



Article

A Survey on Moving Target Defense for Networks: A Practical View

Łukasz Jalowski ¹, Marek Zmuda ² and Mariusz Rawski ^{3,*}¹ Independent Researcher, 80-126 Gdansk, Poland² Intel Corporation, 80-298 Gdansk, Poland³ Faculty of Electronics and Information Technology, Warsaw University of Technology, 00-661 Warsaw, Poland

* Correspondence: mariusz.rawski@pw.edu.pl

Abstract: The static nature of many of currently used network systems has multiple practical benefits, including cost optimization and ease of deployment, but it makes them vulnerable to attackers who can observe from the shadows to gain insight before launching a devastating attack against the infrastructure. Moving target defense (MTD) is one of the emerging areas that promises to protect against this kind of attack by continuously shifting system parameters and changing the attack surface of protected systems. The emergence of network functions virtualization (NFV) and software-defined networking (SDN) technology allows for the implementation of very sophisticated MTD techniques. Furthermore, the introduction of such solutions as field-programmable gate array (FPGA) programmable acceleration cards makes it possible to take the MTD concept to the next level. Applying hardware acceleration to existing concepts or developing new, dedicated methods will offer more robust, efficient, and secure solutions. However, to the best of the authors' knowledge, there are still no major implementations of MTD schemes inside large-scale networks. This survey aims to understand why, by analyzing research made in the field of MTD to show current pitfalls and possible improvements that need to be addressed in future proposals to make MTD a viable solution to address current cybersecurity threats in real-life scenarios.

Keywords: cybersecurity; cyberdefense; network security; moving target defense



Citation: Jalowski, Ł.; Zmuda, M.; Rawski, M. A Survey on Moving Target Defense for Networks: A Practical View. *Electronics* **2022**, *11*, 2886. <https://doi.org/10.3390/electronics11182886>

Received: 9 August 2022

Accepted: 29 August 2022

Published: 12 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Motivations behind Moving Target Defense

Many currently used network systems are static. They are designed, built, and configured once to remain unchanged for a long time [1]. This approach is reasonable from a functional requirements perspective: if use cases are well defined, then network architecture and topology are designed to meet that requirements and optimize cost. On the other hand, the static nature of modern networks gives an unfair advantage to the potential attackers, who can observe the network for a long time to collect data on end-points or traffic patterns that remain unnoticed. Their first goal is to find security vulnerabilities to finally use them to perform the attack at a convenient moment. Because of the static nature of existing networks, data collected this way might remain valid for long periods of time. A blog post [2] presents data from various cybersecurity reports. Reports from 98 data breach statistics were compiled into one post in [3], covering types of breaches, industry-specific stats, risks, costs, defense, and prevention resources. Sources report this situation as a significant threat to the security of modern networks [4].

The static nature of many network systems is only made worse by the fact that many of the prevailing cybersecurity defense mechanisms are reactive by design. Some examples of such an approach can be applying a security software patch that fixes a zero-day in an operating system or an antivirus system updating the threat database after a new malware has been detected. Shortcomings of this reactive attitude are relatively easy to

spot. Before a patch for a zero-day vulnerability can be applied, the issue first has to be recognized and fixed by the company responsible for the software. The process of finding such a security flaw is not necessarily trivial. Companies can rely on their internal security teams or external security researchers who often expect monetary compensation for their discoveries. However, there are other companies that specialize in buying zero-day vulnerabilities, like the Zerodium bug bounty platform [5], which later resells them to interested parties such as governments. In-depth talk about this kind of market can be found in [6]. Similarly, a system can only be protected against a malware that is known and has been fingerprinted previously. Before that happens, the malicious software can achieve truly devastating results. NotPetya, described as the most devastating cyberattack in history [7], is an example of such a powerful attack. All of this shows how much of an asymmetric game the defenders play with the attackers. After getting a foothold inside compromised systems via a zero-day vulnerability, the attackers can then spend weeks or months slowly accumulating information about the network, its topology, main data servers etc. Such data can then be used to identify the most valuable targets and potentially exfiltrate data.

In 2011, Lockheed Martin Corporation proposed the Intrusion Kill Chain Model (Figure 1), which distinguishes particular phases of the attack [8]. The model assumes that every malicious campaign begins at the reconnaissance stage. Later stages are payload delivery and installation, which leads to privilege escalation via found vulnerabilities and lateral movement inside target network. Finally, when ready, the attacker performs action on a target and exfiltrates. This is an example of the so-called linear kill chain. More talk about intrusion kill chains and their types can be found in [4]. Effective defense at early points would not allow the attacker to cause any damage (no material, financial, data, or reputation loss) even if potential zero-day exploits for the target system are available [9].

These arguments pay particular attention to security researchers to work on solutions that make systems resistant to reconnaissance. One solution to this problem has been introduced in [10], a result of the USA Federal Networking and Information Technology Research and Development program working to find novel ways to avoid some of the most pressing cybersecurity problems. It proposed a radical shift in thinking about computer networks, called moving target defense (MTD). Its main idea is that an attack can work at most a single time, if at all. To achieve this goal, the authors proposed to constantly reconfigure the system, such that the same attack vector cannot be reused in the future. MTD is a technique that dynamically shifts the attack surface to increase complexity and cost for attackers, limits the exposure of vulnerabilities and opportunities for attack, and increases system resiliency [11–17]. High hopes are associated with this technology, which is expressed by funding numerous research and development grants by government institutions and the private sector [18]. This goal can be achieved in many ways, but MTD still has no clear direction that would soon become the industrial standard.

The core idea behind MTD has been around for many years prior to [10]. Perhaps the best known example is address space layout randomization (ASLR) [19]. This technique is based on thwarting attackers by randomizing the process memory space layout. In doing so, it effectively renders some of the attack ineffective by making them unreliable. Conceptually, it represents the same idea as the MTD works published later—prevent the attack, instead of dealing with it when it happens, by introducing unpredictability into the system. Although this example proves that the idea behind MTD is correct in general and although the huge number of published works on the topic in recent years, as to the best of authors' knowledge there has not been any significant commercial implementations of MTD techniques in existing networks. The main reason for this is the lack of generally accepted, industry-wide standards, the lack of metrics to assess the effectiveness of individual MTD solutions, and the lack of systematic research that would show the cost of MTD implementation in real systems, like efficiency per dollar spent.

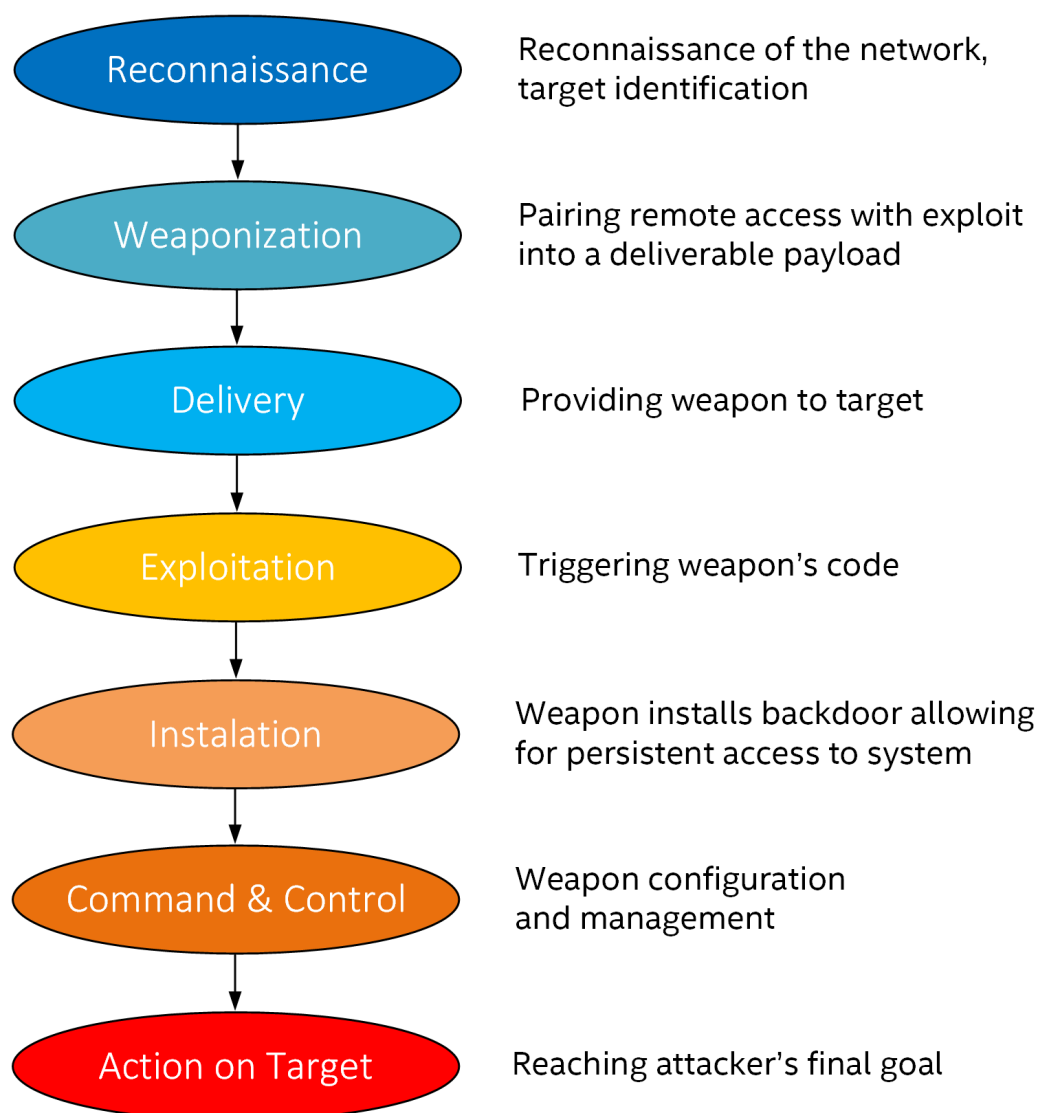


Figure 1. Intrusion Kill Chain Model proposed by Lockheed Martin Corporation [8].

1.2. Our Contribution

This paper presents a review of the current state of knowledge of MTD for network defense. We have surveyed state-of-the-art MTD techniques, focusing on how they work, what kind of attacks they protect against, whether they include threat models, if and how they were tested, as well as metrics used to evaluate the proposals. In contrast to already existing surveys in the field [18,20–24], we have used surveyed papers as a basis on which to discuss the current status of research in the MTD field in the context of real-life scenarios, such as applications to existing networks. This state-of-art review has been enriched with a discussion on the opportunities and benefits of adding potential MTD support in hardware, the need for a better understanding of security levels offered by MTD schemes, and improved metrics and testing scenarios.

1.3. Paper Structure

We first provide a background and introduce MTD-related terminology as well as types of attacks that MTD might protect against. Then we go deep into current trends in MTD literature, analyzing them by what, how, and when to move as well as used testbeds, proposed threat models, and metrics. Finally, we look into the application of MTD to existing network and identify some research directions that need to be addressed in future works in the MTD area.

2. Materials and Methods

2.1. Methods

This review is focused on MTD for networks. For the research work, we have searched the following databases: IEEE, Springer, Association for Computing Machinery (ACM), and Science Direct. To identify relevant articles, the following query was used:

- “moving target defense” and “network”.

We have decided that papers relating to the Internet of things (IoT), wireless and mobile networks, as well as vehicular use cases were outside the scope of this review, and as such they were ignored. We have further excluded papers, in which the proposed method was not explained extensively enough or not validated in any meaningful way. The remaining articles were then sorted by publication date, as we prioritized recent proposals.

2.2. Background

This section presents definitions and background for common key terms associated with MTD and a different existing taxonomies for the field. Furthermore, we introduce major types of attacks that MTD can protect against. Finally, we present two main technologies that allow for much easier MTD implementation.

2.2.1. Attack Surface

A system, from the security point of view, can be viewed as a set of vulnerable components. Some of them can be potentially accessed by an attacker, who may then try to exploit them. It is in the best interest of the defender to minimize that number of vulnerable components in the system, which creates a so-called attack surface. This can be achieved by installing security patches, closing unused ports, or complying with best security practices. Early work on MTD theory [13] describes the attack surface simply as a resource available to an attacker, like a system port or software. The authors of [25], based on earlier works [26,27], describe the attack surface as a subset of resources in the system that can be used to potentially launch an attack. They further propose an attack surface metric as a method by which to determine which system is more secure, by measuring the likelihood of a system being attacked. A number of works have been presented about possible approaches to minimizing the attack surface of a system. Papers [28,29] propose a graph-based, algorithmic approach to manipulating the attacker’s perspective of the system. Authors achieve this by manipulating the attacker’s probes in such a way that the external system view, a notion formally introduced in the paper, is maximum distance away from the internal view given some upper cost for the defender. The authors of [30] modeled interactions between attacker and defender as a stochastic game that allows one to determine the optimal way to shift the system’s attack surface.

