense by introducing a new attack known as the proxy harvesting attack, which explores the surface to collect a set of IPs and then doing

[1]In [89], authors highlight several challenges in the use of supervised machine learning in cloud network settings.

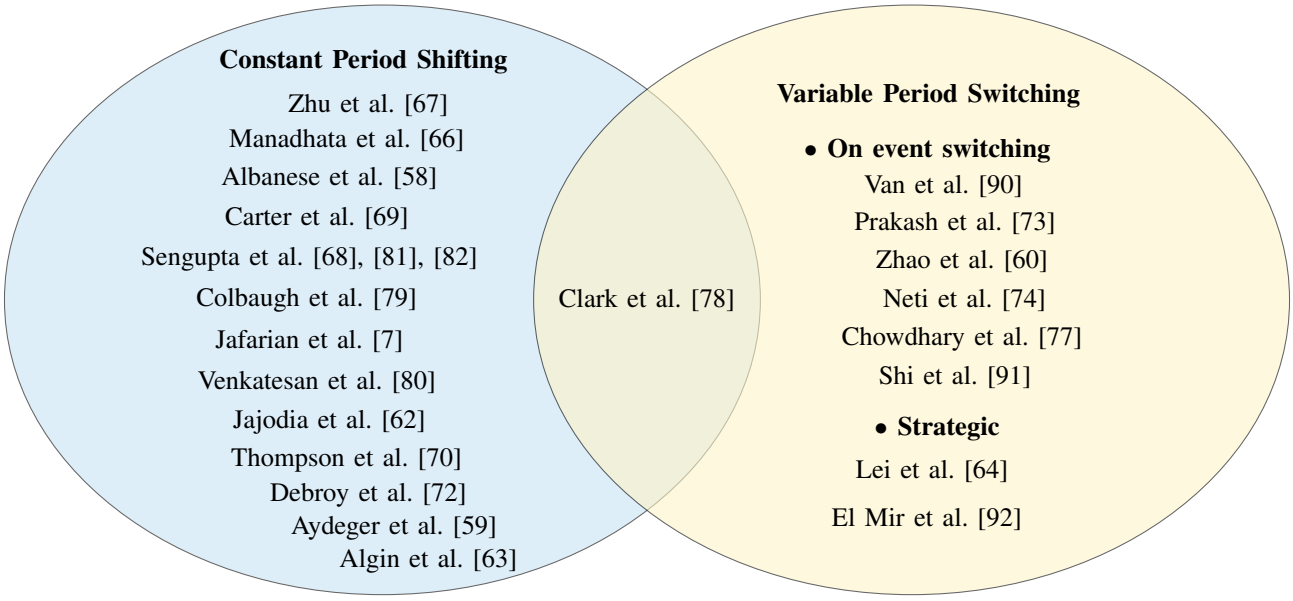



Fig. 5: The time period between two move events is either a fixed interval or decided based on some form of reasoning in the various Moving Target Defenses proposed in the literature.

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 attack-motivated setting, authors in [65] formalize the concept of MTD in which they highlight that shifting configuration can include both the attack and the exploration surface but do not go beyond syntactic notions to show how they can actually be done in the case of a real-world system. In this regard, authors in [64] try to strike a balance and formalize MTD as a *multi-step game* and show some ways in which the exploration surface $ES$ can be shifted or expanded and the attack surface $AS$ can be shifted. Their configuration space, thus, represents the joint space of both the surfaces, i.e. $S = ES \times AS$.

An interesting future direction could be to leverage existing security methods that add capabilities to multiple surfaces and see how MTD can be used in conjunction with them. For example, researchers have inspected the techniques involving detection of attacks and countermeasure selection in a single framework called NICE [93]. Can we leverage this to come with MTD solutions that move both the detection and prevention surface shifting remains an open question?

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 render themselves easily to management by a centralized controller like SDN, while others are better suited for movement in a decentralized fashion. As to how these can be combined together is probably not straightforward. We believe that overcoming these challenges will be an interesting research direction in the coming days.

### *B. When to switch? ≈ When to play?*

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 chooses to execute a switch action that changes their present configuration $s$ to another configuration $s'$*? The answer to this question is better given by dividing the works on MTD systems into two broad categories based on how the time window between multiple switch operations is modified- *Constant Period Switching* (CPS) and *Variable Period Switching* (VPS). We first describe the primary characteristics of these categories and based on it, categorize the different MTDs (see Figure 5).

In CPS, the MTD systems move from one state to another, mostly in a randomized fashion, always after a constant time period of $T$. This time is heavily dependent on what (surface) is being moved. The bulk of the work in MTD systems reside in this region. On the other hand, in VPS, the time period between one switch can be completely different from the time period of any other switch. We will later see that works in this area can be further sub-divided into three different categories. Although the two classes may seem to be the exact opposite by definition, it is not hard to see that when we talk about multiple surfaces being shifted, it is reasonable to have one surface shift using CPS while the other is shifted via VPS. We end the discussion on 'when to switch with a discussion of work in this region, which is sparse, opens up exciting avenues for future work in Section VI.

*1) **Constant Period Switching (CPS)**:* As mentioned, the key idea of these methods is to move the configuration of a system after a constant amount of time. In existing works, this is generally phrased in one of the following ways- (1) The actions of the defender are played after every time period $T$ (starting at zero) and at that instant, the system moves from one state to another based on the state transition probabilities, or (2) a new configuration is deployed by the defender a time

period $T$. However, independent of how it is phrased, they mean the same thing, i.e. to move the surface at time $T$.

A lot of the research works in MTD literature such as [94], [68], [81], [66], [62], do not explicitly bring up the topic of a time period but inherently assume that the system moves after time $T$. 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 think of both when implementing MTD solutions. Some works, such as [66], mention the challenge as non-trivial (that comes up during implementation) but do not provide a solution for it.

Some research works explicitly state that the time period is uniform and the equilibrium strategy is played at the start of each stage [67], [69], [63]. Note that, some of these works, such as [63], [58], try different time periods for switching and can only empirically test which works best. On similar lines, authors in [80] address DDoS attacks use the term $\frac{1}{T}$ to denote the frequency of switching while research in [7] call $T$ as the mutation interval. Notations like $t_i$ in [79] allow a user to use different time intervals for state $i$ of an MTD system but default to a constant time period in their work.

An interesting question is *how long should this time period $T$ be in practice*? In [70], the authors varied the time period $T$ from $60-300$ seconds and evaluated based on the likelihood of thwarting a successful exploit, the magnitude of impact for an exploit, and availability of applications to evaluate a good time for rotation operating systems in a network. They noticed that a smaller $T = 60s$ was often, good enough to thwart Network Mapping (nmap) [95] attacks. Although, OS fingerprinting was accurate in many situations even when $t = 60s$, an attacker could not successfully launch an exploit because of the small time window. For $T = 300s$, these results were weaker when stacked up against automated attack systems but showed promise when evaluated using manual attacks. In order to allow for fast rotations, the authors set up machines with different OS and after every interval, $T$ redirected traffic to a new OS. To reduce the load on having so many resources available and reduce redundancy, we believe that having at least two systems is necessary to allow for such small time windows, especially when shifting between OSs– one VMs sets up an environment while the other handles traffic and the role switch in the next time step.

In [72], researchers use the knowledge obtained from historical attack data to obtain a cyber attack inter-arrival (CAIA) rate. Then, they create an optimization problem to maximize the time period (denoted at $T_m$) of switching constrained upon the fact that $T_m$ is less than CAIA. On similar lines, authors in [59] have looked at *traceroute* [96] data between possible source-destination pairs in order to decide 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 [57] 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 mutation time. Each host has a mutation rate associated with it (denoted as $R_i$ in their paper) and some hosts belong to a set of High-Frequency Mutation (HFM) set while some cluster of hosts belongs to a Low-Frequency Mutation (LFM) set. To ensure that availability of a host is not impacted, the hosts that exhibit HFM 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 [58], the authors also have a local timer, i.e. maintained by each host, whose expiry triggers changes in their vIP. They use ad-hoc messaging to communicate this change to other nodes in the network to enable communication.

*2) **Variable Period Switching (VPS)**:* The idea behind this switching strategy, evident from its name, is to have a variable time period for switching from a configuration to another in an MTD environment. For example, if the system transitions through a series of states denoted as $\langle \dots, s, s' \dots \rangle$ and the time for which $s$ (or $s'$) was up is denoted as $t$ (or $t'$), 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 do not categorize the MTD works under the following sub-classes.

*a) On-event Switching:* Most works that have a variable time period for switching fall under this category. The main idea is that when a particular event occurs, such as detection of an attack, unavailability of a link or a server, the time period is altered and in most cases, a switch action is triggered immediately. A well-known case study is of the *FlipIt* games [90]. The state of the system is represented using a boolean variable that says who is in control of the software environment– $1/0$ indicate that the defender/attacker has control of the server. The defender gets to perform a move action that alters the configuration of the server only when it is in control of the server. Thus, the time period is dependent on how long an attacker may be in control of the server.

In [60], 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 time period of switching. On somewhat similar lines, authors in [74] have a belief indicating whether a vulnerability is compromised or not. Only beyond a certain threshold, the switch occurs. Other works argue that obtaining belief parameters such as *with what probability is the sender an attacker?* or *With what probability ss the system compromised?* are hard to obtain accurately in cybersecurity settings because they are bases on direct detection of cyber-events. In [77], 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 [91] scan open connections routinely and upon detection of unexpected connections, move between MTD configurations to protect the system against port-scanning attacks.

*b) Strategic:* In order to understand these methods, we first define $T_{\max}$ as the maximum time for which an MTD system can stay in the same configuration. We can now divide the time interval from $0$ to $T_{\max}$ into discrete time intervals

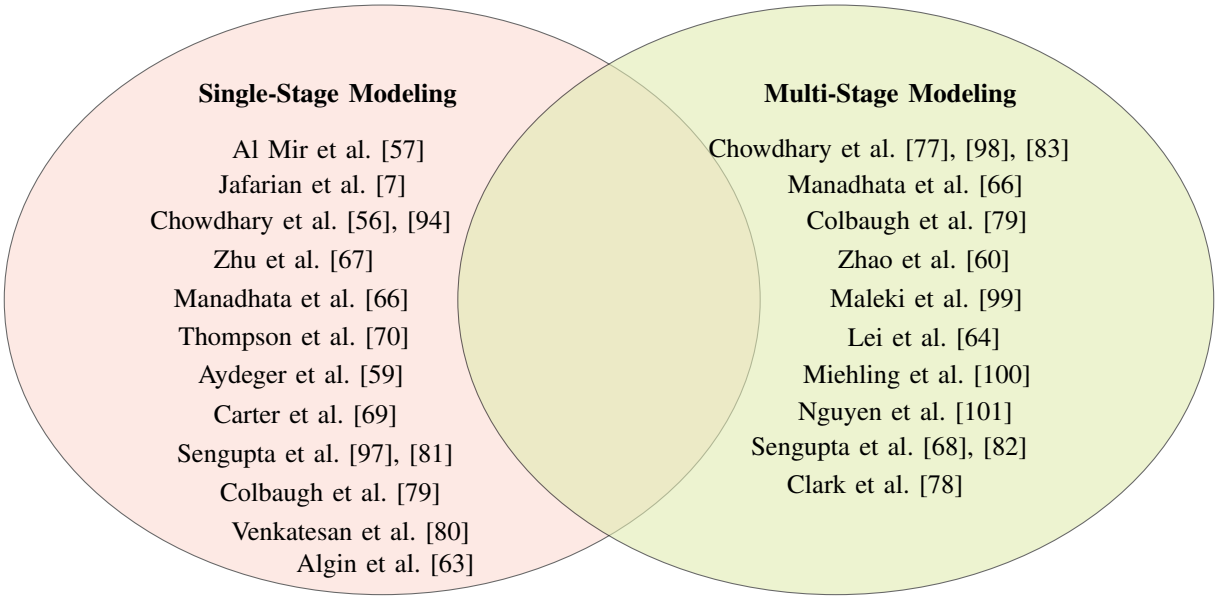



Fig. 6: Game-theoretic modeling for Moving Target Defenses (MTDs), that is necessary for obtaining optimal movement strategies, can be categorized into either single-stage games or multi-stage games. This choice often reflects the type of threat model being considered and the particular system for which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

and then decide how to pick a $T$, such that $0 \leq T \leq T_{max}$, at which a move occurs.

In [64], the authors incorporate the time period as a variable in their state $s$. Thus, $s' = (s, T)$ and the *goal* is to find an *optimal strategy* to play in these *new states*. Thus, if the *optimal policy* in state $(s, 2)$ dictates the admin to *not-switch*, the game moves to the state $(s, 3)$ thereby ensuring that the underlying cyber-surface does not shift. On the other hand, a *switch* policy transitions the state form $(s, 2)$ to $(s', 0)$ and the original cyber surface from $s$ to $s'$. Note that this idea can be extended to other MTD methods that use state information to decide on a movement strategy. On the other hand, authors in [92] resort to a simple strategy and do not present analysis as to how sub-optimal it is. They set $T_{\max} = \infty$ and based on the *impact* of (known) *vulnerabilities* in the deployed configuration and a threshold, select a $T \in \{0, \ldots, \infty\}$. If there are no vulnerabilities whose impact is greater than the predefined threshold, then the system does not move and remains static until a new vulnerability is discovered (either by the defender or an attack).

*c) Hybrid Switching:* Although the idea of having both a CPS and VPS in an MTD mechanism seems counter-intuitive at first, authors in [78] looks at MTD mechanisms where one layer that shifts the time window of replying to a strategic adversary is done in an *event-based* fashion while the other layer that deploys and moves among decoy nodes to hide from an active adversary is done using a constant time period. This opens up the possibility for works that shift more than one cyber-surface to research into when it is better to select CPS, VPS, or find a good combination of both.

### C. How to switch? ≈ How to Play?

In this section, we seek to look at how the policy for a defender who plans to move a particular cyber-surface is decided. As stated before, a few surveys have been solely devoted to analyzing how this question is answered. As all MTD mechanisms can be viewed using the game-theoretic formalism defined above, the survey on the use of game-theoretic methods for network security [8] has some overlap with our categorization. As we investigate a subclass of the defense mechanisms, i.e. ones that are based on MTD, we choose to classify our methods based on the following categorization (shown in Figure 6).

*1) **Single-Stage Modeling**:* In this section, we investigate how works have tried to infer the *defender's policy* over the action set $A^{\mathcal{D}}$ by modeling the current state of the MTD system and ignoring history or future considerations. In such cases, the policy obtained for the defender is optimal (or an equilibrium when formulated as a game) if the time horizon for the system is one.

In [66], the state of the system can be modified by a pair of actions, one by the defender and one by the attacker at each game stage. The defender's action, in this case, is a mapping $a^{\mathcal{D}} : F \rightarrow \{enabled,\ disabled,\ modified,\ unchanged\}$ where the set $F$ represents the set of system features. Thus, by applying these actions on a particular feature of the system, it can shift the attack surface. The attacker has the exact opposite action set $A^{\mathcal{A}}$ and hence, they can *re-enable* a port, *disable* a functionality etc. The rewards, $R^{\mathcal{A}}$ and $R^{\mathcal{D}}$ for the attacker and the defender, are given by weighing the change is system's features $\Delta F$, i.e. the change in the attack surface $\Delta AS$ and the attack surface measurement $\Delta ASM$, as follows.

$$R^{\mathcal{D}}(s, a^{\mathcal{D}}, a^{\mathcal{A}}) = B_1(\Delta F) + B_2(\Delta AS) - C_1(\Delta ASM)$$

$$R^{\mathcal{A}}(s, a^{\mathcal{D}}, a^{\mathcal{A}}) = B_3(\Delta ASM) - C_2(\Delta AS)$$

where the $B_i$-s and $C_j$-s are weights defined by security experts of the system. Note that the *cost of a defense* action

($C_1$) and the cost of attack action ($C_2$) 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 find a Subgame Perfect Equilibrium (SPE) in this game and the equilibrium strategy yields the defender's policy.

In [57] and [7], the defender assumes that more the randomness and switching, less the success of the attacker. Thus, they do not need to reason about multiple agents when coming up with action from $A^{\mathcal{D}}$ in state $s$. A state $s$ is defined by the present set of allocated vIPs to hosts. An action $a \in A^{\mathcal{D}}$ corresponds to selecting $k$ unused vIPs. They either pick $k$ random vIPs from the free vIPs or use the top-k scanned vIPs based on the assumption that vIPs that have been scanned more have less chance of being scanned again. On these unused $k$ vIPs are selected, they randomly allocate the $k$ present host to them by using a random permutation.

In [56], [94], the authors explain the game modeling based on a real-world example and compare the effectiveness of reactive based switching and Uniform Random Strategy (URS) respectively. URS, which is found to be more effective, means to pick any of the available action $a \in A^{\mathcal{D}}$ with probability —$1/A^{\mathcal{D}}|$. Other works that also use URS as the defender's policy are [70], [59], [63]. In [70], given the present deployed OS in state $s$, they consider the remaining OS configurations as defender's potential action in $s$ and play a URS over them. In [59], given all networking paths from source to destinations, one is picked at random while in [63], the authors use to Fisher-Yates shuffling to select a transmission schedule at the start of every round. When no information is available about the attacks and the defender has no preference over the MTD configurations, playing the URS seems a reasonable way to answer the question of *how to switch?* Fortunately, more information is often available to the administrator and in such cases, a URS may be sub-optimal.

To address this, in [69], [81], [97], [80], [67], and [102], the authors design a single-stage normal form game in which the defender's action set comprises of switching to any possible MTD configuration and the rewards are defined over the actions sets of both the defender and the attacker. 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, the defender rolls a dice at the end of each switching period and depending on the result, chooses the next configuration. Note that this is the optimal policy if the game were to be played for just a single time period (time horizon is one). In [102], the authors consider an MTD for IoT scenario and study the characteristics of a Zero-Determinant strategy that results in a dominant strategy for the single-stage game.

In [67], the authors assume (1) a zero-sum reward structure, and (2) an threat model in which the attacker is irrational. Thus, the game is played multiple times and in each step, the attacker's play is observed in order to improve the characterization of the attacker's behavior. In order to do so, the utilities of the game are updated after every observation to reflect the rationality of the attacker. After every update, the resulting Nash equilibrium (NE) is played. In [69], the authors assume that if the present system is working with an operating system $i$ and the system is made to switch to an operating system $j$, lower the similarity between $i$ and $j$, more secure is the move action from $i$ to $j$. The rewards for choosing action $j \in A^{\mathcal{D}}$ are defined accordingly and later fine-tuned by simulating the behavior of an adversary. Similar to [67], the NE strategy is chosen to be the defender's policy.

The other works assume that the attacker can observe (before any attack game-play actually beings) and realize the mixed strategy of the defender using maximum likelihood estimates. Thus, authors of [97], [81] concentrate on finding the Stackelberg equilibrium (SE) strategy for the defender. The rewards of this game, for both the $\mathcal{A}$ and $\mathcal{D}$, are obtained using the CVSS metrics of exploits that work in the context of the defender's actions, i.e. the MTD configurations that can be deployed. On similar lines, authors in [80] 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– paths that the system has taken, which may be a result of the actions taken by the players, in order to reach the present state, (2) the future– how the decision in the present state affects the rewards in the future, or (3) both.

In [77], similar to the work by [66] in single-stage modeling, the authors discretize the continuous action space of the defender, which represent 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 [79] 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 like [60], also update their belief about an attacker based on observations 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 [100], 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 probabilistic paths with defender's imperfect sensing capabilities to form a Partially Observable Markov decision Process (POMDP) [103]. 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* is (or chooses to) deviate away from the assumptions that inform the POMDP modeling. On the other hand, using a Partially Observable Stochastic Game (POSG) [104] seems unreasonable in terms of scalability for any real-world system.

Authors in [99], [83], [105], and [64] relax the assumption about partial observability and formalize MTD in a Markov Game framework. In [99], authors consider policies over

| 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% |

TABLE IV: Common causes of middlebox failures. Misconfiguration is a dominant cause of failure because of different underlay and overlay network. This motivates the need of unified configuration management framework like SDN.

the defender's action set that comprise of either single or multiple IP hops. Each action results in the defender uniformly selecting the next state give that an attacker samples actions randomly from $A^{\mathcal{A}}$. They are able to show that multi-element IP hopping actions increases the defender's reward by a greater magnitude compared to static defense strategies. As we will discuss later, the authors do not consider the defender's cost of performing a hop action or the downtime associated with it as apart of the defender's reward function. Thus, the optimal defender policy might not be optimal for real-world multi-layered network attack scenarios.

In contrast, the works [83] and [64] incorporates 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 given each move of the attacker at each step of the game. In [64], the inclusion of time variable in the state and the trick of including the attack strategies in the state to make solve it as an MDP makes it hard for the solution to seamlessly scale on a large network. 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 [68] model both the performance and security concerns in the different way that the above methods. The performance cost is the cost of picking a particular switch while the rewards of the state represent the security costs of picking 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 it is known that there may not exist a Markov stationary strategy that is the optimal switching strategy in such cases [106], the authors in [68] do not discuss how sub-optimal their switching strategies are.

Lastly, authors in [78] looks at an MTD defense mechanism for deception (obfuscation of the links in order 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 the Section VI.