Passive reconnaissance is the stealthy observation of the system for a specified period to detect a vulnerability (which is equivalent to attack surface detection). A static system attacker’s knowledge of the system, and its security vulnerabilities, increases over time. Figure 2 compares this case with three different MTD strategies. When the system is reconfigured periodically in a finite space, the observer’s level of knowledge grows slower. After each reconfiguration, the attacker’s knowledge of the current status of the system drops, but eventually overall knowledge rises. Increasing the space for reconfiguration, combined with pseudo-random timing, hardens the system even more. From the security point of view, ideal configuration uses asynchronous reconfiguration in an unlimited reconfiguration space. Unfortunately, due to technical limitations, this case is not feasible in practice.

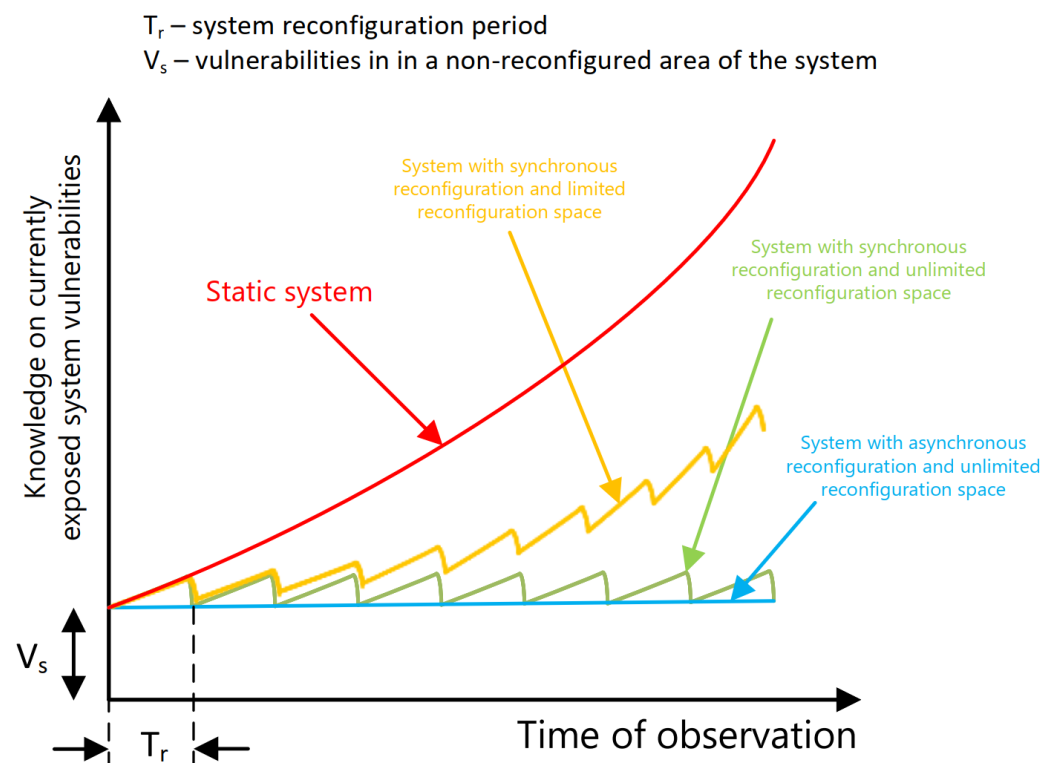


Figure 2. Knowledge of system vulnerabilities vs. time of observation for static system and system with reconfiguration.

2.2.2. MTD Techniques Taxonomy

There has been a significant amount of novel MTD techniques presented in recent years. As such, it was necessary to create a taxonomy to categorize this work. The paper [31] proposed to classify each technique by coverage, unpredictability, and timeliness. Similarly, ref. [13] proposed that an MTD method should consider three main issues: which piece of the system to move, space of movement of said piece, and the time for the movement to occur. This was further expanded upon in [22]. The authors proposed three easy-to-remember elements for each MTD technique based on the moving parameters—what, how, and when to move.

- What to move refers to the choice of moving parameters in the system. Each of them can be dynamically changed within a domain of allowed values. Such changes lead to a change in the system's attack surface, resulting in increased attack complexity. For this reason, each of the moving parameters must have a large enough parameter space to reduce the chance of being guessed by the attacker. Some examples of this category can be:
 - Network level
 - * Internet Protocol (IP) address
 - * Port
 - * Network topology
 - * Servers
 - Address space
 - Virtual machine (VM)
 - Operating system (OS)
 - Software version.
- How to move specifies the means by which to choose a new value and use it to replace a previous parameter. It is meant to increase the unpredictability of the system and confuse the attacker. Such techniques can include the use of:

- Randomness
- Game-theoretic approach
- Approach based on real-life observations.
- When to move defines the optimal time to change the moving parameters. It is crucial to choose the right schedule for that operation, such that performing it too rarely results in not enough security, whereas doing it too often might result in the loss of system performance.
 - Timer-based—a moving parameter is changed in fixed or varying time intervals
 - Event-based (reactive approach)—a change is done after a certain occurrence, such as after an intrusion-detection system (IDS) detects an intruder.

We have decided to use this taxonomy in this survey due to its simplicity and completeness.

A different approach was proposed in [32], in which the authors categorized methods to provide security to moving parameters in the MTD system into three types:

- Shuffling rearranges system components in various layers, like IPs, address spaces, or network topology.
- Diversity is about providing the same functionality by using different means. For example, different OS, compilers, or even programming languages.
- Redundancy ensures the existence of multiple replicas of the same component, for example, redundant nodes or paths.

These three methods can be used in combination with each other to provide higher levels of security.

Yet another taxonomy was introduced in a technical report on MTD [33]. Along with its second edition, released five years later [18], it remains among the most comprehensive surveys on MTD up to the present date.

- Dynamic Data are techniques that change the format or representation of data dynamically.
- Dynamic Software are techniques that change the application's code on the fly.
- Dynamic Runtime Environment are techniques that change the application environment. This group can be further subdivided into address space and instruction set randomization.
- Dynamic Platform are techniques that change the platform properties, like hardware components or OS version.
- Dynamic Networks are techniques that change network characteristics of the system, like topology or protocols used.

2.2.3. Attacks

In this section, we present network threats that might be prevented by using MTD solutions.

- Reconnaissance. Also called a scanning attack, reconnaissance is used by an attacker to obtain information about the target. These might include IP addresses, open ports, running services, OS version, and network topologies. Gathered data might then be used by the attacker to prepare before a real attack is launched, for example after discovering that target runs an unpatched application with a known vulnerability. Reconnaissance attacks can be divided into two types—passive and active. During a passive attack, an attacker does not interact with the target, which might include using public resources. This type of information gathering is also called open source intelligence (OSINT); for a more detailed description, refer to [34]. As for the active reconnaissance, the attacker is allowed to interact with the target, for example by directly scanning it. This might lead to gathering more data faster, but carries a significant risk of detection. There exists a huge number of dedicated tools to perform a target scan, some of the most popular examples of which are Nmap [35] for the network scan, Aircrack-ng [36] focused on WiFi network security, and Nessus [37] for vulnerability assessment. It is important to note here that these tools might be used

not only with malicious intent, but can also be run by the defender in order to provide important information in order to harden the network.

- Denial of Service (DoS). DoS is a type of attack that disrupts the normal functions of a device. A variant of DoS attacks called distributed denial of service (DDoS) is often used to bring down networks or servers. It involves using a large number of hosts, often a part of botnet, all working together to bring the target network down. In networks, this might involve sending a large amount of requests to a server, that overwhelms it and makes it unable to process them in real time. This so called 'flooding' might involve ICMP or ACK packets. A type of DDoS attack is crossfire attack that targets only few, selected links in the network masking itself very well, thus making it particularly hard to detect. More on that attack can be found in [38], and [39] talks about how various network topologies impact the detectability of crossfire attacks. Currently DDoS protection can be offered by vendors like Cloudflare [40] that provide solutions to detect the attack, then drop malicious traffic or reroute it, detect and block offending IPs etc.
- Zero-day. Sometimes also called 0-day, zero-day is a name for vulnerabilities before they are patched. In worst case scenarios, it can take months or even years before they are even detected, allowing the attacker to exploit a system, which is otherwise considered secure, in an unbothered manner. Report [41] indicates a huge rise in the number of exploited zero-days in 2012 compared to previous years. They were mostly used by state actors, targeted to spy on huge companies, but financially motivated attacks are also on a rise. The article [42] attempts to assess the security impact of unknown vulnerabilities on computer systems.
- Advanced persistent threat (APT) as described in [43] refers to an attack strategy by a bad actor with access to significant resources, both technical and financial, and high levels of expertise, allowing him to use multiple attack vectors to achieve his goal, which might be to extract information or to impede critical infrastructure or processes. Proposals for a multistage approach that such attacker must use to fulfill its objectives were introduced in [44,45]. The first stage of an APT is always reconnaissance; the attacker wants to understand as much about the target as possible. This could consist of technical information-gathering techniques, like port or service scanning, as well as the use of social engineering on the company employees to obtain necessary information. When this is finished, the attacker then attempts to gain a foothold in the attacked system, which can be done by means of using malware or zero-day vulnerabilities, as well as spear-phishing and a watering-hole attack. More data on APT campaigns is presented in [46], along with the entry methods used. After gaining access to the system, the attacker begins to slowly spread throughout it, which can take a long time, if one is to avoid detection. Finally, the attacker attempts to exfiltrate the data or impede the system. MTD might be perfect to protect against this type of attack, as shown in [4]. Multiple MTD techniques can be applied at each step of APT. However, from the defender's point of view, the best scenario would be to prevent the malicious actions early to stop it from ever gaining entry into the system, for example by significantly increasing the complexity of the reconnaissance stage. In [47], the authors talk in detail about the targeted nature of APTs, their characteristics and the motives behind them.

2.2.4. Potential MTD Implementation Techniques

This section describes the existing technologies that can be utilized to greatly reduce the complexity for implementing some of the MTD techniques.

Network function virtualization (NFV) is a network architecture that uses virtualization to perform traditionally hardware-based network functions, such as switches, load balancers, or firewalls. The idea was first proposed in [48], co-authored by representatives of many network operators. NFV, runs on VMs or containers on top of existing hardware, which can be instantiated on demand. Among the many benefits of this architecture,

ref. [49] lists cost reduction of capital investments, reduced energy consumption by networking hardware, and decreased time to market new services, as well as swift deployment of targeted solutions based on customer needs. The ETSI report [50] presents detailed user stories for NFV usage. More on NFVs can be found in surveys like [51], and [52] talks about the security aspects of that architecture. The NFV approach can be used as an effective technology for technique implementation providing an abstract layer for the host. This kind of solution results in the isolation of host software from network mechanisms, which makes migrations to the new architecture easier and cheaper to maintain.

Software-defined networking (SDN) [53] is a network paradigm that allows for dynamic and flexible network management. It decouples the control plane from the data plane, allowing for more efficient management. On the control plane resides one or more SDN controllers with direct control over the state of data plane elements. Consequently, all of the logical control power has been removed from those devices, which now only serve as simple forwarding devices, working according to rules programmed by controllers. The communication between control and data planes is done only by using interfaces, such as OpenFlow protocol [54]. Such separation allows one to achieve more flexibility and makes it easier to transform the network in the future. Some examples of SDN controllers are ONOS [55], OpenDaylight [56], NOX-MT [57], Beacon [58], and Ryu [59]. Due to their importance, SDN controllers are the potential point of failure that can bring down the entire network, and refs. [60–62] talk specifically about the problem of controller placement in SDN, whereas [63] provides a more general survey on fault management in SDN. Overall, there is a great amount of literature regarding SDN. Some examples of surveys discussing their architecture, applications and security are [64–67]. The isolated SDN control plane seems to be a natural candidate to consider in the context of dynamic network reconfiguration management mechanisms for MTD.

Overall, NFV and SDN seem like a good match for MTD architectures, providing dynamic network control and dynamic resource allocation, greatly reducing deployment cost and time by removing the need for deploying new network devices, and allowing for easier implementation of an MTD mechanism such as route or host mutation. One of the technologies that allows for the merging of NFV and SDN is OpenVSwitch [68], a high-performance software switch used in virtualized environments. Articles [69,70] describe integrating NFV and SDN in more depth. An example of MTD architecture utilizing both SDN and NFV is found in [71]. The authors proposed a framework, using route mutation to defend ISP networks against DDoS attacks and reconnaissance that was capable of network forensics. The proposed architecture utilizes a virtualized shadow network, each composed of multiple VMs, and deployed across the real network. With regard to incoming traffic, the SDN controller can flag them as potentially malicious and choose one of three scenarios. The first one is forwarding it on a random route across networks to a real host destination. The second one sends it to a shadow network, or to a virtualized shadow host. The last strategy sends the packet through both real and virtual networks, but it ends in a real destination host. Each of those strategies has a mutation probability, used to determine which of them will be applied, but that probability can be modified depending on past actions taken against a given traffic. For the legitimate traffic, the first strategy is applied every time.

3. Trends in MTD

We studied research papers introducing novel MTD architectures, especially focused on what, how, and when they move. We also noted the assumed threat model, what they protect against, and how were they tested and validated. Results of our study were combined and presented in Table 1. We found out that a relatively high number of articles were utilizing SDN, suggesting that it is a popular research direction. Out of the articles surveyed, most apply SDN to either perform route or address mutations inside networks.

3.1. What to Move

This section focuses on the most common MTD techniques found in the surveyed literature. We noticed that among the most popular techniques are IP, port, route, and host mutations, using decoys and proxy servers. There are also a number of studies utilizing MTD architecture to dynamically place conventional defense systems, like IDS. This part also introduces all of the surveyed papers, providing short and concise, one-paragraph descriptions for them.

Table 1. A taxonomy of surveyed MTD architectures, covering the what, how, and when aspects of the MTD strategy used, attack types they prevent, and whether they are SDN-based.

Article	MTD Type			Defense Against	SDN
	What	How	When		
[72], 2021	IP	Randomness	Event	Recon	✓
[73], 2021	Route	Randomness	Event	Crossfire	✓
[74], 2021	All	Randomness	Timer, Event	Recon	✓
[75], 2021	Proxy	Randomness	Timer	DDoS	
[76], 2021	Strategy	Game-theory	Event	Recon	✓
[77], 2020	VM	State	Event	DDoS	✓
[78], 2020	Proxies	Deep Q-learning	Event	DDoS	
[79], 2020	IP	Randomness	Timer, Event	Recon	✓
[80], 2019	IDS	Markov game	Event	APT	
[81], 2019	Various	Genetic algorithm	Event	Various threats	
[82], 2019	Topology	Randomness	Timer, Event	Recon	✓
[83], 2019	Route	State	Timer	Recon, APT	
[84], 2019	IP	Randomness	Event	Recon	✓
[85], 2018	IP	Randomness	Timer	Recon	✓
[86], 2018	Path, Host, IP	Not defined	Event	DoS	✓
[87], 2018	Countermeasure	Markov game	Event	APT	
[88], 2018	Port	Randomness	Event	DoS	✓
[89], 2018	IDS	Game theory	Event	Recon, DoS	
[90], 2018	IP	Randomness	Timer	Recon	✓
[91], 2017	Topology	Finding optimal shuffle	Timer, Event	Recon, DDoS	✓
[92], 2017	Hosts, ports	Randomness	Event	Recon	✓
[93], 2017	Path	Pre-shared key	Event	Recon	
[94], 2017	Fingerprint	Signaling game	Event	Recon	✓
[95], 2017	Domain name, IP	Randomness	Event	Recon	✓
[96], 2017	Network bandwidth	Game theory	Event	DoS	✓
[97], 2016	Topology	Randomness	Event	Recon	✓
[98], 2016	IP, decoy	Randomness	Timer	Recon	

Table 1. Cont.

Article	MTD Type			Defense Against	SDN
	What	How	When		
[99], 2016	Path	Randomness	Event	Recon	✓
[100], 2016	IP, Path	Randomness	Event	Recon	✓
[101], 2016	IP, path, hosts	Randomness	Event	Recon	✓
[102], 2016	VM	State	Event	DoS	✓
[103], 2016	Proxies	Randomness	Event	Recon, DoS	
[104], 2016	Path	State	Event	Crossfire	✓
[105], 2016	IP	Markov game	Event	Recon	
[106], 2016	VM, IP	Randomness	Timer, Event	Recon	
[107], 2016	IDS	Defined strategies	Event	Recon	
[108], 2015	IP, MAC	Randomness	Timer	Recon	✓
[109], 2015	IP	Randomness	Timer	Recon	
[110], 2015	Port, Address	Pre-shared key	Timer	Recon	
[111], 2015	IP	Game theory	Event	Recon	
[112], 2014	IP	Randomness	Event	Recon	
[113], 2014	VM	Randomness	Event	DDoS	
[114], 2014	VM	State	Timer	Recon	
[115], 2013	Proxies	Randomness	Event	DDoS	
[116], 2013	IP, Decoy	Randomness	Event	Recon	
[17], 2012	IP	Randomness	Timer	Recon	✓
[117], 2011	VM	Randomness	Event	Recon, DDoS	
[118], 2011	IP	Randomness	Timer	Recon	
[119], 2011	Software	Randomness	Timer, Event	Recon	

3.1.1. Address and Port Mutation

Among the surveyed literature, address mutation was the most common one. On the other hand, port mutation by itself was present in only one article, but was commonly applied together with IP mutation. We identified the two most popular approaches to implementing this MTD technique. One of them is to use short-lived addresses, which requires frequent domain name system (DNS) queries to obtain the current host IP. The second approach requires each host to have two IPs associated with it. One of them is permanent and the second one is virtual and frequently changed, unknown to the host. The first and last switch on the packet's way are the most important, as they are responsible for substituting real with virtual IPs or vice versa, making it transparent from the host point of view. This approach is used mostly in SDN networks, as the controller has an easy way to program the flow rules into switches.

In [17], the authors propose using SDNs to perform the host's IP address translation. Each of the hosts in the network is provided with a permanent (real) IP address and a short-lived (virtual) IP address. The entire process was designed to be transparent for end-hosts, which are only aware of their assigned real IPs. This technique requires two components to work: a switch capable of translating real IPs to virtual ones and a controller coordinating this process, keeping track of current real-to-virtual address mapping across the entire network.

Another SDN-based MTD technique is found in [108], utilizing DNS and network address translation (NAT) to achieve IP and medium access control (MAC) address transla-

tion. Each of the hosts is assigned a virtual IP with a very short time to live (TTL), which forces frequent resolution requests from a defender's DNS records that are constantly updated with new addresses. Each time such a request is detected, an SDN controller will update NAT with mapping rules that translate the virtual IPs and MACs to real ones. To achieve protection from insiders' threats, the systems' hardware and OS is within trusted computing base (TCB).

The authors of [118] utilize a vast address space on IPv6 to constantly rotate IPs of both clients and servers. The proposed technique performs address translation regardless of the current connection state, which increases the number of packet loss in the network, but does not drop or renegotiate sessions. This method allows two hosts to communicate over the Ethernet by encapsulating every frame as a user datagram protocol (UDP) datagram and adding new headers and such a packet could potentially be further encrypted. Both address obfuscation and optional tunnel encryption is done by using a symmetric key, which needs to be known to both parties beforehand; however the key establishment is not part of the scheme and has to be out-of-band.

In [109], an address mutation MTD technique is introduced that is fast and unpredictable, while also being adaptive to adversarial behaviour. When trying to connect to a host, the client must send a DNS query and will receive a short-lived destination IP address. A controller (or several controllers to improve scalability) manages the flow rules based on temporary IPs and assigns the address ranges to routers. To decrease the size of routing tables, the address ranges are usually within the same physical subnet, which comes at the cost of unpredictability.

The paper [85] proposes an SDN-based MTD solution that utilizes an IP mutation to thwart reconnaissance attacks. Each of the servers in the network has a set of short-lived, randomly assigned, virtual IPs associated with its real network address. The virtual IPs are dynamically updated every time period. An SDN controller is responsible for assigning those virtual IPs to hosts inside the network, installing appropriate flows into switches, and updating DNS queries for server host names.

The only article utilizing the only port mutation is [88], which proposed an SDN-based MTD technique by using port hopping as a countermeasure. The SDN controller is responsible for utilizing common vulnerability scoring system (CVSS) information coming from the vulnerability scanner and active protection from the IDS to identify the VMs with the highest security risk. If the threat level is above a certain threshold, the port hopping is applied to that host.

The paper [105] introduced a general Markov game framework for MTD analysis. This method can be applied to a broad range of different techniques and can measure the strength or effectiveness of each of them based on their specific parameters and general system configuration. The framework was then successfully applied for an IP-hopping MTD technique for both single and multi-target hiding.

The authors of [106] proposed a bio-inspired MTD technique by using SDN to protect distributed systems utilizing short-lived nodes that are used during computation, then discarded, but which can reappear later with changed properties. The algorithm is capable of moving nodes between hardware and hypervisors and performing IP hopping, as well as changing the characteristics of the node itself, like different OS. The proposed system also performs proactive node monitoring on deployed machines to detect anomalies in memory to discover potential ongoing attacks.

In [111], an MTD technique is proposed that utilizes IP hopping for real and decoy nodes, based on a game between defender and attacker. The assumed attacker can distinguish the real from the fake node by exploiting the timing differences in responses or protocol fingerprinting. Those two attack methods were modeled by using game theory and were further used to formulate a game wherein the attacker attempts to find the real node. This model is subsequently used to identify the optimal IP randomization scheme.

In [90], an SDN-based MTD method is introduced that utilizes frequent and fast IP hopping performed on every node in the path of the packet. The SDN controller task

is to provide each of the switches in a network with two random seeds, one of which is used for seeding the pseudo-random generator and the other for synchronization. There is also a third seed, called the IP seed, generated locally on each of the nodes and known to every other node in the network and is required to calculate the routing. Together with the pseudo-random generator seed, it is also used for generating new IP packets; after every hop both destination and source addresses are randomized. The last seed, the synchronization seed, is used to generate hash chains, which are precomputed and stored on every node in the network. The hashes are then used in reversed order to generate a specific IP address, which is used in an update packet that synchronizes all the nodes in the forwarding path.

The authors of [84] propose an SDN-based MTD method performing transparent IP mutation. When the packet reaches the first switch along its route, the SDN controller is responsible for assigning new, random network IPs for destination and source addresses. Every other switch along the packets' way sends a message to the controller, which then checks whether it is the last switch before the destination node. If not, then the packet is forwarded; otherwise the controller restores the original IPs in the packet.

The paper [120] improved upon [85] by utilizing multiple SDN controllers to be responsible for IP shuffling. This approach allows one to create a decentralized architecture, in which each of the controllers exchange information with the others and the workload can be split between the others in case of failure. As such, two communication channels are used—one between the controllers and the other between controllers and switches to program MTD mutations.

The MTD technique presented in [72] is based on IP hopping triggered by a timer or a detector, utilizing a lightweight convolutional neural network to detect threats. The architecture takes a multistep process to decide its strategy. It starts with packet sampling, collecting every probe packet, and optionally subsequent packets as well. These collected samples are then processed by removing all the information that is not related to scanning attacks and finally, the data is then passed to neural network, which can trigger address mutation if the packets are deemed malicious.

The paper [79] proposed an MTD architecture utilizing SDNs to create an IP randomization scheme where the mutation interval is different for every host in the network and can be adapted based on current needs. The interval time can be modified by the presence of an active session between two hosts, during which their virtual IPs will not be changed as long as it is open. This allows one to preserve stability and performance in the network. The system is also capable of monitoring traffic to discover the most active of the hosts, which are selected for more frequent address mutation.

The authors of [112] introduce an address-mutation scheme that allows one to dynamically change hosts' IPs depending on the location and time. Each of the hosts has a set of short-lived IPs that can be used to connect to other hosts in the network, such that one particular IP can be used only by the assigned host to connect to one particular host for a given period of time, which can be accessed via DNS request; after this time frame all the IPs are mutated. The switches are responsible for substituting real IPs of the sender with a short-lived one appropriate for the particular connection. The IP assignment and programming flow rules on switches is handled by a controller.

In [110], an address- and port mutation-based MTD technique is proposed by using source and service identity as well as time to achieve high unpredictability rates. At the beginning of each time interval, new virtual addresses and ports are assigned to hosts from the available pool by using the secret key shared between authenticated clients and server. The real-to-virtual address translation, as well as the interception and replacement IPs in DNS queries, is performed by a port- and address-hopping gateway.

In [93], an MTD technique using port hopping, capable of achieving per packet synchronization without needing additional communication channels is introduced. This is achieved by calculating a keyed-hash message authentication code (HMAC) of the message, which is used not only to verify data integrity on the receiver side, but also as an input to

encode/decode algorithms. It is then able to calculate virtual addresses and ports to be assigned to the message and to reverse this process to recover the real network data. Such an approach, however, requires the use of a shared session key known by both parties.

3.1.2. Route Mutation

Another popular MTD method present in multiple articles is route mutation, which is capable of assigning a random path between hosts per traffic or even per packet. This is significantly easier to implement in SDNs, as the defender is not limited to existing, physical links between network devices. This technique seems especially effective against DDoS, including crossfire attacks, as it disperses network traffic among all possible routes in the network by design, thus greatly increasing the amount of traffic required to bring down the network.