## IV. Implementation of Moving Target Defenses

In this section, we discuss how the various Moving Target Defenses (MTDs) have been implemented using research testbeds or 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 have been evaluated in table V. We select a few of these works and do a case-study about them to highlight how movement strategies can be implemented in practice. Third, the sub-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 datapath from source to destination, e.g., Web Proxy, Firewall, IDS. This provides a decentralized framework that can be using alongside existing network technology to implement strategies for MTDs. Furthermore, researchers have focused significant effort on several issues associated with middleboxes, such as easier to use, easier to manage and deploy, 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.* [142] reveal factors such as increased operating costs caused by misconfiguration and overloads that affect their normal functioning. As shown in the Table IV based on results of survey [142] of 57 enterprise network administrators, from *NANOG* network operators group, the misconfigured and overloaded middleboxes are the major reasons of middlebox failure. 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 a assymetric advantage to the attackers as noted by Zhuang *et al.* [65]. 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

| Research Work | SDN/NFV | Implementation Layer | Technologies/ Testbed | Maturity Level |
|---|---|---|---|---|
| [83], 2019 | ✓ | network and application | Sample Use-Case | simulation |
| [107], 2018 | ✓ | network and application | ONOS [108] | emulation |
| [61], 2018 | | network | Personalized use cases | simulation |
| [62], 2018 | | abstract | Personalized use cases | simulation |
| [98], 2018 | ✓ | network and application | Personalized use-cases | simulation |
| [94], 2018 | ✓ | network | Personalized test-bed with VMs | emulation |
| [109], 2018 | | application | Polyverse | commercial |
| [37], 2018 | | network and application | Personalized Test-bed with VMs | simulation |
| [101], 2018 | | network | personalized use-cases | simulation |
| [110], 2018 | | network and application | Deception Grid [110], Crypto Trap [111] | commercial |
| [81], 2018 | | network and application | Mininet | both |
| [112], 2017 | ✓ | network | Opendaylight Helium [113] | simulation |
| [114], 2017 | ✓ | network | Ryu SDN [115] | simulation |
| [116], 2017 | | application | LLVM compiler [117] | emulation |
| [68], 2017 | | application | personalized use-cases | simulation |
| [118], 2017 | | application | CMS application [119] | simulation |
| [120], 2017 | | application | Simpy [121] | simulation |
| [77], 2017 | ✓ | network | network simulator + ODL controller | emulation |
| [64], 2017 | | abstract | Personalized Test-bed with VMs | simulation |
| [60], 2017 | ✓ | application | Mininet with POX controller | emulation |
| [91], 2017 | ✓ | network | CloudLab [122] | emulation |
| [123], 2016 | | network | Smart grid, energy management system (EMS) | emulation |
| [68], 2017 | | application | personalized test-bed | simulation |
| [124], 2017 | | network and application | Morphisec | commercial |
| [63], 2017 | | network | Personalized Simulation Environment | simulation |
| [59], 2016 | ✓ | network | Mininet with Floodlight controller | emulation |
| [99], 2016 | | network | Personalized use-cases | simulation |
| [56], 2016 | ✓ | abstract | OpenStack [125] with ODL [113] controller | emulation |
| [126], 2016 | | network | OpenStack [125] | emulation |
| [72], 2016 | ✓ | network | GENI [127] | emulation |
| [80], 2016 | | network | Rocketfuel Dataset | simulation |
| [128], 2016 | | network | ARCSYNE [129] & SDNA [130], [131] | emulation |
| [92], 2016 | ✓ | network and application | Mininet with POX controller | simulation |
| [132], 2016 | ✓ | network | CyberVAN | commercial |
| [72], 2016 | ✓ | application | GENI [127] | emulation |
| [73], 2015 | | abstract | Personalized simulation environment | simulation |
| [133], 2015 | | network and application | Cyber Quantification Framework | emulation |
| [78], 2015 | | network | matlab | simulation |
| [100], 2015 | | network | personalized use-cases | simulation |
| [134], 2015 | | network and application | RPAH framework | simulation |
| [70], 2014 | | network and application | CORE Impact Pro | commercial |
| [69], 2014 | | application | personalized test-bed | simulation |
| [135], 2014 | ✓ | network | Personalized Cloud System | emulation |
| [136], 2013 | | network and application | PlanetLab [137] | emulation |
| [93], 2013 | | network and application | Personalized virtual cloud system | emulation |
| [90], 2013 | | abstract | Personalized test-bed | simulation |
| [67], 2013 | | application | game-based theoretical framework | analytic |
| [7], 2012 | ✓ | network | NOX in Mininet | emulation |
| [74] , 2012 | | abstract | Proposes to use a modified CTF environment | analytic |
| [75] , 2012 | | network and application | Personalized test-bed | emulation |
| [79], 2012 | | network | KDD dataset [138] | simulation |
| [57], 2012 | ✓ | network | Personalized use cases | simulation |
| [139], 2011 | | network and application | SLAAC [140] | simulation |
| [141], 2011 | | network | MUTE | simulation |

TABLE V: A taxonomy of MTD Implementation– classifies based on use of SDN/NFV, the layer of network protocol stack they target, key technologies used, and the level of maturity at which these MTDs were implemented.

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 [143]. SDN and NFV have emerged as a state-of-the-art network architecture for data centers and backbone networks. Google's *B4 project* [144] shows the feasibility of SDN for handling real-world network challenges such as traffic engineering and Quality of Service (QoS).

The decoupling allows a 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 [145]. 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 [108], OpenDaylight [113] 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 to provide a virtualized implementation of hardware-based equipment such as firewall, routers, Intrusion Detection System (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 enabling technology for NFV. Despite the great benefits offered by SDN and NFV, the security, privacy, and trust management [146] has been addressed only by few research works (see Fig. 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. Most research works have been implemented at network, and pplication layers. We also identify the key technologies used by these research works in the column 4 of the table. If a certain research work utilizes and implementation testbed, e.g., GENI [127],

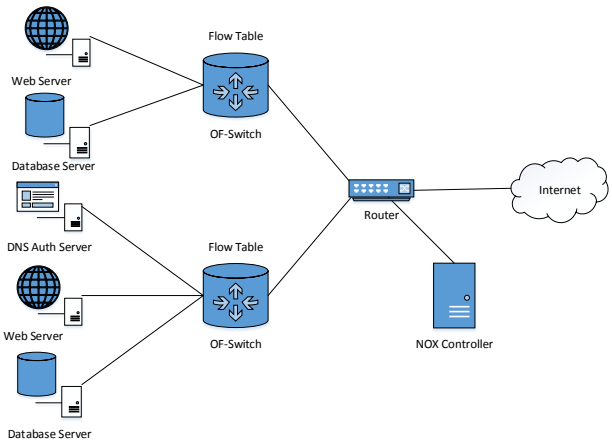



Fig. 7: OpenFlow-based Random Host Mutation. The solution mutates virtual IP (vIP) of hosts based on pool of available IP addresses (treatment of MTD as an adaptation problem).

the name of the testbed has been 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 [127]. 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 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 [135].

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.* [135]. 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 [112], route modification [107], IP, and port obfuscation as discussed by Wang *et al.* [91].

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.* [7] use OpenFlow based random host mutation technique to switch virtual IP (what to switch) targeted by reconnaisance attempts. Debroy *et al.* [72] use SDN framework to identify optimal rate of migration (when to switch) and ideal migration location for VMs under DoS attack. Chowdhary *et al.* [77] 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.* [7] proposed 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.

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.* [77]. 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 [147] detects malicious DDoS patterns, and uses Nash Flok Theorem [148] to analyze the behavior of a malicious node, e.g., red color VM shown in Figure 8 (a). The information is represented as an extensive-form game of 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 $t = \{0, 1, 2, ..\} \in T$ 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 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* [72] 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

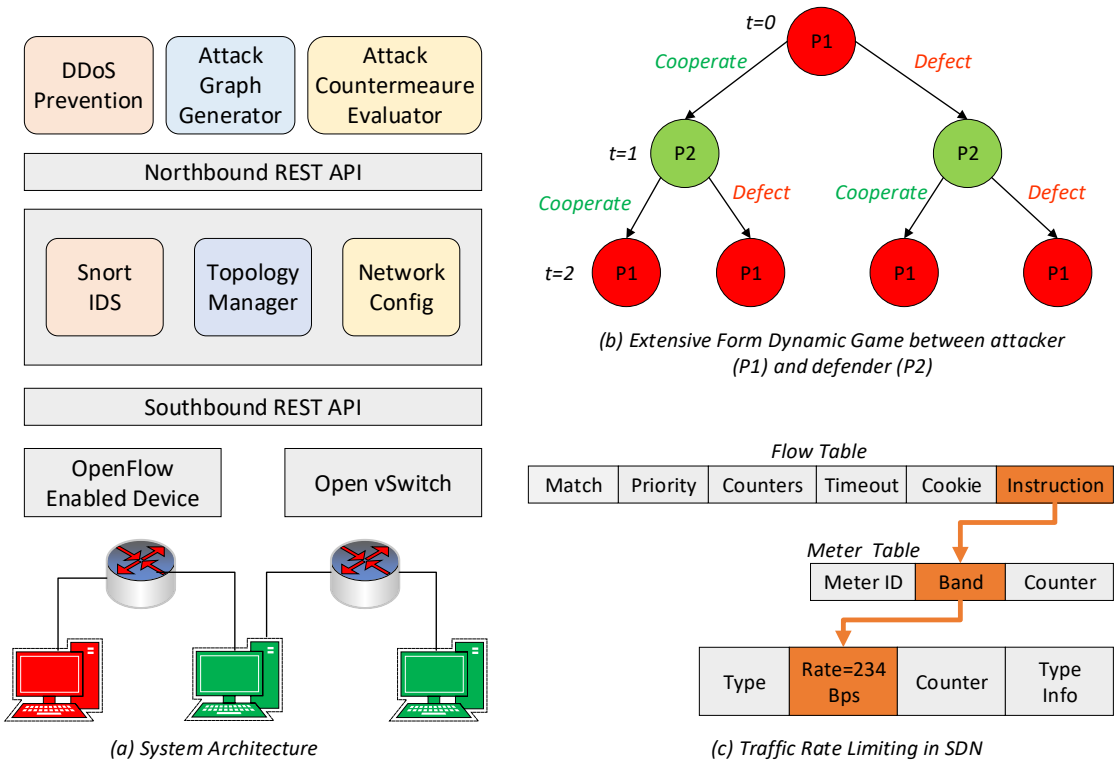



Fig. 8: SDN/NFV based MTD that models the interaction as a game between an attacker and the defender in order to select an optimal countermeasure strategy. The work considers bandwidth limitation, and treats the MTD as an adaptation problem.

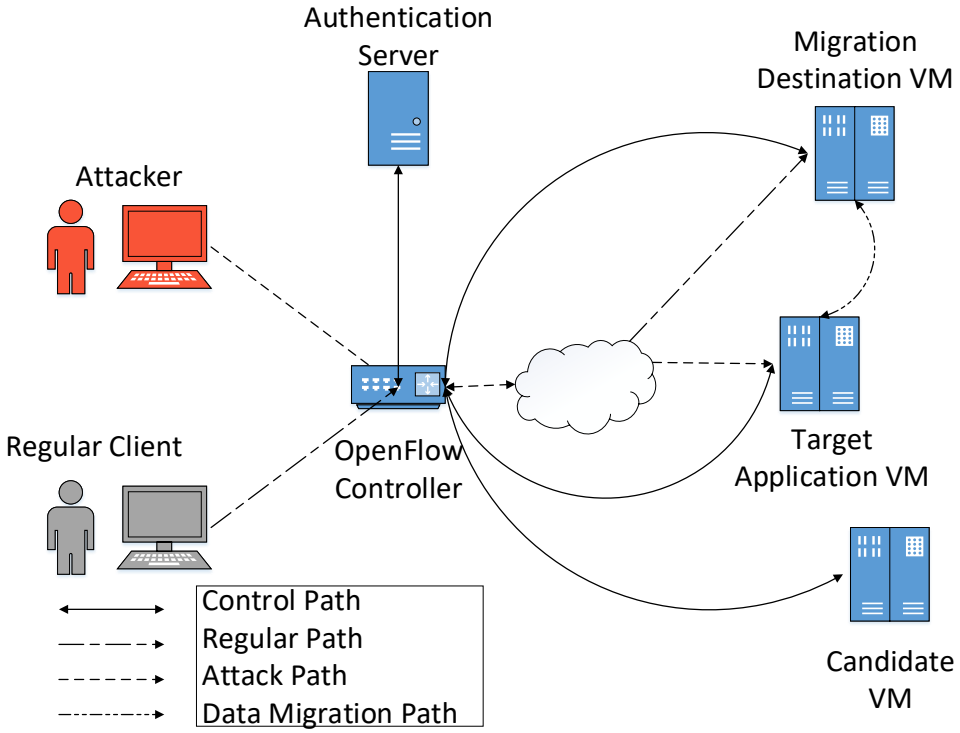



Fig. 9: Virtual Machine (VM) migration using frequency minimal MTD that is effective against DoSS attacks. The goal of this work was to find an optimal migration frequency, thereby answering the question *when to move*.

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) [127] 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.* [72] utilized 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 [125] 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 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** [126] 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.* [149] conducted 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* [150], 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) [151] 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:* [132] 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* [152] 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:* MTD system for IP hopping in a smart grid environment has been discussed by Pappa *et. al* [123] as shown in the 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 substa-

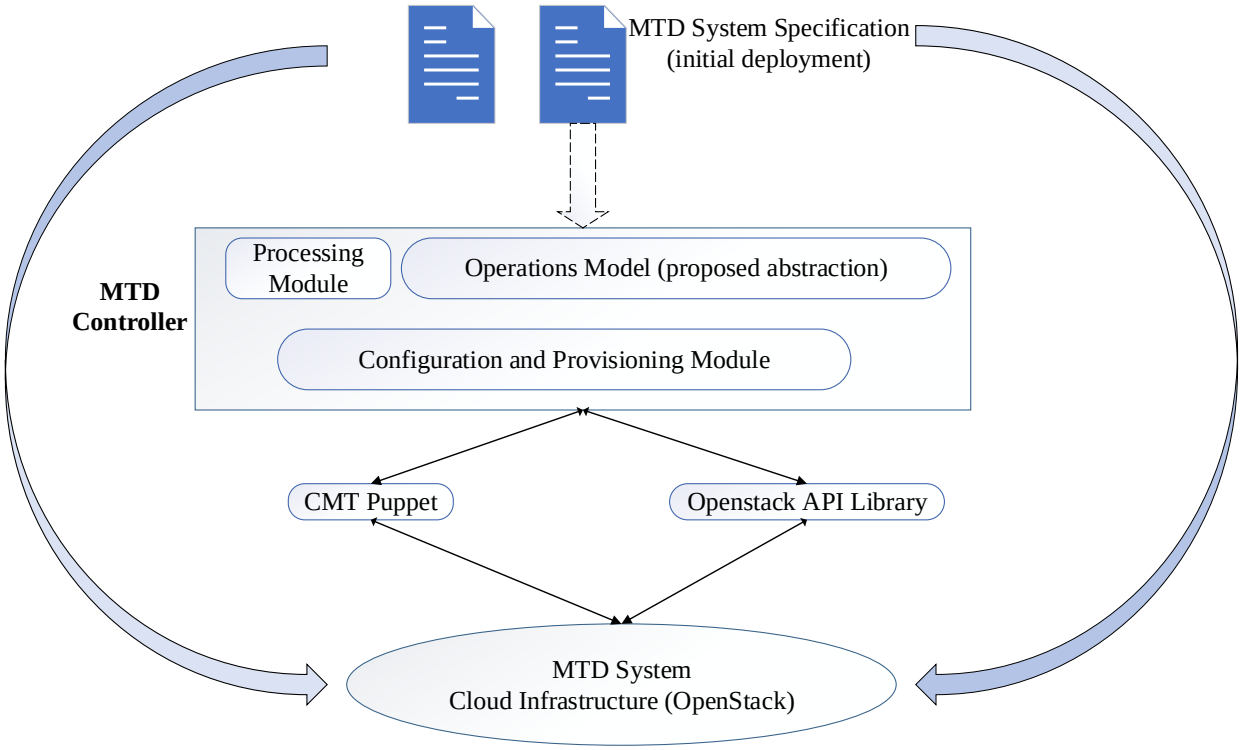



Fig. 10: A platform that takes an abstract specification of a cloud system as input and outputs the corresponding system on a cloud infrastructure. Advantages of cloud automation using Automated enterprise net-work compiler (ANCOR), which also includes performing live instance migration.

tion 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 synchrony between the two gateway routers, 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* [110] and *CryptoTrap* [111] deceive the attackers by imitating the true assets. DeceptionGrid provides agentless 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 help in countering 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* [153] (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* [109] 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

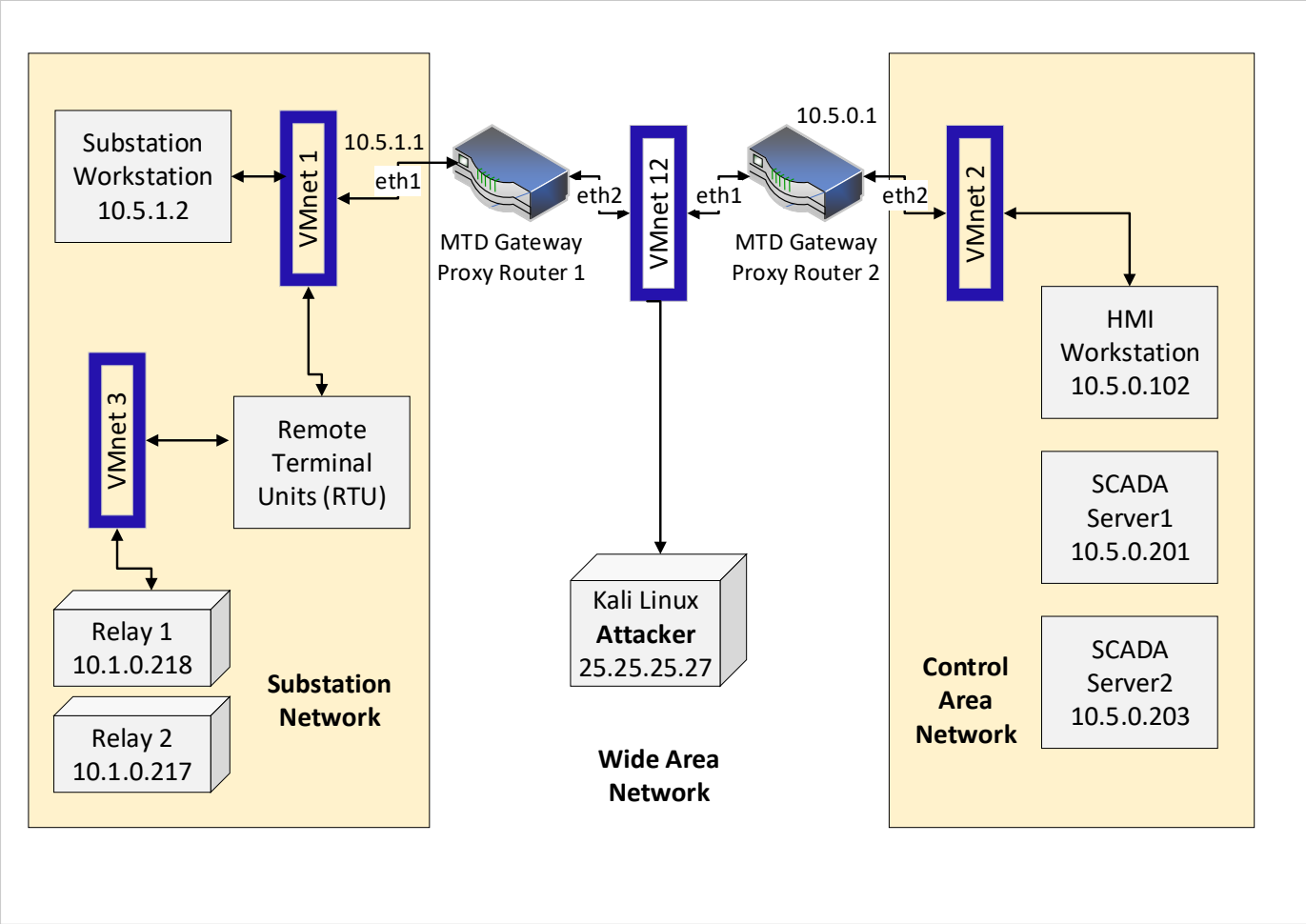



Fig. 11: MTD based architectural platform for IP hopping. The solution has been targeted at Supervisory Control and Data Access Systems (SCADA). The MTD gateway routers act as dynamic proxies, periodically mutating the IP address of the external interfaces exposed on the public Wide Area Network (WAN).

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* [124] 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 help us determine how effective a Moving Target Defense (MTD) will be or is. We first discuss the quality of the defender actions, i.e. the various system configurations it can choose to deploy, in terms of performance and security. This discussion helps us identify a set of qualitative metrics that, as we will see, most works either assume implicitly, or are simply unaware of.