In [91], a technique that protects the system against DoS and reconnaissance attacks by dynamically reconfiguring the network topology by changing the connections between nodes in network using SDNs is proposed. This mechanism can work in an active mode that periodically reconfigures the topology, or a reactive one, that selects the best configuration against the detected attack. Due to the used heuristic algorithm, a near-optimal solution for large networks can be achieved.

The article [99] proposed an MTD SDN-based technique relying on path hopping to mask traffic in the network. This method is capable of not only dispersing traffic over multiple paths, but also scrambles IP addresses and ports in the process. An SDN controller is responsible for guiding the packet to its destination by installing proper flow rules on every OpenFlow switch along its path and providing them with new virtual source and destination addresses. When a switch gets a new packet and positively matches it against its flow rules, it first changes the address and port of both source and destination and then sends it to the next hop. Such an approach makes it impossible for the intercepted packets to be matched against either the sender or the receiver.

The authors of [104] proposed an SDN-based MTD technique for mitigating the impact of crossfire attacks by using path mutations. The proposed approach consists of a two-stage defense managed by SDN controllers—stage one is obfuscating the link map construction for the attacker during the reconnaissance phase and the second one is detecting and mitigating the attack itself. The architecture is capable of reorganizing the routes in such a way that the congested connections are avoided while avoiding any significant disruptions to network services.

The paper [83] proposed an MTD framework for spatiotemporal route mutation by using a stochastic optimization model to provide network security. The system is supervised by a controller that disperses the traffic among nodes taking into account parameters like link capacity, predetermined network and user requirements, and node stability, but also overall user quality of experience (QoE). After every time period, new routes are programmed into nodes by the controller.

A number of MTD architectures utilize route mutation as a means to reroute the traffic into a shadow network, which is a preferably virtualized, honeypot-like environment, often mimicking real infrastructure used to lure attackers in order to provide them with forged signatures and obtain their fingerprints. This is often paired with more conventional defense methods, capable of separating malicious traffic from benign traffic, and rerouting it into the shadow network. One example of such a method is found in [71].

In [97] an SDN-based method to thwart reconnaissance attacks by using a shadow network to forge scan responses is presented. The system is composed of three main elements: a scan sensor, reflector, and shadow network. A scan detector is responsible for detecting scan traffic and notifying the reflector, which is a network device capable of redirecting the traffic flow. There could be a multiple of those elements in a system, with parts of a protected network located behind each of them. When scan traffic is detected, it is redirected to a shadow network, which then generated forged scan responses for the attacker.

The paper [73] proposed MTD architectures by utilizing SDN and intent-based networking, a methodology providing dynamic network management to prevent crossfire attacks. Every packet from SDN switches is forwarded to an SDN controller which decides whether it is malicious and can act in one of two ways to route the traffic into a shadow network, effectively preventing the attacker from successfully executing the attack. The first approach is to change the destination address of the malicious packet and the second one modifies the destination port in the packet.

The author of [82] proposed an SDN-based MTD topology mutation technique that consists of four main components: virtualized network (VN), network topology management (NTM), network monitoring (NM) and topology mutation control (TMC). VN represents the physical network components as well as the software-defined ones. NTM is comprised of specialized submanagers that control and orchestrate the life cycle of virtual resources in the network function virtualization infrastructure domain and virtual network function instances. NM is responsible for gathering network information from various security elements throughout the network. Finally, TMC decides on a new network topology. This change can occur periodically or after a trigger from NM and a new network state can be chosen from a set of predetermined configurations or be generated as a response to a threat.

In [86], a technique is proposed that utilizes SDN and MTD to defend against DDoS threats in high-speed networks, such as Internet service provider (ISP) networks. This method requires multiple ISPs to create a collaborative environment in which each of them monitors the data flow and exchange security events. The border routers, using border gateway protocol (BGP), with multiple backup routers are responsible for changing the shape of the network. Additionally, IP hopping and host-level MTD is responsible for creating honeypots to trap the attacker.

3.1.3. Host Mutation

Host mutation allows for a rapid, often spatiotemporal VM swap. Affected VMs can be identical and instantly replicated in different locations, possibly with different addresses. Otherwise, each can be unique, which can be achieved by differentiating a software stack on every VM or through other means while preserving the same functionality. The swapped VM can then reappear at a later time. Similarly to route mutation, this method proves very effective against DDoS attacks.

The authors of [117] introduce a technique that uses a set of VM-based servers, each with unique software, to diversify the attack surface and increase unpredictability. The VMs can be rotated on a fixed or event-driven schedule, and only a subset of them is online at a given time. All the client requests of the server go through a load balancer that distributes them between VMs that are currently available. The VMs are connected with a control station and periodically send status data, which can be used to trigger a state change.

The paper [114] presents an MTD technique based on VM migration by using a probabilistic model. By using the assumption that the attacker can target any of the active VMs, the longer they stay active, the more likely it is they will be attacked. This is taken into account when calculating the risk factor to decide whether to migrate a VM, and after each time period any of the active VMs can decide to replicate onto one of its replacements based on that information. The appropriate threat level is also calculated for each of the replacement locations that the VM might move to and the one with the lowest risk factor is chosen.

In [102], an SDN-based MTD technique is proposed that utilizes VM mutation based on attack probability to minimize cloud management overheads. A new VM location is chosen by SDN controller based on the computation and storage capacity, network bandwidth, and a VM reputation, derived from its history of cyberattacks. The migration process is triggered by a DoS detection by an IDS and when completed, all the clients except the attacker are rerouted to the new VM.

Similarly, ref. [77] introduced an MTD technique that uses SDN-enabled cloud infrastructure to perform VM migration against DDoS attacks. The architecture allows for a proactive approach and a reactive one, after an attack has been discovered. The authors aim to minimize the migration frequency and to pick the ideal new location with the lowest attack probability to drive the defender cost down. As part of the reactive defense approach, a honeypot node is used to trick the attacker into continuing the attack in an attempt to gain more data about it.

A subset of articles utilizing the host mutation apply this technique to proxy servers, placed between clients and the actual application server. This provides means of discovering an attacker, by shuffling the client to a proxy assignment after each detected attack. Clients from the affected server would be then relocated among other available ones, with a central server keeping track of past assignments. This technique allows one to flag potentially malicious users, and given they repeat their attacks, eventually discern them.

In [113], an MTD technique is introduced to thwart DDoS attacks by using server relocation and shuffling client assignments. When a server detects it is under attack, it is quickly replicated at different network environments, with a subset of original clients assigned to each of the new instances. Because a determined attacker might be able to pose as one of the clients and launch a new attack on the newly replicated server, a coordination server was introduced to keep track of the defense system. The process of replicating servers and shuffling clients might be performed multiple times, until the attacker is detected and isolated. A greedy algorithm was proposed to speed up this process.

The paper [103] proposed an MTD technique to reduce the impact of DDoS attacks by using proxies and client shuffling, while also replacing the proxies to reduce the attacker's ability to collect information about them. A lookup server is responsible for authenticating and assigning the clients to proxies. To reduce the number of IPs harvestable by insider threats, the client-to-proxy assignment remains consistent between sessions. When a server is attacked, there are two new ones activated in its place, and its clients are split between them. This process can be repeated many times, until the insider attacker is revealed and isolated.

In [115] a novel MTD technique is introduced to protect against DDoS attacks by introducing a layer of proxies between the client and application server that work only for authenticated users. The defender has a large pool of such proxies out of which only some are active at a given time to reduce operation costs, and each of them has a concealed IP address. Connecting user-first authenticates by using proof-of-work on an authentication server, it then assigns a single proxy. When a proxy detects it is under attack, it informs the authentication server and subsequently shuts down; each user is then assigned to a newly activated node at a different network location.

The authors of [78] proposed an MTD technique by utilizing deep reinforcement learning to protect against DDoS attacks. Each user has a score and is randomly assigned to one of many reverse proxies, which are passing requests to application server and are the only way to communicate with it. When one of them gets attacked, each of the users currently assigned to it has its score lowered and is reassigned to a different reverse proxy. To increase the efficiency of isolating malicious users, the deep Q-learning algorithm has been applied to suggest the most optimal user shuffle.

In [75], an MTD architecture is introduced that adds a layer of proxy servers between the user and application server to reduce the impact of DDoS attacks. Each of the proxies act as a NAT, storing the address mappings between internal and external network. The system is managed by a controller, which uses a DNS to make sure that only a subset of those proxies are active and accepting connections at any given time by modifying the address returned by DNS query. Users are assigned to a proxy through the use of short-lived DNS entries. The controller keeps track of past and current users to proxy assignment and is capable of isolating users that are deemed malicious in a few shuffles.

3.1.4. Dynamic Resource Allocation via MTD

This MTD method is based on a very grounded assumption that the defender operates only a limited amount of countermeasure systems, like IDS and that their placement has a negative impact on network, by slowing it down or other means. The proposed techniques thus focus on providing the most optimal (yet still changing over time) placement of those defense systems in the protected network.

The authors of [80] used the attack graph to formulate a general-sum Markov game to provide the optimal security resource allocation for cloud network security. Authors utilize both CVSS and the architect's quantification of how the placement of the security systems impacts performance to help come up with long-term strategies against multistage attacks, such as APTs.

The paper [87] modeled the interactions between attacker and defender as a dynamic game of perfect information by using a zero-sum Markov game on attack graphs assuming a multistage attack scenario. This method identifies the best tactic for the defender, forcing the attacker to use strategies that are not optimal, thus increasing the cost. The optimal tactic selection utilizes CVSS scores as a reward metric to identify the critical nodes and apply countermeasures when necessary.

The authors of [89] describe a problem of placing a limited number of IDS in a large network by using a Stackelberg game between the network administrator and the attacker to find the optimal configuration. The architecture uses a security scanner to provide CVSS scores for detected vulnerabilities together with the assumed value of the node to compute the optimal solution to the problem which the network or host-based IDS should activate or disable.

In [96], an SDN-based MTD approach is introduced based on a dynamic game modeling using a reward-and-punishment system. In this game, the network administrator is playing against other players, which are clients and some of whom have malicious intent. If an undesirable behaviour is detected, for example through IDS, the administrator is capable of reducing the network resources allocated to that player for the next time period.

The paper [107] presents an MTD technique that aims to thwart botnet operations by increasing the difficulty and likeness of detection by using dynamic placement of detection systems. The paper proposes two metrics: minimum detection probability and the attacker's uncertainty to dynamically change the detection points inside to a resource-constrained network. The first of the metrics provides the lower bound that a botnet will be detected by using the defender's strategy and the other describes the effort needed by the attacker to discover the detector placement.

3.1.5. Architectures Utilizing Multiple MTD Methods

For this section, we want to focus on architectures combining multiple MTD methods. This approach should provide the most versatile defense against the attacker.

The paper [95] proposed an SDN-based technique that randomizes domain names and mutates network addresses to prevent attackers from using DNS query lists and the time window attack to find application servers. This system requires an authentication server, as well as DHCP and DNS servers deployed by the defender. A legitimate client first needs to verify itself by sending proof-of-work schemes to the authentication server to obtain the current domain name of a requested server, which can then be used to query the DNS system for the current IP address associated with it.

In [92] an SDN-based MTD technique is introduced that changes the attack surface by host mutation and port obfuscation, as well as by introducing decoy servers. A new connection getting into the system first goes through an IDS, which can obfuscate the response if it is deemed malicious. Otherwise, it is checked against a hierarchy of hosts in the network, called a tower. Based on predetermined rules, the connection can then be obfuscated if necessary. Such an approach allows one to save network performance by obfuscating everything.

In [101], an SDN-based system is proposed, capable of deceiving reconnaissance attacks by providing every host within the network with different views. The proposed system is capable of performing address translation, path mutation, moving vulnerable hosts, and placing honeypots, as well as detecting malicious traffic flows. The system architecture consists of an SDN controller managing the generation of flow rules, a deception server manipulating the traffic, and a virtual network view generator, which outputs a description of the virtual network.

The paper [100] proposed an MTD technique by using SDNs to mutate both data paths and information about communicating sides. An SDN controller is responsible for randomizing the IPs in packets and selecting its path by using a weighted random path selection algorithm. By using this newly generated information, the controller sets the flow tabled in switches along the packets' way.

In [98], a network address randomization mechanism is introduced that ensures continuous service availability. To increase uncertainty for the attacker, a large number of decoy servers is launched alongside the real ones. The address translation is performed on both types of servers. The system consists of two main components: an authentication server, which verifies users and provides them with current, real application server IP, and a randomization controller, responsible for deploying decoys and translating network addresses.

The paper [116] proposed an MTD technique by utilizing decoy nodes and IP hopping. The interaction between attacker and defender has been modeled as an optimal stopping problem to minimize the tradeoff between security and performance when performing IP randomization. The time between IP randomization is decided based on the time that has passed since the last address shuffle and the expected attacker's speed of detecting false nodes and compared against cost for the clients to reconnect to the node, denoted by the number of connections.

The authors of [81] presented an MTD solution for defending the network against multiple different types of attack. The system is capable of recognizing the threat type and choosing the most appropriate response, among many possible mutations. The best approach is calculated from several parameters. Each of the attacks is represented by a set of indicators, which are used after quantization to calculate the total attack cost. Similarly, each of the defenses has a set of indicators used to calculate the total defense cost. The third parameter is defense efficiency describing the effectiveness of a given defense against a given threat. The last two parameters are defender/attacker payoff, reflecting the losses/gains during the attack. Given the described parameters, a genetic algorithm has been trained to find the optimal solution. Such an approach allows for maximizing the defense chance while minimizing the defense cost. Ref. [74] is an SDN-based MTD engine, providing nine varied MTD strategies that can be combined together to achieve a desirable defense effect against diverse threats, like crossfire attacks. Available strategies are network configuration, route, topology, and DNS service mutation, address shuffling, traffic reflection and manipulation, network diversification, and network elements migration. The engine is managed by a controller, which holds all the strategies for the network and can work both proactively and reactively, choosing the most suitable approach to the current threat. The decision is then passed on to multiple agents responsible for enforcing them at the node level.

3.1.6. Other

In [76], an SDN-based MTD technique is presented that uses a multiagent reinforcement-learning framework to model the attacker–defender interactions to maximize the rewards for the attacker. The authors used a Deep-Q learning algorithm to select the most appropriate action for a given state. Using a neural network-based solution provides more generality and provides decisions even during states, that were never seen before.

The paper [94] is an SDN-based MTD technique focused on thwarting fingerprinting attacks. Incoming traffic is monitored by a detection engine for patterns suggesting network scan attempts, when detected outgoing traffic is then modified to obscure real network

fingerprints. To help discern malicious from benign communications, the interactions between attacker and defender were modeled as a signaling game.

A biologically inspired MTD technique based on splitting architecture into smaller components to increase diversity was presented in [119]. The system is capable of dividing the software into smaller tasks, each of them generating a set of functionally identical, but behaviorally different variants that can be interchanged with each other. Depending on the current requirements, the system can shuffle the entire software stack to confuse the attacker.

3.2. How to Move

In the surveyed literature, the most common strategy on how to apply the MTD method was randomness. However, different approaches are also present in research papers, such as [93,110], which proposed an architecture, deriving new values from the pre-shared key between host and server. Articles [77,83,102,104,114] proposed calculating new values based on the current state of the system, possibly also taking past events into consideration. This could include network parameters, such as current link congestion, storage space, or even how many times a node had been under attack in the past. Another trend in the literature regarding strategies to choose new system state is the use of game theory to model the interactions between attacker and defender [76,80,87,89,94,96,105,111].

3.3. When to Move

We found three main approaches as to when to apply MTD strategies. The most basic of them is timer-based, when new values are calculated after a certain period of time, such as [17,109,110] or [75]. It is mostly used for short-lived virtual addresses or temporal client-proxy assignments. Many MTD architectures were found to rely instead on event-based triggers for state change, most often after detecting malicious traffic. This technique seems to be especially well-suited for various MTD architectures against DDoS attacks, as they are relatively easy to detect [78,103,104]. Other architectures were applying MTD strategies against reconnaissance attacks, after IDS has detected fingerprinting or scanning attempts [88,92,96]. We believe this could threaten the defender's system, when an attacker exploits new, unknown methods, or is very stealthy. The third approach is a combination of both previous ones, merging timer- and event-based methods, frequently applying MTD strategies on its own, but also depending on the current state of the network or host. MTD strategies can be applied when an active attack has been detected or modified to occur more often for hosts that are more active in the network. This approach can be found in [79,82,91,116] or [106]. Lastly, architectures utilizing per packet address or route mutation apply this MTD strategy to every single packet reaching the switch [73,93,99].

3.4. Testbeds

We categorized the surveyed literature based on the maturity level of the architecture and how was it implemented. We found that some of the papers never went past the simulation stage. These were most often performed by using Matlab [121] or Python [122] or other means. Some examples from the surveyed literature are [76,80,81,83,85,103,111,114–116,119]. The proposal in [107] was validated on the Rocketfuel [123] dataset containing network topologies, both from real ISPs and generated based on those real ones. However, even more of the articles surveyed have reached the emulation stage. The paper [93] was tested on a real network and [95] used a laboratory network to verify its architecture. OpenStack was used for [87] to make a small test environment out of three VMs. The authors of [106] used 10 physical machines with a gigabit Ethernet switch between them. Similarly, ref. [113] tested its proposal on a few servers deployed in the cloud. Out of the architectures utilizing SDN, by far the most popular choice to implement them was by using Mininet [124], although the choice of SDN controllers varied greatly. Among the most popular controllers used throughout the surveyed literature were POX [125], used by [94,101,109,112] and Ryu [59] used in [72,74,79,84]. Other choices of controllers include ONOS [55], which appeared in [73,86,120]. NOX [126] was used in [17,99], OpenDaylight [56]

in [96], and Floodlight [127] in [104]. In [108], the authors wrote one SDN controller in Python. The remaining part of surveyed MTD architectures were implemented by using various different available testbeds and cloud services. Jikecloud [128] was used by [78], PlanetLab [129] by [110], GENI [130] by [77,102], Science DMZ [131] by [88], CloudLab [132] by [90], and finally NS3 [133] has been used by [75].

3.5. Threat Model

In this section, we look at the assumed threat model in surveyed papers. Most of the articles had a section dedicated to that topic. For almost all of them, the attacker was located outside of the network and was launching some expected type of attack. The only one out of the surveyed papers to explicitly state that it does not deal with insider attacks was [98]. Threat models for this paper also assume that attackers cannot be in the same subnet as the server and there must exist a way to exchange secret keys between legitimate users and the server. The paper [108] stated that insider threats are not a problem if a defender launches the MTD architecture out of TCB, which would prevent administrator-level attacks and that user-level attacks are not a threat to the system. The article [101] protects against internal attacks located in an unknown node in the network, as long as SDN controllers are not compromised. In [115], internal threats are defined as attackers, which were able to uncover some of the secret proxies by compromising legitimate clients. Both external and insider attackers that have placed themselves in the network are considered in [82]. Similarly, external attackers and ones that have managed to infiltrate the network are acknowledged in [90], but the paper goes one step further and assumes they are dynamic and adaptable, and capable of learning and improving depending on the defensive actions taken.

3.6. Metrics

This section describes various metrics utilized by the authors of surveyed papers to evaluate their proposed MTD techniques. There are currently no defined standard metrics with which those proposals can be tested. This forces the authors to come up with their own measures as they see fit. This in turn makes it much harder to compare MTD papers to one another. We've looked at the MTD metrics used to evaluate proposed techniques and attempted to group them together under common categories.

Attack success probability [17,72,73,77,83,85,92,93,100,102,103,109,111,116,117,120]: This metric describes a chance for the attack to be successfully performed. We've decided to make this an umbrella term for both the attacker and defender perspective, as they can be viewed as the two sides of the same coin. In [83], the authors measure the number of potential victims and success rate of eavesdropping and DDoS attacks independently. The paper [100] considers the possibility of an attacker obtaining complete communication data. The scan success rate is measured in [93]. In [72], nodes are measured for their minimal survival rate and the average lifetime of targets. The paper [109] calculates ratios of infected/uninfected hosts out of a total host pool. The articles [111,116] measure the probability of identifying real nodes from decoys by the attacker. The technique proposed in [17] is evaluated by using the number of infected hosts in a network. In [92], the percentage of information disclosed to the attacker is used as a measure. The paper [103] measures the decrease in the number of innocent victims. Ref. [75] uses the percentage of saved benign clients, and the number of shuffles to save benign clients is used in [113]. The ratio of missed hosts during a reconnaissance attack or the infected hosts in a worm attack is used in [95]. This is also true of [101], in which the authors named these metrics host detection/infection rate depending on the attack type. In [114], the authors use asset survival rate as a measure, which describes a percentage of defenders' assets that have not yet been captured by the attacker.

Performance overhead [17,73,74,77,83,90,91,93,95,98–104,109,110,115,118]: This metric describes additional overhead introduced into the system after applying the proposed MTD technique.

- Network based performance metrics:

- Packet overhead [118]: Measures the increase of packet size after applying MTD techniques.
- Packet Loss [77,91,118]: This metric shows how many packets have been lost in the network when using proposed techniques.
- Latency (delay) [73,74,90,91,95,98–101,104,109,110,115,118]: Measures the change in time it takes for a packet to reach its destination.
- Routing overhead [17,95,99,101,109]: This metric describes the overhead introduced by changes to network routing rules introduced by the applied MTD technique. In [109], the authors measure routing table size and convergence time. The number of flow rules in switches generated by the MTD technique is used as a metric in [17,95,101]. The solving time for route selection is measured in [99].
- Throughput [74,93,98,102,104,110,115]: Shows change in amount of data flowing in the network.
- Address space overhead [17,109]: Refers to the required number of network addresses needed for mutation space.
- DNS [109]: Measures the change in number of queries to DNS.
- Traffic dispersion [83]: Measures per-flow and per-node traffic dispersion.
- Hop count [73]: This metric shows the change in hop count between two hosts due to changed paths.
- Computational overhead [73,77,90,100,109]: Measures the amount of additional calculations required by the MTD technique. In [73,77,100], it is tracked as a CPU utilization. The authors of [90] calculate the number of signatures that have to be precomputed in their proposal. The paper [109] measures the time required to find a solution by a satisfiability module theories solver required to plan the MTD mutations.
- Proxy count [103]: This metric calculates the number of proxies required to isolate 90% of innocent users from attackers.

Address Entropy [118]: This metric shows how much harder the hosts using MTD address shuffling are to find in the network, compared to static hosts.

Quality of experience [102]: This measures the subjective human satisfaction with service before attack, during which and when it is repelled it uses the proposed MTD technique.

Number of dropped connections [111,116]: This shows the number of connections that are dropped during each network randomization and that need to be later restored.

Defense payoff [81]: This metric reflects losses avoided by the attacker and is split into two submetrics called attacker and defender cost. The defense payoff is then a total cost of all attacks and reduces the total cost to the defender.

Three metrics have been introduced in [112] to measure the effectiveness of MTD techniques called deterrence, deception, and detectability. Each of them quantify one of the mitigation vectors used.

- Deterrence—Quantifies the cost incurred by the attacker expressed as a time required to complete an attack compared with legacy network.
- Deception—Quantifies the ratio of missed targets, or the percentage of resources saved, due to applied deception techniques.
- Detectability—Measures the ratio of illegitimate actions committed during an attack, like probing nonexistent destinations in an MTD network compared with a legacy network.

The authors of [108] use a set of MTD metrics first proposed in [134]:

- Unpredictability—Requires that defended assets must be moved in a manner that seems random to clients without proper authorization.
- Vastness—Guarantees that the destination space must be large enough so that it is infeasible by an attacker to find its target by means of an exhaustive search.
- Periodicity—Ensures that the defended assets are moved frequently enough so that any data gathered by an attacker is quickly expired.

- Uniqueness—Guarantees that the system can authorize each client, in a way that cannot be shared with any other client, after it meets preconditions set by that system.
- Revocability—Provides a way for the system to revoke or expire once granted authorization without causing disruption to other clients.
- Availability—Guarantees that once a client is authorized, it can successfully reach its target. This also requires the MTD technique to not introduce any new denial-of-service vulnerabilities.
- Distinguishability—Allows the system to separate trustworthy clients from untrustworthy ones.

Another set of metrics has been proposed in [86]:

- Diversification—Requires the system to support multiple configuration choices.
- Adaptations—Ensures that the system supports both the the movement within the system that does not change the network graph's shape and size as well as the movement that does change it.
- Randomization—This metric covers the unpredictability of both movement and network configuration transformations.
- MTD Entropy—Measures the effectiveness of MTD defense solution.
- Ease of Deployment—Measures whether the system supports platform independence as well as protects against a large set of cyberattacks.
- Timelines—Ensures the system is capable of performing MTD movement within a given time period.
- Scalability—Measures the capability of the system to handle huge amounts of clients, IP prefixes, and matching rules.
- Wide-area load balancing—This metric describes the system's ability to disperse attack traffic over available resources.
- Cost consciousness—Measures the cost to deploy and run proposed MTD defense.

4. Discussion

4.1. Application to Existing Networks

In this section, we want to discuss how current state-of-the art MTD techniques integrate with existing network models. In Section 3.4 it can be observed that almost all of the surveyed MTD articles were in either simulation or emulation phase, whereas almost none was tested on a real scale network. Two exceptions are [93], which tested on two subnets inside a university campus network, and [95], which used a laboratory network comprised of 36 hosts split between three subnetworks to verify the architecture. Moreover, live network implementation of [118] has been presented in [135] on a network of 30,000 IPv6 hosts. This shows a potential gap in most of the state-of-the art MTD concepts which were never properly validated inside a realistic environment. Together with the huge variety in used test setups, this makes it exceptionally hard to compare different techniques against one another. This might factor in a slower widespread adoption of MTD techniques in large-scale enterprise networks.