We then devote a sub-section on qualitative evaluations metrics that are mathematical functions to help a defender gauge the cost-benefit, the risk analysis, etc. of an MTD. These, as we shall see, can be measured by simulation on test-beds or by using existing security metrics such as CVSS, attack graphs, 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 either when modeling or during evaluation. To evaluate the quality of an MTD, we first look at two major aspects– security considerations and performance considerations. Under each of

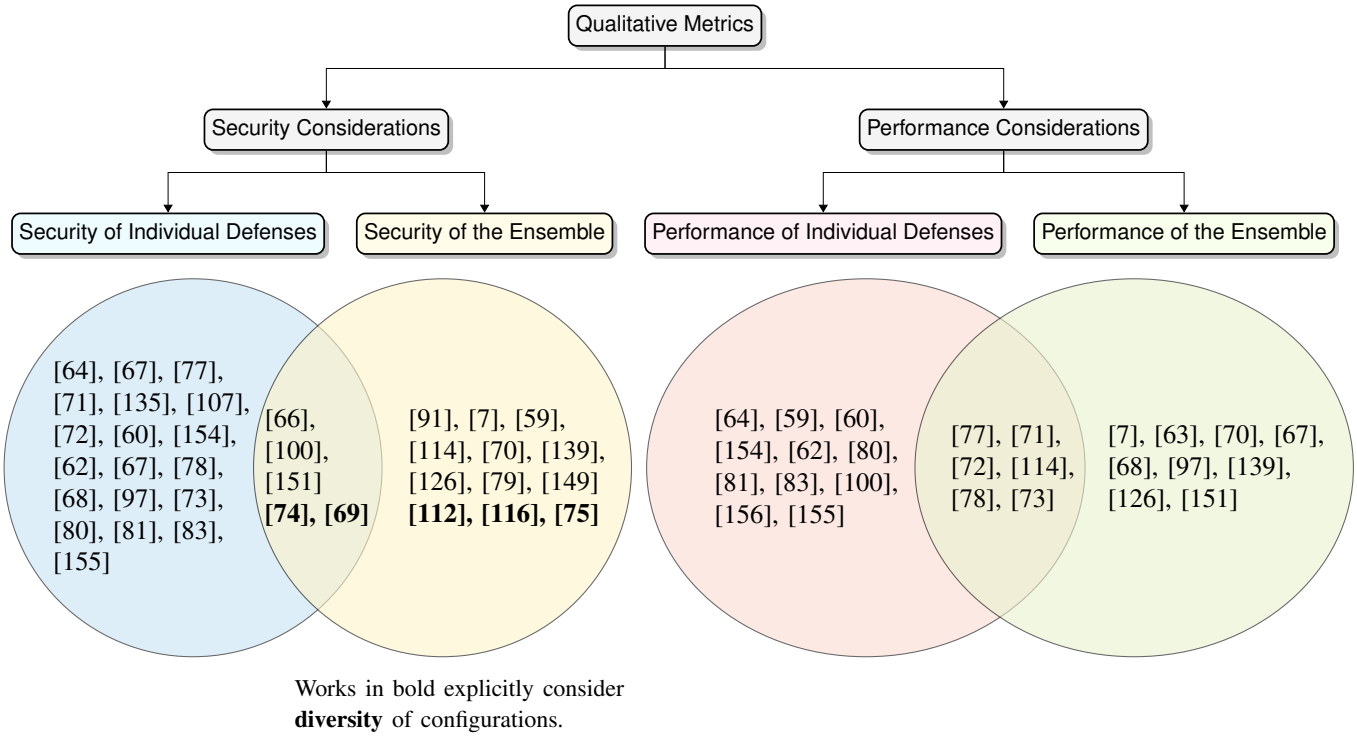



Fig. 12: A Moving Target Defense is more effective if it considers the security and performance impacts of the system configurations, both in unison and also when placed in an ensemble of system configurations that the MTD leverages. We try to categorize the works which explicitly either consider these metrics while modeling or during 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.

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 in which all the constituent system configurations are vulnerable to the same attack can never be successful even if implemented efficiently in practice.

In regards to quality metrics, we categorize the various MTDs in Figure 12. A work is a part of a set if they explicitly consider the particular qualitative metric when designing either the action set $A^{\mathcal{D}}$ (i.e. *what to move?*), the timing (i.e. *when to move?*), or the movement strategy (i.e. *how to move?*). Furthermore, 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 found that many MTD works only consider (and show) the improvement in regards to security and either assuming that it has no impact on performance or ignore that aspect altogether. 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 the 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 [64], [81], [62], [97], [69], [81], [155], [83]. For most of these works, the security metrics considered for each action are obtained using scoring metrics designed by experts like the Common Vulnerability Scoring Services (CVSS) discussed earlier. These works are able to come up with intelligent movement strategies based on reasoning over known vulnerabilities (more specifically, common vulnerabilities and exposures (CVE)) for which CVSS scores are readily available and may result in highly sub-optimal behavior when faced with zero-day vulnerabilities. In that regard, authors in [69] 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 [155], the authors try to model zero-day 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 lack of actual black-hat hackers who invent zero-day attacks in the set of security experts who annotate the defense methods), we believe the effectiveness of their MTD cannot be accurately determined. As to whether utility values can capture (and if so, how) the security risks associated with zero-day vulnerabilities is an interesting question and remains to be explored by research works.

Other works that also only capture the security of individual constituent system configurations are more domain or problem

specific in nature. In [77], 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 [71], authors choose an action (to migrate a VM or keep running) after reasoning about the security risks associated with the present vulnerable state. In [72], 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 [67] find a more compact way to model and continuously update the risk associated with deploying a particular defense action. Some other works that also consider the longitudinal effects of movement strategy, model the epistemic effects on the attacker's knowledge about the network. For example, [60] considers the fingerprint of the overall network when a particular defense action is selected, while [154] reason about the topology information a particular defense action leaks to an attacker. In contrast to all these works, [135] 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 [91], [7], [59], [70], [149]. 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 [139], 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 [126] 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 [112] 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 [116], [75], 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 [79] which seems to be an incorrect assumption to make given current research works [30], [85]. 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 [116], who create an approach at the compiler level to increase software diversity and [75], 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 [66], 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 [151], 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 [100], 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 [69] and [74] model both the diversity of constituent defenses present in the MTD ensemble and also model the security of each individual defense. In [74], 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 [69], 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) dealing with 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 first describe how the different MTD systems consider the performance costs associated with each individual configuration followed by the performance cost associated with the

ensemble. Lastly, we will highlight works which reason about both these factors that will affect the overall Quality of Service (QoS) when an MTD 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 [64], [80], [81], [83], [101]. 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 [64] and [101], 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 [80] and [81], 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.

For problem-specific settings, authors in [59] perform experiments to Most works look at problem-specific instances. Authors in [59] perform experiments to evaluate their MTD against crossfire attacks and notice that the packet delay from one point to another increase because a particular path selected at random is highly sub-optimal. On similar lines, in [60], the authors consider the performance cost of doing a *defend* action on legitimate user traffic. On a different note, [154] 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 [62], 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 [155] 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 [67] and [68], 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 the MTD defense to a static system configuration by measuring usability metrics such as latency, availability to legitimate users, etc. In [139], 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 one reducing this gap to an extent. On similar lines, authors in [126] notice that creating an entire file system replica takes two more minutes in the case of an MTD system when compared to a non-MTD enabled system refresh. On the contrary, [151] 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 [70] 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 [63], 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.

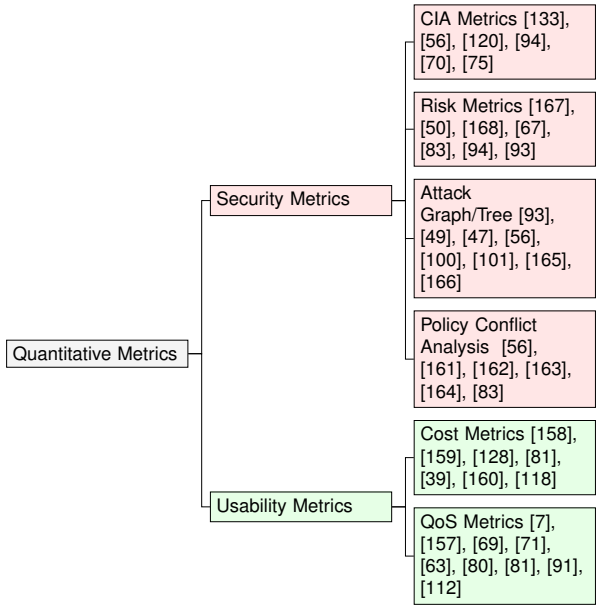



Fig. 13: Overview of Quantitative Metrics for MTD. The Security and Usability metrics research works have been categorized under sub-metrics along with research works that consider this metric while evaluating MTD.

*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 [77], 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 [73], 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 [71] consider the availability of each defense action over a *bounded time horizon*

| | |
|---|---|
| **Reconnaissance** | [7] [57] [120] [11] [12] [60] [158] |
| | [61] [66] [73] [75] [91] [94] [134] [141] |
| | [99] [100] [101] [135] [112] [114] [139] |
| **APT and Data Exfiltration** | [90] [128] [37] [169] [155] [128] [80] |
| **Vulnerability Exploitation** | [97] [68] [69] [71] [74] [70] [76] [101] |
| | [170] [118] [171] [126] [151] [172] [116] |
| **Multi-Stage Attack** | [165] [83] [56] [64] [98] [171] [173] [149] |
| **DDoS Attack** | [107] [59] [12] [72] [77] [134] [141] [158] |
| **Other types of Attack** | [78] [94] [93] [79] [81] [128] [80] [85], [58] |

TABLE VI: Threat Model for MTD Surveyed Papers. Most MTD research works with major threat models they target has been highlighted in the table.

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 [72] 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 [114] or cost of implementing a system that can support the various configuration [78]. More specifically, authors in [114] create a system that ensures that the logical virtual identifier distances between any two nodes remain the same while in [78], the authors consider the extra cost of creating an ensemble that is not necessary when not using MTD.

### *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.

*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 $\sim$ 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 $\sim 10\%$ research works). Also, some papers used other types of attacks such as timing-based attacks [78], and network intrusions [94], [93], [79], [81].

**CIA Metrics:** *Confidentiality*, *Integrity*, and *Availability* (CIA) are used as quantitative metrics for measurement of impact on system under attack. Zaffarano *et al.* [133] use 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. in [120] considered 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 [94] 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* [70] 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.* [75]. 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* [49] and *Attack Tree* [47] 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 [56] 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.

The Figure 14 shows system modules and operating layers, which are the part of SDN based MTD framework. The overlay

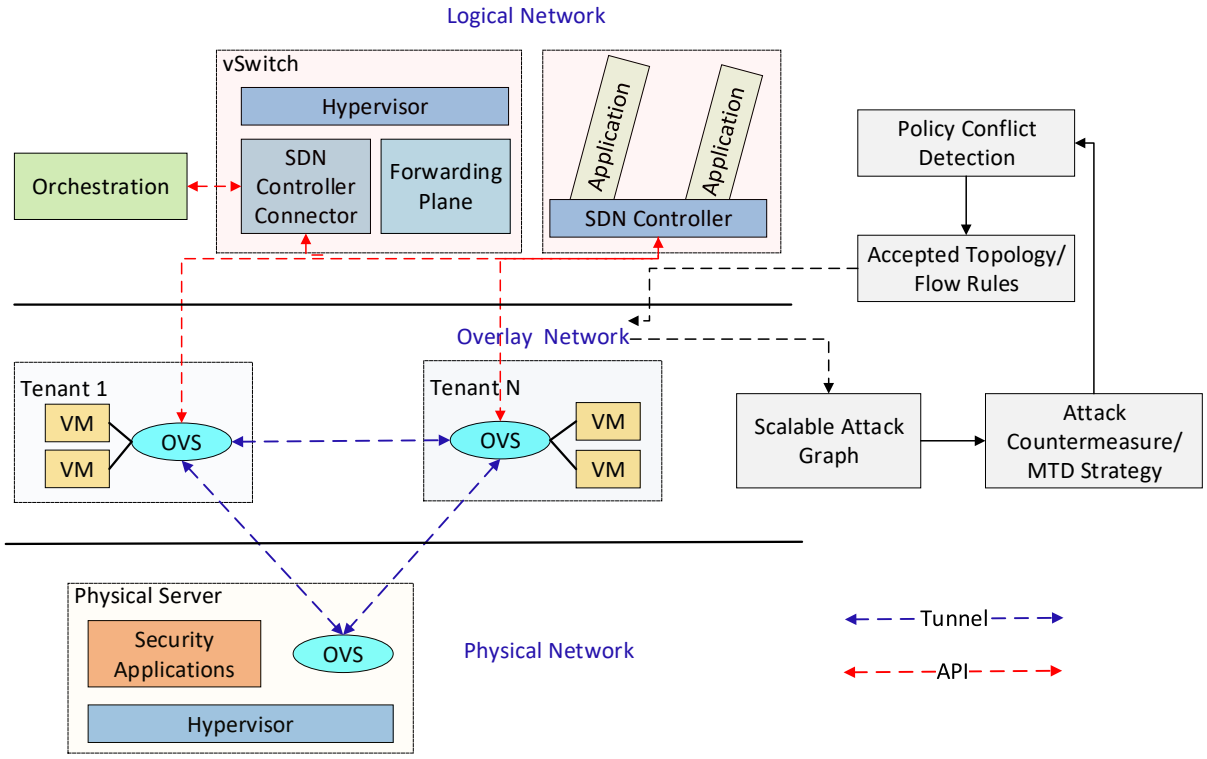



Fig. 14: System modules and operating layers of an MTD-based solution using SDN. The system comprises of quantitative metric based on vulnerabilities, attack graph at overlay network and policy conflict detection at the logical network layer.

network is responsible for vulnerability analysis, attack graph generation. The physical network consists of Open vSwitch (OVS), running on top of the physical server. The SDN controller interacts with OVS using OpenFlow APIs.

Bayesian attack graphs have been used by Miehling *et al.* [100] 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 [174]. 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 [167], the system administrator can determine how good the MTD solution is by examining the associated risk. Therefore, the authors [167] 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 [50].

Feedback-driven multi-stage MTD has been proposed by Zhu *et al.* [67] for dealing with multi-stage attacks like Stuxnet [175]. 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 [128], the authors provided 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.

Cheng *et al.* [64] considered the game theoretical part of MTD systems Specifically, the 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 [80]. 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 [176]. 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.* [93] provided 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* [93] 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.* [56] used 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.* [161], [162] 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 [163], loops, and blackholes in the network as discussed by Khursid *et al.* [164]. 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.* [7] 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.* [157]. 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. in [128] used 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.* [69]. 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 [71] model performance parameters such as availability, downtime, and downtime cost using a Continuous Time Markov Chain (CTMC) model. The experimental results show 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 basis for VM migration.

Sengupta *et al.* [81] analyze the performance impact of placing IDS (NIDS and HIDS) at all possible enforcement points in a cloud network. It is noteworthy that 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 16 Gbps in the case of a single detection agent to $\sim 6$ Gbps when 15 detection agents are placed.

CHAOS [91] analyzes how the delay intentionally introduced by MTD impacts the packet count in a SDN-managed

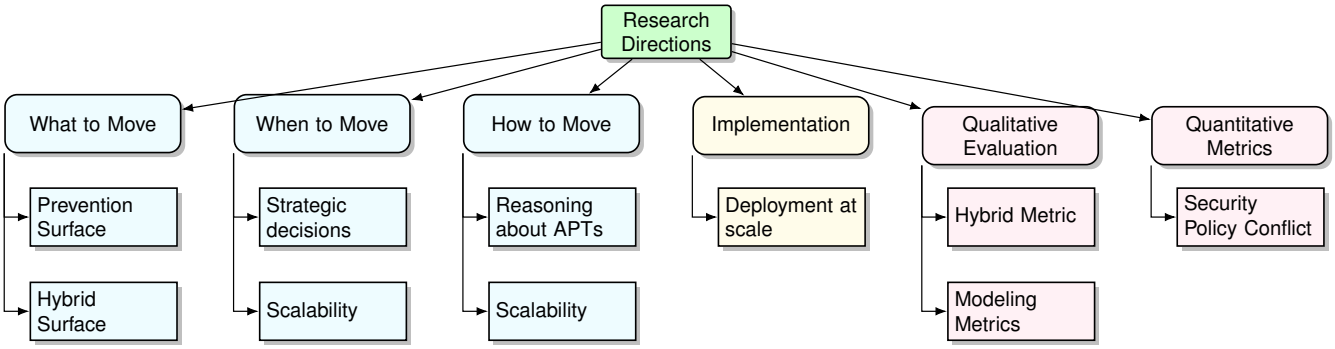



Fig. 15: Our categorizations help one identify various aspects that have not received attention during the development, implementation and evaluation of Moving Target Defenses (MTDs). It also creates a terminology that helps in quickly identifying related works (and novel research directions) for developing MTDs or simply describing a proposed MTD.

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 $1.5$s 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.* [158] presented a cost-effective MTD solution against DDoS and *Covert Channel* attacks. Through MTD adaptation, their work [158] 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.* [159] 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.* [128] 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 less studied aspects of Moving Target Defenses (MTDs). These lead to a discussion on promising directions for future research (a brief summary is highlighted in Figure 15).

### A. What to move

Works in MTD concentrated mostly on the movement of the exploration, the detection and the attack surface. While moving the exploration and the attack surface benefits the defender by taking away the advantage of reconnaissance for an attacker, movement of detection measures are often done to improve scalability and Quality of Service (QoS) metrics.

The movement of the prevention surface, which comprises of security modules such as Firewall and Intrusion Prevention Systems (IPS), has only been investigated by a couple of works (see discussion in Section III-A4). MTD research can consider exploring Next-Generation Firewall (NGFW) architectures, which combines security modules such as firewall, content filter, anti-virus, in order 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* [177] and *PAN Firewall* [178]. Further, with the rise of mobile technologies, MTDs can prove to be effective defenses. Although we discuss a few works on thwarting jamming attacks [63] and identity shifting in mobile ad-hoc networks [58], 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 should make one wonder about the possibility of creating logical surface level divisions. At that point, MTDs for shifting these surfaces introduce a new set of challenges for researchers to work on. For example, *Microsegmentation* [179] 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 [83], 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. When to Move

The timing problem has mostly been ignored in most work 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 [63], 60 seconds time periods are needed to defend against Network Mapping attacks [95] 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 address the timing problem theoretically, suffer from scalability issues. In [64], 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. How to Move

Randomized movement is a necessary part of effective MTDs. Emerging from notions of cryptography, mechanism designers in MTD often tend to believe that pure randomness or a Uniform Random Strategy (URS) offer the best defense [65]. Several works have argued that with information about known attacks, game-theoretic modeling can result in better strategies that offer higher gains in terms of both security and performance metrics. Unfortunately, scalability issues with the later methods encourage practitioners to choose URS in many cases.

Recently, it has been shown that attacks on networks resort to slow and low approaches like 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 [169], [155], they also suffer from scalability challenges, especially when faced with real-world cloud service providers who host thousands of hosts [83], [82]. Furthermore, works such as [100] and [101] that try to consider partial observability can limit scalability to less that 30 nodes. Study on the design of approximation approaches and their effectiveness when compared to scalable baselines such as URS is 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 [56] 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 forsee 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 [83]. 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 work, we look at various Moving Target Defenses (MTDs) that have been proposed for enhancing network security. First, we categorize them based at *what* surfaces these works move, *when* the move operation occurs, and *how* they move between the different constituent system configurations. 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 against such 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 also breifly describe a few test-beds that have either been leveraged by existing MTDs (or are similar) and encourage researchers to use them for evaluating the effectiveness of proposed MTDs. We conclude that SDN/NFV is a dominant technology used by MTDs and cyber-deception methods and that the industry adoption of MTD solutions is still limited to few application-security products.

Third, we discuss the various (qualitative and quantitative) metrics used for measuring the effectiveness of these MTDs under a single categorization that talks about both security and performance of the system. Then, we categorize the different works depending on whether an MTD either models a particular metric or evaluates its defense based on that metric. With this holistic view, we note that none of the MTDs consider all the different metrics we put forth. One wonders a defense that models all these metrics will be realizable in practice and prove to be better both in regards to performance and security.

Lastly, we highlight the areas of network security where the scope of developing MTDs have not been explored much and can prove to be quite effective in terms of improving security. We conclude by showcasing how our categorization provides a common terminology for researchers to describe and even develop future MTDs.

## VIII. Acknowledgement

This research is supported in part by following research grants: Naval Research Lab N00173-15-G017, AFOSR grant FA9550-18-1-0067, the NASA grant NNX17AD06G, ONR grants N00014-16-1-2892, N00014-18-1-2442, N00014-18-12840, NSFUS DGE-1723440, OAC-1642031, SaTC-1528099, 1723440 and NSFChina 61628201 and 61571375. Sailik Sengupta is supported by the IBM Ph.D. Fellowship. Also, Abdulhakim Sabur is a scholarship recipient from Taibah University through Saudi Arabian Cultural Mission (SACM).

## References

[1] ESDS, "Cloud Computing Trends Driving Growth in 2018," https://www.esds.co.in/blog/cloud-computing-trends-driving-growth-2018/#sthash.Ke1nS90k.dpbs, 2018, online; accessed 11 Nov 2018.

[2] Kasey Panetta, "Gartners Top 10 Security Predictions 2016," https://www.gartner.com/smarterwithgartner/top-10-security-predictions-2016/, 2016, online; accessed 11 Nov 2018.

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

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

[5] R. Saha and A. Agarwal, "SDN approach to large scale global data centers," *Proceedings of Open Networking Summit, Santa Clara, California, USA*, 2012.

[6] C.-J. Chung, T. Xing, D. Huang, D. Medhi, and K. Trivedi, "SeReNe: On establishing secure and resilient networking services for an sdn-based multi-tenant datacenter environment," in *Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4–11.

[7] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 127–132.

[8] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *2010 43rd Hawaii International Conference on System Sciences*. IEEE, 2010, pp. 1–10.

[9] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein, "Survey of cyber moving target techniques," MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, Tech. Rep., 2013.

[10] 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, 2014.

[11] K. A. Farris and G. Cybenko, "Quantification of moving target cyber defenses," in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security, Defense, and Law Enforcement XIV*, vol. 9456. International Society for Optics and Photonics, 2015, p. 94560L.

[12] G.-l. Cai, B.-s. Wang, W. Hu, and T.-z. Wang, "Moving target defense: state of the art and characteristics," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 11, pp. 1122–1153, 2016.

[13] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Security and Communication Networks*, vol. 2018, 2018.

[14] O. S. V. D. (OSVDB), "Open source vulnerability database (osvdb)," 2012.

[15] C. V. CVE, "Exposures," 2012.

[16] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, 2019.

[17] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2014, pp. 63–72.