Surveyed SDN-based MTD techniques were mainly based on full SDN networks. Although we were unable to find any source to show how popular this solution is inside huge enterprise networks, we believe most of them still rely on more traditional solutions. This adds another layer of complexity, uncertainty, and cost to the implementation of surveyed MTD techniques because the networks would have to be reworked severely in order to accommodate them. This might potentially outweigh the benefits the additional security MTD would provide. One solution to this would be to implement a hybrid SDN network, in which both traditional and SDN paradigms coexist. Hybrid SDN architectures were classified in [136]. One example of such a network is to allow an SDN controller to interact with legacy hardware and might even take full management over them. This might serve as a first step in organization before introducing SDN data plane in place of existing hardware. Another possible solution is to place the SDN nodes only on the edges of the network, passing the responsibility for all outside traffic onto the SDN controller.

Another notable architecture introduces SDNs by region, creating islands based on the same paradigm and connected via a gateway. Although this solution might allow one to introduce SDN-based MTD techniques into legacy networks, much work is required to assess how much of the claimed benefits those techniques would retain, as well as what would be the performance impact on those networks. More on existing approaches to hybrid SDNs can be found in [137].

4.2. Hardware-Accelerated MTD

This section discusses the potential applications of hardware acceleration to MTD techniques. We see this as a promising future direction for implementing MTD in realistic scenarios. We have identified two main ways hardware acceleration could fit in MTD.

- Improved introduction to established networks. With the help of specialized devices, MTD could be introduced simply and without major disruption to existing networks. Such hardware should require little configuration to work properly to reduce the chance of human error during installation and provide expected security levels out of the box.
- Enhancing MTD in networks already defended by MTD. In this scenario, the addition of hardware accelerators to offload MTD-related computations can improve both network performance and MTD defense. Such devices could take away additional operations needed to operate MTD from existing infrastructure, thus increasing throughput, latency, or other parameters of the network. Additionally, they might be added to improve defensive parameters of MTD, like more frequent parameter change, with no negative impact to network user experience.

Over the last several years, a significant development in the programmable device market has been observed [138]. This stems from the specific needs created by architects of telecommunications network infrastructure and data centers. Optimizing infrastructure for the performance of a given solution often requires the use of properly tuned algorithms, which is not possible when using components of a classical network infrastructure [139]. The development of programmable, high-performance system-on-a-chip (SoC) solutions provides technical possibilities to build highly programmable, and thus more flexible, devices. From the host OS perspective, programmable devices improve separation of infrastructure from tenant and offload infrastructure [140]. The numerous advantages of this approach have resulted in the creation of a new term in the field of technology—infrastructure processing unit (IPU)—which is a programmable network device that intelligently manages system-level infrastructure resources by securely accelerating those functions. Among the fastest growing segments of the IPU market are smart network interface cards (SmartNIC) [140] and programmable switches [141]. The authors believe that optimizing performance is not the only use case for programmable network devices. IPU hardware can also become a powerful platform for dedicated, tailor-made security solution implementation [142].

By analyzing papers surveyed in Section 3, we were able to identify few areas that show particular promise to be accelerated by using hardware solutions.

- Address and port mutation
- SDN-based MTD
- Route mutation
- Host migration
- Algorithms

Address and port mutation appear to be ideal candidates for hardware acceleration. The most promising examples in this category are [17,84,85,90], which implement real to virtual address and/or port translation in switches for every packet. This operation can be greatly accelerated by using programmable switches. After the controller writes mutation rules into them, this device allows one to perform lookup and modify the packet in near real time, without much impact to overall latency in the network. Similarly, as most of

the surveyed papers are utilizing SDN, the use of SDN-enabled hardware switches would provide much better performance than any software solutions. Such solutions merge the best of both worlds, providing high-speed, low-latency data throughput, while also separating the data plane from the control plane.

A very promising improvement idea for higher speed MTD schemes might be to implement algorithms that are part of control logic in hardware circuits. One solution might be to use application-specific integration circuits (ASIC), which provide much faster computation times than software implementation, but it comes at longer design times and higher implementation costs. ASICs are also much less flexible, as each change in logic requires a redesign of the circuit. A superior solution for this case might be field-programmable gate array (FPGA), which allows for quick reprogramming in the field by using hardware design languages (HDL) to implement logic. Although FPGAs don't offer as high performance as ASICs, they provide much greater flexibility when it comes to development. To get the most of an implemented MTD solution, one should use both types of devices. FPGA might be ideal for offloading proprietary MTD control logic, while off-the-shelf ASIC-based hardware accelerators can greatly enhance the performance of a more generic type of utilized algorithm. As many of the surveyed articles use randomness to obtain a new system state, specialized hardware could benefit them by providing a hardware random number generator (HRNG), increasing unpredictability of the data while working at a faster rate than software solutions. Adequate hardware accelerators can also be applied for increased computational rates of cryptographic algorithms.

4.3. Security of MTD Techniques

For the next topic of the discussion, we want to take a look at the security benefits of MTD. Although many articles present impressive results in simulated scenarios, it is unclear how these solutions would behave in real-life applications. We've identified several areas in the field that might help researchers understand how MTD affects network security, but are currently not explored enough.

- Lack of clearly defined and realistic threat model: In Section 3.5, we have presented our findings on threat models defined in surveyed papers. We found that in many cases, it might be too simplistic, and thus not realistic enough. These models often boil down to assuming that an attacker is located outside the network and launches some specific kind of attack. A common pitfall we've noticed across the literature was assuming the attacker actively engages with the network, most commonly by probing IP addresses. In real networks, this kind of behaviour would likely be quickly picked up by existing sensors and such an incident would alert defender's security team.
- Protection against insider threats: Although closely tied to the previous point, insider threats deserve a mention on their own. Cybersecurity and infrastructure security agency defines insider threats as potential for people with elevated access and knowledge in the organization to harm it [143]. According to a report by IBM, malicious insiders were responsible for 5 to 29% of the attacks, depending on the industry [144]. Although this is a serious threat to computer systems, most MTD papers never consider their impact on the security of the scheme. It is unclear how damage can be caused by a malicious employee who leaks the real IP of a machine protected by constant address mutation or MTD algorithm details.
- Lack of realistic testing scenarios: In Section 3.4, we have presented test methods that were used throughout surveyed papers. As further stated in Section 4.1, not many of those proposals were actually tested on real hardware. Because of that, it is almost impossible to assess the impact that proposed MTD techniques might have on the availability of network resources to users. Particularly vulnerable might be protocols that require establishing a session, as they might be negatively affected if mutations occur mid-session. Another issue might be the potentially detrimental impact of MTD on overall performance of the network, which might not be noticeable in a simplified, simulation-based environment.

- Lack of understanding how security levels behave after MTD have been enabled for a long periods of time: Figure 2 shows how attackers' knowledge of a system, which periodically mutates in a limited reconfiguration space, slowly increases over time. Analyzing the surveyed articles, it is not clear enough to conclude for how long these systems would actually work against a determined attacker. State actors might have the funding necessary to observe the system for months or years, slowly gathering intelligence about the network. More research is required to help understand if proposed techniques have a large enough mutation space to effectively protect against this kind of threat over prolonged periods of time.
- Lack of consideration of alternative attack vectors: The next security gap we have identified is the lack of flexibility of surveyed MTD techniques. This proposal often protects against one particular type of attack, but because the aim of MTD is to overcome the attackers' asymmetric advantage over the defender, these proposals are just not enough. In the case of address mutation MTD schemes, if the attackers purely operate on IP addresses, these techniques might indeed protect the network. However, if the attacker tries to implement more sophisticated attack vectors, like packet analysis, it might overcome this defense completely. We propose that more research time needs to be spent on flexible MTD schemes, which are able to protect against a wide range of threats.

4.4. Metrics

In Section 3.6, different metrics proposed by authors throughout surveyed literature were presented. Most commonly, these could be grouped into one of two main categories: the performance-based category and the attack parameter category, which includes the chance of success, or cost, for both attackers and defenders. In general, performance-based metrics might be a great benchmark, showing how a scheme might impact network parameters. The same, however, does not apply to the second category, as they seem particularly difficult to calculate in real-life scenarios. Moreover, every proposed scheme is evaluated by using different metrics. It is therefore crucial that better benchmarks for assessing the security of MTD schemes are developed. Ideally, they should then be adopted by future authors in the field. The benefit of this can be twofold—different schemes could be easily compared against one another, and engineers could use them to pick one that best suits their needs.

4.5. Research Directions

After reviewing a large number of papers related to MTD, it is apparent that extra work is required in different areas in this field. Based on findings from surveyed literature, presented below is the list of topics that authors of this paper believe are important to be researched further.

- Better metrics need to be developed—As shown in Section 3.6, many of the reviewed papers use some performance-based metrics which provide a solid ground on which the proposed technique can be evaluated. However, there is an apparent lack of metrics that provide good understanding of the security level those proposals provide to the network. Although many of the articles use the attack success probability metric, there is a huge variance in both the attacker and how this metric is calculated. Moreover, the proposed attacker may often not be a good representation of a threat to the network in real-world applications. As such, more work needs to be done to provide a set of common and universal metrics that can be applied to any proposed MTD techniques so that they can be easily compared against each other.
- Application of MTD to existing networks—As discussed in Section 4.1, currently there is little work done to assess the applicability of MTD to existing networks. Of the surveyed papers, the majority were tested on a simulator or on small-scale implementations in local networks. There needs to be more research of applications of these MTD techniques in large-scale corporate networks, especially in aspects like the

initial costs required in terms of both money and time spent reconfiguring the network, the decrease of general network performance, or the impact to network stability.

- **Hardware-accelerated MTD**—Closely tied to the previous point, we see great potential in the use of hardware-accelerated MTD to protect the network. Although such solutions don't necessarily increase security on their own, they provide significant performance improvement over typical computational devices, while being often more powerfully effective and requiring less management. One example of such an accelerator is SmartNIC, which is a type of a programmable device, based on an ASIC or FPGA [145]. These are already utilized in data centers for accelerating SDNs [146] or security-related tasks like DDoS protection [147,148].
- **Realistic testbeds**—During our research, we identified the need for better testbeds that allow one to simulate MTD in conditions that are as close to real as is feasible. This is especially important when it comes to simulating realistic attackers to measure the effectiveness of a given MTD technique. Although there is a significant amount of testbeds that allow one to simulate even large networks (especially for SDNs) and that offer a great deal of configuration possibilities, replicating realistic traffic on them is much harder. A trivial solution for this problem is to replay the saved packet traffic from files, but this comes with issues of its own. Perhaps the best source of them might be the save the packets from the network that we are going to implement the MTD in, but this might not always be possible. Otherwise one could use many of the traffic files shared for free on the Internet. No matter the source of the file, they often require a lot of storage space that further needs to be multiplied by the number of hosts on the network if one wants different traffic from each of the machines. On top of that, extra work is required to replay packets from protocols that establish a session, like transmission control protocol (TCP). This shows the need to create an easy way to configure a testbed for MTD validation with capabilities of a packet generator that is able to produce different types of traffic.
- **Economics of MTD**—Another poorly researched aspect of MTD is the overall economics of this technique. More work needs to be done to understand the cost of running the MTD technique across its lifetime, in terms of initial implementation costs, the work and hardware required, and the running costs of this solution. On top of that, closely tied to the previous two research directions, more work is necessary to help define the long-term financial benefits offered by enhanced protection by MTD compared with unprotected network.
- **Application of MTD to kill-chain phases other than reconnaissance**—Almost all of the surveyed papers were focused on disrupting an attacker's reconnaissance actions. An interesting research direction might be to assess viability of applying MTD to later phases of the Intrusion Kill Chain Model presented in Figure 1. The potential application of MTD might be an "Action on Objective" step, which would aim to prevent attackers from exfiltrating the stolen data.
- **Hybrid MTD**—The last research direction that we identified is MTD utilizing multiple strategies to protect against a wide range of attackers. During our survey, we identified only individual papers utilizing this technique, but we believe it has great potential against real-life threats. However, more work needs to be done to identify when and how certain strategies need to be applied to optimize the protection and minimize the impact to the network.

5. Conclusions

Moving target defense shows the potential to completely change the approach to network security. This security paradigm has inspired a large number of research publications in various stages of maturity—from theoretical proposals to implemented techniques. The first part of this paper was aimed at explaining the concepts behind the MTD paradigm. We discussed attack surface, examples of MTD taxonomy, and attack types that MTD might help defend against. We also introduced SDN and NFV as two promising network

architectures that can be utilized for implementing various MTD techniques. In the second part of the article, we have surveyed existing literature to identify current trends, as well as their advantages and disadvantages in order to provide guidance for further research work in the area. We focused on what, when, and how to move, as well as the used testbeds, threat models, and proposed metrics. During our research, we have discussed several shortcomings of currently available MTD schemes and identified future research directions in the field in the last part of this survey.