[18] L. Martin, "Cyber kill chain (ckc)," https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html, 2017, online; accessed 11 Nov 2018.

[19] K. Glass and R. Colbaugh, "Web analytics for security informatics," in *Intelligence and Security Informatics Conference (EISIC), 2011 European*. IEEE, 2011, pp. 214–219.

[20] D. Ackley, M. Plank, L. Johnson, and K. Glass, "Cactus: A web crawling and indexing tool," *ICASA Technical Report*, 2009.

[21] P. Mehra, "A brief study and comparison of snort and bro open source network intrusion detection systems," *International Journal of Advanced Research in Computer and Communication Engineering*, 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*. Prentice Hall Professional, 2003.

[23] P. Deshpande, S. Sharma, S. Peddoju, and S. Junaid, "Hids: A host based intrusion detection system for cloud computing environment," *International Journal of System Assurance Engineering and Management*, 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 *Proceedings of the 2nd ACM Conference on Computer and Communications Security*. ACM, 1994, pp. 18–29.

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

[26] H. Welte, "The netfilter framework in linux 2.4," in *Proceedings of Linux Kongress*, 2000.

[27] F. Pouget, M. Dacier *et al.*, "Honeypot-based forensics," in *AusCERT Asia Pacific Information Technology Security Conference*, 2004.

[28] M. Oosterhof, "Cowrie honeypot," 2014.

[29] SANS, "Intrusion detection evasion: How attackers get past the burglar alarm," https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-evasion-attackers-burglar-alarm-1284, 2003, online; accessed 11 Nov 2018.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

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

[32] N. Stojanovski, M. Gusev, D. Gligoroski, and S. J. Knapskog, "Bypassing data execution prevention on microsoftwindows xp sp2," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. IEEE, 2007, pp. 1222–1226.

[33] M. I. Center, "Apt1: Exposing one of chinas cyber espionage units," *Mandian. com*, 2013.

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

[35] 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.

[36] R. Kissel, *Glossary of key information security terms*. Diane Publishing, 2011.

[37] Z. Shu and G. Yan, "Ensuring deception consistency for ftp services hardened against advanced persistent threats," in *Proceedings of the 5th ACM Workshop on Moving Target Defense*. ACM, 2018, pp. 69–79.

[38] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*. IEEE, 2006, pp. 121–130.

[39] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*. IEEE, 2012, pp. 1–12.

[40] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*. IEEE, 2002, pp. 49–63.

[41] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 336–345.

[42] J. Lee, H. Lee, and H. P. In, "Scalable attack graph for risk assessment," in *Information Networking, 2009. ICOIN 2009. International Conference on*. IEEE, 2009, pp. 1–5.

[43] J. Homer, X. Ou, and M. A. McQueen, "From attack graphs to automated configuration managementan iterative approach," *Kansas State University Technical Report*, 2008.

[44] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 18–34.

[45] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost sat solving," in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011, pp. 31–40.

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

[47] B. Schneier, "Attack trees," *Dr. Dobbs journal*, vol. 24, no. 12, pp. 21–29, 1999.

[48] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *Icisc*, vol. 3935. Springer, 2005, pp. 186–198.

[49] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in *International Symposium on Formal Methods for Components and Objects*. Springer, 2003, pp. 344–371.

[50] J. B. Hong and D. S. Kim, "Scalable security model generation and analysis using k-importance measures," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2013, pp. 270–287.

[51] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 217–224.

[52] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *null*. IEEE, 2002, p. 273.

[53] J. B. Hong, D. S. Kim, C.-J. Chung, and D. Huang, "A survey on the usability and practical applications of graphical security models," *Computer Science Review*, vol. 26, pp. 1–16, 2017.

[54] X. Ou, S. Govindavajhala, and A. W. Appel, "Mulval: A logic-based network security analyzer." in *USENIX Security Symposium*. Baltimore, MD, 2005, pp. 8–8.

[55] J. B. Hong and D. S. Kim, "Performance analysis of scalable attack representation models," in *IFIP International Information Security Conference*. Springer, 2013, pp. 330–343.

[56] A. Chowdhary, S. Pisharody, and D. Huang, "Sdn based scalable mtd solution in cloud network," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 2016, pp. 27–36.

[57] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2012, pp. 310–327.

[58] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun, "A moving target defense mechanism for manets based on identity virtualization," in *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2013, pp. 278–286.

[59] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*. IEEE, 2016, pp. 627–630.

[60] Z. Zhao, F. Liu, and D. Gong, "An sdn-based fingerprint hopping method to prevent fingerprinting attacks," *Security and Communication Networks*, vol. 2017, 2017.

[61] A. Schlenker, O. Thakoor, H. Xu, F. Fang, M. Tambe, L. Tran-Thanh, P. Vayanos, and Y. Vorobeychik, "Deceiving cyber adversaries: A game theoretic approach," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 892–900.

[62] S. Jajodia, N. Park, E. Serra, and V. Subrahmanian, "Share: A stackelberg honey-based adversarial reasoning engine," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 3, p. 30, 2018.

[63] R. Algin, H. O. Tan, and K. Akkaya, "Mitigating selective jamming attacks in smart meter data collection using moving target defense," in *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. ACM, 2017, pp. 1–8.

[64] 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.

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

[66] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," in *Moving Target Defense II*. Springer, 2013, pp. 1–13.

[67] Q. Zhu and T. Başar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *International Conference on Decision and Game Theory for Security*. Springer, 2013, pp. 246–263.

[68] S. Sengupta, S. G. Vadlamudi, S. Kambhampati, A. Doupé, Z. Zhao, M. Taguinod, and G.-J. Ahn, "A game theoretic approach to strategy generation for moving target defense in web applications," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 178–186.

[69] K. M. Carter, J. F. Riordan, and H. Okhravi, "A game theoretic approach to strategy determination for dynamic platform defenses," in *Proceedings of the First ACM Workshop on Moving Target Defense*. ACM, 2014, pp. 21–30.

[70] M. Thompson, N. Evans, and V. Kisekka, "Multiple os rotational environment an implemented moving target defense," in *Resilient Control Systems (ISRCS), 2014 7th International Symposium on*. IEEE, 2014, pp. 1–6.

[71] I. El Mir, A. Chowdhary, D. Huang, S. Pisharody, D. S. Kim, and A. Haqiq, "Software defined stochastic model for moving target defense," in *International Afro-European Conference for Industrial Advancement*. Springer, 2016, pp. 188–197.

[72] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev, "Frequency-minimal moving target defense using software-defined networking," in *2016 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2016, pp. 1–6.

[73] A. Prakash and M. P. Wellman, "Empirical game-theoretic analysis for moving target defense," in *Proceedings of the Second ACM Workshop on Moving Target Defense*. ACM, 2015, pp. 57–65.

[74] S. Neti, A. Somayaji, and M. E. Locasto, "Software diversity: Security, entropy and game theory."

[75] M. Crouse, E. W. Fulp, and D. Canas, "Improving the diversity defense of genetic algorithm-based moving target approaches," in *Proceedings of the National Symposium on Moving Target Research*, 2012.

[76] B. Bohara, "Moving target defense using live migration of docker containers," Ph.D. dissertation, Arizona State University, 2017.

[77] A. Chowdhary, S. Pisharody, A. Alshamrani, and D. Huang, "Dynamic game based security framework in sdn-enabled cloud networking environments," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2017, pp. 53–58.

[78] A. Clark, K. Sun, L. Bushnell, and R. Poovendran, "A game-theoretic approach to ip address randomization in decoy-based cyber defense," in *International Conference on Decision and Game Theory for Security*. Springer, 2015, pp. 3–21.

[79] R. Colbaugh and K. Glass, "Predictability-oriented defense against adaptive adversaries," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2721–2727.

[80] S. Venkatesan, M. Albanese, G. Cybenko, and S. Jajodia, "A moving target defense approach to disrupting stealthy botnets," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 2016, pp. 37–46.

[81] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "Moving target defense for the placement of intrusion detection systems in the cloud," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 326–345.

[82] ——, "General sum markov games for strategic detection of advanced persistent threats using moving target defense in cloud networks," 2019.

[83] A. Chowdhary*, S. Sengupta*, D. Huang, and S. Kambhampati, "Markov game modeling of moving target defense for strategic detection of threats in cloud networks," *AAAI Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2019.

[84] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[85] S. Sengupta, T. Chakraborti, and S. Kambhampati, "Mtdeep: boosting the security of deep neural nets against adversarial attacks with moving target defense," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

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

[88] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 485–492.

[89] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," in *2016 International Conference on Information Science and Security (ICISS)*. IEEE, 2016, pp. 1–5.

[90] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "Flipit: The game of stealthy takeover," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013.

[91] Y. Shi, H. Zhang, J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, F. Yan, and B. Zhao, "Chaos: An sdn-based moving target defense system," *Security and Communication Networks*, vol. 2017, 2017.

[92] I. El Mir, A. Chowdhary, D. Huang, S. Pisharody, D. S. Kim, and A. Haqiq, "Software defined stochastic model for moving target defense," in *International Afro-European Conference for Industrial Advancement*. Springer, 2016, pp. 188–197.

[93] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE transactions on dependable and secure computing*, vol. 10, no. 4, pp. 198–211, 2013.

[94] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "Mtd analysis and evaluation framework in software defined network (mason)," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2018, pp. 43–48.

[95] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

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

[97] S. G. Vadlamudi, S. Sengupta, M. Taguinod, Z. Zhao, A. Doupé, G.-J. Ahn, and S. Kambhampati, "Moving target defense for web applications using bayesian stackelberg games," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1377–1378.

[98] A. Chowdhary, S. Sengupta, A. Alshamrani, D. Huang, and A. Sabur, "Adaptive mtd security using markov game modeling," *arXiv preprint arXiv:1811.00651*, 2018.

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

[100] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on bayesian attack graphs," in *Proceedings of the Second ACM Workshop on Moving Target Defense*. ACM, 2015, pp. 67–76.

[101] T. H. Nguyen, M. Wright, M. P. Wellman, and S. Singh, "Multistage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis," *Security and Communication Networks*, vol. 2018, 2018.

[102] 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 of Things Journal*, 2019.

[103] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

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

[105] S. Valizadeh and M. van Dijk, "Toward a theory of cyber attacks," *arXiv preprint arXiv:1901.01598*, 2019.

[106] Y. Vorobeychik and S. Singh, "Computing stackelberg equilibria in discounted stochastic games (corrected version)," 2012.

[107] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo, "Ddos defense using mtd and sdn."

[108] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[109] Polyverse, "Moving target defense: Redefining the power of defense," https://view.attach.io/ByfWW3KGf, 2018, online; accessed 11 Feb 2019.

[110] TrapX, "Deceptiongrid trapx," http://trapx.com/wp-content/uploads/2018/03/, 2018, online; accessed 11 Feb 2019.

[111] CryptoTrap, "Cryptotrap trapx," http://trapx.com/landing/wp-content/uploads/2018/12/Product_Brief_TrapX_CryptoTrap.pdf, 2018, online; accessed 11 Feb 2019.