In general, we believe MTD as an area lacks maturity. There are no standards and no universal metrics that can be used to compare different techniques. We believe this might discourage both engineers and company decision-makers from adopting this paradigm. If MTD is to ever become widespread, more research is required to show how to implement it in existing large-scale networks, with a special focus on performance and costs. Realistic proof of concepts are required to show that MTD schemes are viable and working as expected, if they are to work alongside or even replace current defenses in network systems.

Author Contributions: Conceptualization, Ł.J., M.Z. and M.R.; methodology, Ł.J., M.Z. and M.R.; investigation, Ł.J., M.Z., and M.R.; writing—original draft preparation, Ł.J. and M.Z.; writing—review and editing, Ł.J., M.Z., and M.R.; visualization, M.Z.; supervision, M.R.; project administration, M.R. and M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Polish National Centre for Research and Development under Project CYBERSECIDENT/369234/I/NCBR/2017.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MTD	Moving Target Defense
NFV	Network Functions Virtualization
SDN	Software-Defined Networking
DOS	Denial of Service
DDoS	Distributed Denial of Service
OSINT	Open Source Intelligence
ACM	Association for Computing Machinery
IoT	Internet of Things
NAT	Network Address Translation
DNS	Domain Name System
MAC	Medium Access Control
IP	Internet Protocol
TTL	Time to Live
TCB	Trusted Computing Base
UDP	User Datagram Protocol
BGP	Border Gateway Protocol
CVSS	Common Vulnerability Scoring System
IDS	Intrusion Detection System
VM	Virtual Machine
OS	Operating System
ISP	Internet Service Provider
HMAC	Keyed-Hash Message Authentication Code
QoE	Quality of experience
NTM	Network Topology Management
VN	Virtualized Network
NM	Network Monitoring
TMC	Topology Mutation Control
APT	Advanced Persistent Threat

TCP	Transmission Control Protocol
HRNG	Hardware Random Number Generator
ASIC	Application Specific Integration Circuits
FPGA	Field-Programmable Gate Arrays
HDL	Hardware Design Languages
SoC	System-on-a-Chip
NIC	Network Interface Card
IPU	Infrastructure Processing Unit

References

- Okhravi, H.; Streilein, W.W.; Bauer, K.S. Moving Target Techniques: Leveraging Uncertainty for Cyber Defense. *Linc. Lab. J. Spec. Issue Cyber Secur.* **2016**, *22*, 100–109.
- How Long Does It Take to Detect a Cyber Attack? Available online: <https://www.itgovernanceusa.com/blog/how-long-does-it-take-to-detect-a-cyber-attack> (accessed on 4 January 2022).
- 98 Must-Know Data Breach Statistics for 2021. Available online: <https://www.varonis.com/blog/data-breach-statistics> (accessed on 4 January 2022).
- Khosravi-Farmad, M.; Ahmadian Ramaki, A.; Bafghi, A. Moving Target Defense Against Advanced Persistent Threats for Cybersecurity Enhancement. In Proceedings of the 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 25–26 October 2018; pp. 280–285. [CrossRef]
- Zerodium. Available online: <https://zerodium.com/> (accessed on 14 August 2021).
- Perlroth, N. The Untold History of America’s Zero-Day Market. Available online: <https://www.wired.com/story/untold-history-americas-zero-day-market/> (accessed on 14 August 2021).
- Greenberg, A. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Available online: <https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/> (accessed on 14 August 2021).
- Hutchins, E.; Cloppert, M.; Amin, R. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *Lead. Issues Inf. Warf. Secur. Res.* **2011**, *1*, 80.
- Pal, P.; Schantz, R.; Paulos, A.; Benyo, B. Managed Execution Environment as a Moving-Target Defense Infrastructure. *IEEE Secur. Priv.* **2014**, *12*, 51–59. [CrossRef]
- National Cyber Leap Year Summit 2009 Co-Chairs’ Report; Networking and Information Technology Research and Development: Please add more information 2009. Available online: https://www.nitrd.gov/nitrdgroups/images/b/bd/National_Cyber_Leap_Year_Summit_2009_CoChairs_Report.pdf (accessed on 1 September 2022)
- Jajodia, S.; Ghosh, A.; Swarup, V.; Wang, C.; Wang, X. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*; Springer: New York, NY, USA, 2011; Volume 54. [CrossRef]
- Jajodia, S.; Ghosh, A.; Subrahmanian, V.; Swarup, V.; Wang, C.; Wang, X. *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*; Springer: New York, NY, USA, 2013. [CrossRef]
- Zhuang, R.; Deloach, S.; Ou, X. Towards a Theory of Moving Target Defense. *Proc. ACM Conf. Comput. Commun. Secur.* **2014**, *2014*, 31–40. [CrossRef]
- Crosby, S.; Carvalho, M.; Kidwell, D. A layered approach to understanding network dependencies on moving target defense mechanisms. In Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop, Oak Ridge, TN, USA, 8–10 January 2013. [CrossRef]
- Carroll, T.; Crouse, M.; Fulp, E.; Berenhaut, K. Analysis of network address shuffling as a moving target defense. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 701–706. [CrossRef]
- Zhuang, R.; Deloach, S.; Ou, X. A model for analyzing the effect of moving target defenses on enterprise networks. In Proceedings of the 9th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 8–10 April 2014. [CrossRef]
- Jafarian, J.; Al-Shaer, E.; Duan, Q. OpenFlow random host mutation: Transparent moving target defense using software defined networking. In Proceedings of the HotSDN’12—Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012. [CrossRef]
- Ward, B.; Gomez, S.; Skowyra, R.; Bigelow, D.; Martin, J.; Landry, J.; Okhravi, H. *Survey of Cyber Moving Targets Second Edition*; Technical Report; Lincoln Laboratory, Massachusetts Institute of Technology: Lexington, MA, USA, 2018.
- Team, P. PaX Address Space Layout Randomization (ASLR). 2003. Available online: <https://pax.grsecurity.net/docs/aslr.txt> (accessed on 14 August 2021).
- Lei, C.; Zhang, H.Q.; Jinglei, T.; Zhang, Y.C.; Liu, X.H. Moving Target Defense Techniques: A Survey. *Secur. Commun. Netw.* **2018**, *2018*, 3759626. [CrossRef]
- Cho, J.H.; Sharma, D.; Alavizadeh, H.; Yoon, S.; Ben-Asher, N.; Moore, T.; Kim, D.; Lim, H.; Nelson, F. Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 709–745. [CrossRef]
- Cai, G.L.; Wang, B.S.; Hu, W.; Wang, T.Z. Moving target defense: State of the art and characteristics. *Front. Inf. Technol. Electron. Eng.* **2016**, *17*, 1122–1153. [CrossRef]

23. Zheng, J.; Siami Namin, A. A Survey on the Moving Target Defense Strategies: An Architectural Perspective. *J. Comput. Sci. Technol.* **2019**, *34*, 207–233. [[CrossRef](#)]
24. Sengupta, S.; Chowdhary, A.; Sabur, A.; Alshamrani, A.; Huang, D.; Kambhampati, S. A Survey of Moving Target Defenses for Network Security. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1909–1941. [[CrossRef](#)]
25. Manadhata, P.; Wing, J. An Attack Surface Metric. *Softw. Eng. IEEE Trans.* **2011**, *37*, 371–386. [[CrossRef](#)]
26. Manadhata, P.; Wing, J. *Measuring a System's Attack Surface*; Technical Report cmu- cs-04-102; School of Computer Science, Carnegie Mellon University : Pittsburgh, PA, USA, 2004.
27. Howard, M.; Pincus, J.; Wing, J. Measuring Relative Attack Surfaces. In *Computer Security in the 21st Century*; Springer: Boston, MA, USA, 2005; pp. 109–137. [[CrossRef](#)]
28. Albanese, M.; Battista, E.; Jajodia, S.; Casola, V. Manipulating the attacker's view of a system's attack surface. In Proceedings of the 2014 IEEE Conference on Communications and Network Security, San Francisco, CA, USA, 29–31 October 2014; pp. 472–480. [[CrossRef](#)]
29. Albanese, M.; Battista, E.; Jajodia, S. Deceiving Attackers by Creating a Virtual Attack Surface. In *Cyber Deception*; Springer: Cham, Switzerland, 2016; pp. 169–201. [[CrossRef](#)]
30. Manadhata, P. Game Theoretic Approaches to Attack Surface Shifting. In *Moving Target Defense II*; Springer: New York, NY, USA, 2013; pp. 1–13. [[CrossRef](#)]
31. Hobson, T.; Okhravi, H.; Bigelow, D.; Rudd, R.; Streilein, W. On the Challenges of Effective Movement. *Proc. ACM Conf. Comput. Commun. Secur.* **2014**, *2014*, 41–50. [[CrossRef](#)]
32. Hong, J.; Kim, D.S. Assessing the Effectiveness of Moving Target Defenses Using Security Models. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 163–177. [[CrossRef](#)]
33. Okhravi, H.; Rabe, M.; Mayberry, T.; Leonard, W.; Hobson, T.; Bigelow, D.; Streilein, W. *Survey of Cyber Moving Targets*; Technical Report; Lincoln Laboratory, Massachusetts Institute of Technology : Lexington, MA, USA, 2013.
34. Pastor-Galindo, J.; Nespoli, P.; Gomez Marmol, F.; Martinez Perez, G. The Not Yet Exploited Goldmine of OSINT: Opportunities, Open Challenges and Future Trends. *IEEE Access* **2020**, *8*, 10282–10304. [[CrossRef](#)]
35. Nmap: The Network Mapper. Available online: <https://nmap.org/> (accessed on 20 March 2022).
36. Aircrack-ng. Available online: <https://www.aircrack-ng.org/> (accessed on 20 March 2022).
37. Nessus: Vulnerability Assessment. Available online: <https://www.tenable.com/products/nessus> (accessed on 20 March 2022).
38. Kang, M.; Lee, S.B.; Gligor, V. The Crossfire Attack. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 127–141. [[CrossRef](#)]
39. Liaskos, C.; Ioannidis, S. Network Topology Effects on the Detectability of Crossfire Attacks. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1682–1695. [[CrossRef](#)]
40. Cloudflare: Comprehensive DDoS Protection. Available online: <https://www.cloudflare.com/ddos/> (accessed on 20 March 2022).
41. Zero Tolerance: More Zero-Days Exploited in 2021 Than Ever Before. Available online: <https://www.mandiant.com/resources/zero-days-exploited-2021> (accessed on 29 May 2022).
42. Wang, L.; Jajodia, S.; Singhal, A.; Cheng, P.; Noel, S. k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities. *Dependable Secur. Comput. IEEE Trans.* **2014**, *11*, 30–44. [[CrossRef](#)]
43. Ross, R. Managing Information Security Risk: Organization, Mission, and Information System View. 2011. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=908030 (accessed on 1 September 2022).
44. Chen, P.; Desmet, L.; Huygens, C. A Study on Advanced Persistent Threats. In *IFIP International Conference on Communications and Multimedia Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 63–72. [[CrossRef](#)]
45. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [[CrossRef](#)]
46. Ussath, M.; Jaeger, D.; Cheng, F.; Meinel, C. Advanced persistent threats: Behind the scenes. In Proceedings of the 2016 Annual Conference on Information Science and Systems (CISS), Princeton, NJ, USA, 16–18 March 2016; pp. 181–186. [[CrossRef](#)]
47. Sood, A.; Enbody, R. Targeted Cyber Attacks—A Superset of Advanced Persistent Threats. *IEEE Secur. Priv.* **2013**, *11*, 54–61. [[CrossRef](#)]
48. Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1. October 22–24, 2012 at the “SDN and OpenFlow World Congress”, Darmstadt-Germany. Available online: https://portal.etsi.org/nfv/nfv_white_paper.pdf (accessed on 1 September 2022).
49. Han, B.; Gopalakrishnan, V.; Ji, L.; Lee, S. Network Function Virtualization: Challenges and Opportunities for Innovations. *Commun. Mag. IEEE* **2015**, *53*, 90–97. [[CrossRef](#)]
50. Network Functions Virtualisation ETSI Industry Specification Group ETSI GR NFV 001 V1.3.1 Network Functions Virtualisation (NFV); Use Cases. 2021. Available online: https://www.etsi.org/deliver/etsi_gr/NFV/001_099/001/01.03.01_60/gr_NFV001v010301p.pdf (accessed on 7 October 2021).
51. Yi, B.; Wang, X.; Li, K.; Das, S.; Huang, M. A Comprehensive Survey of Network Function Virtualization. *Comput. Netw.* **2018**, *133*, 212–262. [[CrossRef](#)]
52. Alwakeel, A.; Alnaim, A.; Fernández, E. A Survey of Network Function Virtualization Security. In Proceedings of the Southeast-Con 2018, St. Petersburg, FL, USA, 19–22 April 2018. [[CrossRef](#)]