[112] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online mtd," in *Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on*. IEEE, 2017, pp. 234–243.

[113] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.

[114] Y. Wang, Q. Chen, J. Yi, and J. Guo, "U-tri: unlinkability through random identifier for sdn network," in *Proceedings of the 2017 Workshop on Moving Target Defense*. ACM, 2017, pp. 3–15.

[115] OSRG, "Ryu sdn controller," https://osrg.github.io/ryu/, 2017.

[116] A. Homescu, T. Jackson, S. Crane, S. Brunthaler, P. Larsen, and M. Franz, "Large-scale automated software diversityprogram evolution

redux," *IEEE Transactions on Dependable and Secure Computing*, no. 1, pp. 1–1.

[117] C. Lattner and V. Adve, "The llvm compiler framework and infrastructure tutorial," in *International Workshop on Languages and Compilers for Parallel Computing*. Springer, 2004, pp. 15–16.

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

[119] S. Baxter and L. C. Vogt, "Content management system," Mar. 12 2002, uS Patent 6,356,903.

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

[121] S. Scherfke and O. Lnsdorf, "Simpy testbed," https://simpy.readthedocs.io/en/3.0/.

[122] R. Ricci, E. Eide, and C. Team, "Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications," *; login:: the magazine of USENIX & SAGE*, vol. 39, no. 6, pp. 36–38, 2014.

[123] A. C. Pappa, "Moving target defense for securing smart grid communications: Architectural design, implementation and evaluation," Ph.D. dissertation, Iowa State University, 2016.

[124] Morphisec, "Moving target defense," https://www.morphisec.com/resources, 2018, online; accessed 12 Feb 2019.

[125] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[126] N. O. Ahmed and B. Bhargava, "Mayflies: A moving target defense framework for distributed systems," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 2016, pp. 59–64.

[127] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

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

[129] J. Yackoski, H. Bullen, X. Yu, and J. Li, "Applying self-shielding dynamics to the network architecture," in *Moving target defense II*. Springer, 2013, pp. 97–115.

[130] J. Yackoski, J. Li, S. A. DeLoach, and X. Ou, "Mission-oriented moving target defense based on cryptographically strong network dynamics," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. ACM, 2013, p. 57.

[131] J. Yackoski, P. Xie, H. Bullen, J. Li, and K. Sun, "A self-shielding dynamic network architecture," in *2011-MILCOM 2011 Military Communications Conference*. IEEE, 2011, pp. 1381–1386.

[132] R. Chadha, T. Bowen, C.-Y. J. Chiang, Y. M. Gottlieb, A. Poylisher, A. Sapello, C. Serban, S. Sugrim, G. Walther, L. M. Marvel *et al.*, "Cybervan: A cyber security virtual assured network testbed," in *Military Communications Conference, MILCOM 2016-2016 IEEE*. IEEE, 2016, pp. 1125–1130.

[133] K. Zaffarano, J. Taylor, and S. Hamilton, "A quantitative framework for moving target defense effectiveness evaluation," in *Proceedings of the Second ACM Workshop on Moving Target Defense*. ACM, 2015, pp. 3–10.

[134] 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 *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 263–270.

[135] P. Kampanakis, H. Perros, and T. Beyene, "Sdn-based solutions for moving target defense network protection," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*. IEEE, 2014, pp. 1–6.

[136] Q. Jia, K. Sun, and A. Stavrou, "Motag: Moving target defense against internet denial of service attacks," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. IEEE, 2013, pp. 1–9.

[137] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[138] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[139] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, "Mt6d: A moving target ipv6 defense," in *2011-MILCOM 2011 Military Communications Conference*. IEEE, 2011, pp. 1321–1326.

[140] S. Thomson, T. Narten, and T. Jinmei, "Ipv6 stateless address autoconfiguration," Tech. Rep., 2007.

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

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

[143] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.

[144] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 3–14.

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

[146] A. Chowdhary, A. Alshamrani, D. Huang, M. Kang, A. Kim, and A. Velazquez, "Trufl: Distributed trust management framework in sdn," *arXiv preprint arXiv:1811.00635*, 2018.

[147] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[148] D. Fudenberg, D. Levine, and E. Maskin, "The folk theorem with imperfect public information," in *A Long-Run Collaboration On Long-Run Games*. World Scientific, 2009, pp. 231–273.

[149] 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 *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*. IEEE, 2013, pp. 162–169.

[150] A. Kosowski and V. Mosorov, "Nessi2 simulator for large-scale ddos attack analysis," in *Perspective Technologies and Methods in MEMS Design*. IEEE, 2011, pp. 157–159.

[151] A. G. Bardas, S. C. Sundaramurthy, X. Ou, and S. A. DeLoach, "Mtd cbits: Moving target defense for cloud-based it systems," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 167–186.

[152] 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.

[153] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, "Heartbleed 101," *IEEE security & privacy*, vol. 12, no. 4, pp. 63–67, 2014.

[154] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "Nethide: secure and practical network topology obfuscation," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 693–709.

[155] S. Rass, S. König, and S. Schauer, "Defending against advanced persistent threats using game-theory," *PloS one*, vol. 12, no. 1, p. e0168675, 2017.

[156] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[157] M. Crouse, B. Prosser, and E. W. Fulp, "Probabilistic performance analysis of moving target and deception reconnaissance defenses," in *Proceedings of the Second ACM Workshop on Moving Target Defense*. ACM, 2015, pp. 21–29.

[158] H. Wang, F. Li, and S. Chen, "Towards cost-effective moving target defense against ddos and covert channel attacks," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. ACM, 2016, pp. 15–25.

[159] C. Lei, D.-h. Ma, H.-q. Zhang, and L.-m. Wang, "Moving target network defense effectiveness evaluation based on change-point detection," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[160] 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 *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 573–578.

[161] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "Brew: A security policy analysis framework for distributed sdn-based cloud environments," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[162] S. Pisharody, A. Chowdhary, and D. Huang, "Security policy checking in distributed sdn based clouds," in *Communications and Network Security (CNS), 2016 IEEE Conference on*. IEEE, 2016, pp. 19–27.

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

[164] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, 2013, pp. 15–27.

[165] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2008, pp. 283–296.

[166] A. Roy, D. S. Kim, and K. S. Trivedi, "Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees," *Security and Communication Networks*, vol. 5, no. 8, pp. 929–943, 2012.

[167] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, 2016.

[168] M. Schiffman and C. Cisco, "A complete guide to the common vulnerability scoring system (cvss)," *White paper. Identification of Basic Measurable Security Components in Software Intensive Systems*, 2005.

[169] S. Rass and Q. Zhu, "Gadapt: A sequential game-theoretic framework for designing defense-in-depth strategies against advanced persistent threats," in *International Conference on Decision and Game Theory for Security*. Springer, 2016, pp. 314–326.

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

[171] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *National symposium on moving target research*, vol. 246, 2012.

[172] J. Xu, P. Guo, M. Zhao, R. F. Erbacher, M. Zhu, and P. Liu, "Comparing different moving target defense techniques," in *Proceedings of the First ACM Workshop on Moving Target Defense*. ACM, 2014, pp. 97–107.

[173] J. B. Hong and D. S. Kim, "Scalable security models for assessing effectiveness of moving target defenses," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*. IEEE, 2014, pp. 515–526.

[174] 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 *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, 2019, pp. 473–485.

[175] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[176] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 133–145.

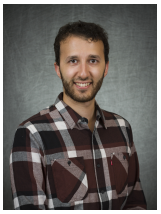
[177] J. Frahim, O. Santos, and A. Ossipov, *Cisco ASA: All-in-one Next-Generation Firewall, IPS, and VPN Services*. Cisco Press, 2014.

[178] P. Alto, "Palo alto next generation firewall," 2018.

[179] O. Mämmelä, J. Hiltunen, J. Suomalainen, K. Ahola, P. Mannersalo, and J. Vehkaperä, "Towards micro-segmentation in 5g network security," in *European Conference on Networks and Communications (EuCNC 2016) Workshop on Network Management, Quality of Service and Security for 5G Networks*, 2016.


**Sailik Sengupta** is a Ph.D. student in Computer Science at Arizona State University. He received his B.E. in Computer Science and Engineering from Jadavpur University (2013) and then worked for Amazon where he became an application-security certifier. Sailik is interested in solving problems that arise in multi-agent 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. Sailik was awarded the IBM Ph.D. fellowship in 2018.

**Ankur Chowdhary** is a Ph.D. candidate in Computer Science at Arizona State University, Tempe, AZ, USA. He received B.Tech in Information Technology from GGSIPU in 2011 and MS in Computer Science from ASU in 2015. He has worked as Information Security Researcher for Blackberry Ltd., RSG and Application Developer for CSC Pvt. Ltd. His research interests include SDN, Web Security, Network Security, and application of AI and Machine Learning in the field of Security.



**Abdulhakim Sabur** is a Ph.D student in Computer Engineering at Arizona state University, Tempe, AZ, USA. He received his B.S. degree (with Honor) in Computer Science and Engineering from King Saud University, Saudi Arabia in 2015 and a Master degree in Computer Engineering from Arizona State University in 2018. He worked as a researcher at the National Center for Cybersecurity (C4C) in King Abdulaziz City for Science and Technology (KACST) and A teaching assistant in Taibah University. His research interest include Network and information security, vulnerability analysis and management, automated policy and security checking in software defined networking systems.

**Dijiang Huang** received the B.S. degree from 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. His research interests include computer networking, security, and privacy. He is an Associate Editor of the Journal of Network and System Management (JNSM) and the IEEE Communications Surveys and Tutorials. 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 (CES). He was the recipient of the ONR Young Investigator Program (YIP) Award.

**Adel Alshamrani** is an assistant professor in department of cybersecurity, College of Computer Science and Engineering at University of Jeddah, Saudi Arabia. He received his B.S. degree in computer science from Umm Al-Qura University, Saudi Arabia in 2007, M.S. degree in computer science from La Trobe University Melbourne, Australia, in 2010, and PhD in computer science from Arizona State University in 2018. He has eight years of work experience in information security, network engineering, and teaching while working in the Faculty of Computing and Information Technology, King Abdul Aziz University, and University of Jeddah. His research interests include information security, intrusion detection, and software defined networking. He is the Chief Information Security Officer (CISO) at the University of Jeddah.

**Subbarao Kambhampati (Rao)** is a professor of Computer Science at Arizona State University. He received his B.Tech. in Electrical Engineering (Electronics) from Indian Institute of Technology, Madras (1983), and M.S.(1985) and Ph.D.(1989) in Computer Science (1985,1989) from University of Maryland, College Park. Kambhampati studies fundamental problems in planning, decision making, and game theory. Kambhampati is a fellow of AAAI and AAAS, and was an NSF Young Investigator. He received multiple teaching awards, including a university last lecture recognition. Kambhampati is the past president of AAAI and was a trustee of IJCAI. 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. Kambhampati's research, as well as his views on the progress and societal impacts of AI, have been featured in multiple national and international media outlets.