53. Kreutz, D.; Ramos, F.; Verissimo, P.; Esteve Rothenberg, C.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2014**, *103*, 14–76. [[CrossRef](#)]
54. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *Comput. Commun. Rev.* **2008**, *38*, 69–74. [[CrossRef](#)]
55. Berde, P.; Gerola, M.; Hart, J.; Higuchi, Y.; Kobayashi, M.; Koide, T.; Lantz, B.; O'Connor, B.; Radoslavov, P.; Snow, W.; et al. ONOS: Towards an open, distributed SDN OS. In Proceedings of the HotSDN 2014—Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking, Chicago, IL, USA, 22 August 2014. [[CrossRef](#)]
56. Medved, J.; Varga, R.; Tkacik, A.; Gray, K. OpenDaylight: Towards a Model-Driven SDN Controller architecture. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014 Sydney, NSW, Australia, 19 June 2014; pp. 1–6. [[CrossRef](#)]
57. Tootoonchian, A.; Gorbunov, S.; Ganjali, Y.; Casado, M.; Sherwood, R. On Controller Performance in Software-defined Networks. In Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, San Jose, CA, USA, 24 April 2012; p. 10.
58. Erickson, D. The Beacon OpenFlow Controller. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 13–18. [[CrossRef](#)]
59. Ryu, a Component-Based Software Defined Networking Framework. Available online: <https://ryu-sdn.org/> (accessed on 14 October 2021).
60. Zhang, Y.; Cui, L.; Wang, W.; Zhang, Y. A Survey on Software Defined Networking with Multiple Controllers. *J. Netw. Comput. Appl.* **2017**, *103*, 101–118. [[CrossRef](#)]
61. Kumari, A.; Sairam, A. A Survey of Controller Placement Problem in Software Defined Networks. *arXiv* **2019**, arXiv:1905.04649.
62. Wang, G.; Zhao, Y.; Huang, J.; Wang, W. The Controller Placement Problem in Software Defined Networking: A Survey. *IEEE Netw.* **2017**, *31*, 21–27. [[CrossRef](#)]
63. Yinbo, Y.; Li, X.; Leng, X.; Song, L.; Bu, K.; Chen, Y.; Yang, J.; Zhang, L.; Cheng, K.; Xiao, X. Fault Management in Software-Defined Networking: A Survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 349–392. [[CrossRef](#)]
64. Farhady, H.; Lee, H.; Nakao, A. Software-Defined Networking: A survey. *Comput. Netw.* **2015**, *81*, 79–95. [[CrossRef](#)]
65. Benzekki, K.; El Fergougui, A.; El Belrhiti El Alaoui, A. Software-defined networking (SDN): A survey. *Secur. Commun. Netw.* **2017**, *9*, 5803–5833. [[CrossRef](#)]
66. Nisar, K.; Welch, I.; Hassan, R.; Sodhro, A.; Pirbhulal, S. A Survey on the Architecture, Application, and Security of Software Defined Networking. *Internet Things* **2020**, *12*, 100289. [[CrossRef](#)]
67. Sahay, R.; Meng, W.; Jensen, C.D. The application of Software Defined Networking on securing computer networks: A survey. *J. Netw. Comput. Appl.* **2019**, *131*, 89–108. [[CrossRef](#)]
68. Pfaff, B.; Pettit, J.; Koponen, T.; Jackson, E.; Zhou, A.; Rajahalme, J.; Gross, J.; Wang, A.; Stringer, J.; Shelar, P.; et al. The Design and Implementation of Open vSwitch. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015; USENIX Association: Oakland, CA, USA, 2015; pp. 117–130.
69. Duan, Q.; Ansari, N.; Toy, M. Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw.* **2016**, *30*, 10–16. [[CrossRef](#)]
70. Li, Y.; Chen, M. Software-Defined Network Function Virtualization: A Survey. *IEEE Access* **2015**, *3*, 2542–2553. [[CrossRef](#)]
71. Saputro, N.; Aydeger, A.; Akkaya, K. A Moving Target Defense and Network Forensics Framework for ISP Networks using SDN and NFV. *Future Gener. Comput. Syst.* **2019**, *94*, 496–509. [[CrossRef](#)]
72. Xu, X.; Hu, H.; Liu, Y.; Zhang, H.; Chang, D. An Adaptive IP Hopping Approach for Moving Target Defense Using a Light-Weight CNN Detector. *Secur. Commun. Netw.* **2021**, *2021*, 8848473. [[CrossRef](#)]
73. Hyder, M.F.; Fatima, T. Towards Crossfire Distributed Denial of Service Attack Protection Using Intent-Based Moving Target Defense Over Software-Defined Networking. *IEEE Access* **2021**, *9*, 112792–112804. [[CrossRef](#)]
74. Wang, L. Shoal: A Network Level Moving Target Defense Engine with Software Defined Networking. *ICST Trans. Secur. Saf.* **2021**, *7*, 170011. [[CrossRef](#)]
75. Bandi, N.; Tajbakhsh, H.; Analoui, M. FastMove: Fast IP switching Moving Target Defense to mitigate DDOS Attacks. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Japan, 30 January–2 February 2021; pp. 1–7. [[CrossRef](#)]
76. Chowdhary, A.; Huang, D.; Sabur, A.; Vadnere, N.; Kang, M.; Montrose, B. SDN-based Moving Target Defense using Multi-agent Reinforcement Learning. In Proceedings of the first International Conference on Autonomous Intelligent Cyber defense Agents (AICA 2021), Paris, France, 15–16 March 2021.
77. Debroy, S.; Callyam, P.; Nguyen, M.; Neupane, R.; Mukherjee, B.; Eeralla, A.K.; Salah, K. Frequency-Minimal Utility-Maximal Moving Target Defense against DDoS in SDN-based Systems. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 890–903. [[CrossRef](#)]
78. Chai, X.; Wang, Y.; Yan, C.; Zhao, Y.; Chen, W.; Wang, X. DQ-MOTAG: Deep Reinforcement Learning-based Moving Target Defense Against DDoS Attacks. In Proceedings of the 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), Hong Kong, China, 27–30 July 2020; pp. 375–379. [[CrossRef](#)]
79. Gudla, C.; Sung, A. Moving Target Defense Discrete Host Address Mutation and Analysis in SDN. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 16–18 December 2020; pp. 55–61. [[CrossRef](#)]

80. Sengupta, S.; Chowdhary, A.; Huang, D.; Kambhampati, S. General Sum Markov Games for Strategic Detection of Advanced Persistent Threats Using Moving Target Defense in Cloud Networks. In Proceedings of the International Conference on Decision and Game Theory for Security, Stockholm, Sweden, 30 October–1 November 2019; pp. 492–512. [[CrossRef](#)]
81. Zhang, H.; Zheng, K.; Wang, X.; Luo, S.; Wu, B. Efficient Strategy Selection for Moving Target Defense Under Multiple Attacks. *IEEE Access* **2019**, *7*, 65982–65995. [[CrossRef](#)]
82. Rawski, M. Network Topology Mutation as Moving Target Defense for Corporate Networks. *Int. J. Electron. Telecommun.* **2019**, *65*, 571–577. [[CrossRef](#)]
83. Zhou, Z.; Xu, C.; Kuang, X.; Zhang, T.; Sun, L. An Efficient and Agile Spatio-Temporal Route Mutation Moving Target Defense Mechanism. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [[CrossRef](#)]
84. Macwan, S.; Lung, C.H. Investigation of Moving Target Defense Technique to Prevent Poisoning Attacks in SDN. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; pp. 178–183. [[CrossRef](#)]
85. Sharma, D.; Kim, D.; Yoon, S.; Lim, H.; Cho, J.H.; Moore, T. FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks. In Proceedings of the 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), New York, NY, USA, 1–3 August 2018; pp. 579–587. [[CrossRef](#)]
86. Steinberger, J.; Kuhnert, B.; Dietz, C.; Ball, L.; Sperotto, A.; Baier, H.; Pras, A.; Dreo, G. DDoS defense using MTD and SDN. In Proceedings of the NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–9. [[CrossRef](#)]
87. Chowdhary, A.; Sengupta, S.; Alshamrani, A.; Huang, D.; Sabur, A. Adaptive MTD Security using Markov Game Modeling. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019. [[CrossRef](#)]
88. Chowdhary, A.; Alshamrani, A.; Huang, D.; Liang, H. 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, Tempe, AZ, USA, 21 March 2018; pp. 43–48. [[CrossRef](#)]
89. Sengupta, S.; Chowdhary, A.; Huang, D.; Kambhampati, S. Moving Target Defense for the Placement of Intrusion Detection Systems in the Cloud. In Proceedings of the 9th International Conference, GameSec 2018, Seattle, WA, USA, 29–31 October 2018.
90. Chang, S.Y.; Park, Y.; Babu, B. Fast IP Hopping Randomization to Secure Hop-by-Hop Access in SDN. *IEEE Trans. Netw. Serv. Manag.* **2018**, *16*, 308–320. [[CrossRef](#)]
91. Hong, J.; Yoon, S.; Lim, H.; Kim, D.S. Optimal Network Reconfiguration for Software Defined Networks Using Shuffle-Based Online MTD. In Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, China, 26–29 September 2017; pp. 234–243. [[CrossRef](#)]
92. Wang, J.; Xiao, F.; Huang, J.; Zha, D.; Hu, H.; Zhan, H. CHAOS: An SDN-based Moving Target Defense System. *Secur. Commun. Netw.* **2017**, *2017*, 3659167. [[CrossRef](#)]
93. Luo, Y.b.; Wang, B.s.; Wang, X.F.; Zhang, B.f. A keyed-hashing based self-synchronization mechanism for port address hopping communication. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 719–728. [[CrossRef](#)]
94. Zhao, Z.; Liu, F.; Gong, D. An SDN-Based Fingerprint Hopping Method to Prevent Fingerprinting Attacks. *Secur. Commun. Netw.* **2017**, *2017*, 1560594. [[CrossRef](#)]
95. Wang, K.; Chen, X.; Zhu, Y. Random domain name and address mutation (RDAM) for thwarting reconnaissance attacks. *PLoS ONE* **2017**, *12*, e0177111. [[CrossRef](#)]
96. Chowdhary, A.; Pisharody, S.; Alshamrani, A.; Huang, D. 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, Scottsdale, AZ, USA, 24 March 2017; pp. 53–58. [[CrossRef](#)]
97. Wang, L.; Wu, D. Moving Target Defense Against Network Reconnaissance with Software Defined Networking. In Proceedings of the 19th International Conference, ISC 2016, Honolulu, HI, USA, 3–6 September 2016; Volume 9866, pp. 203–217. [[CrossRef](#)]
98. Sun, J.; Sun, K. DESIR: Decoy-enhanced seamless IP randomization. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9. [[CrossRef](#)]
99. Zhang, L.; Wei, Q.; Gu, K.; Yuwen, H. Path hopping based SDN network defense technology. In Proceedings of the 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, China, 13–15 August 2016; pp. 2058–2063. [[CrossRef](#)]
100. Zhao, Z.; Gong, D.; Lu, B.; Liu, F.; Zhang, C. SDN-Based Double Hopping Communication against Sniffer Attack. *Math. Probl. Eng.* **2016**, *2016*, 8927169. [[CrossRef](#)]
101. Achleitner, S.; Porta, T.; McDaniel, P.; Sugrim, S.; Krishnamurthy, S.; Chadha, R. Cyber Deception: Virtual Networks to Defend Insider Reconnaissance. In Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats, Vienna, Austria, 28 October 2016; pp. 57–68. [[CrossRef](#)]
102. Debroy, S.; Calyam, P.; Nguyen, M.; Stage, A.; Georgiev, V. Frequency-Minimal Moving Target Defense using Software-Defined Networking. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016. [[CrossRef](#)]

103. Venkatesan, S.; Albanese, M.; Amin, K.; Jajodia, S.; Wright, M. A moving target defense approach to mitigate DDoS attacks against proxy-based architectures. In Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, USA, 17–19 October 2016; pp. 198–206. [[CrossRef](#)]
104. Aydeger, A.; Saputro, N.; Akkaya, K.; Rahman, M. Mitigating Crossfire Attacks Using SDN-based Moving Target Defense. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, United Arab Emirates, 7–10 November 2016. [[CrossRef](#)]
105. Maleki, H.; Valizadeh, S.; Koch, W.; Bestavros, A.; van Dijk, M. Markov Modeling of Moving Target Defense Games. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, Vienna, Austria, 24 October 2016; pp. 81–92. [[CrossRef](#)]
106. Ahmed, N.; Bhargava, B. Mayflies: A Moving Target Defense Framework for Distributed Systems. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, Vienna, Austria, 24 October 2016; pp. 59–64. [[CrossRef](#)]
107. Venkatesan, S.; Albanese, M.; Cybenko, G.; Jajodia, S. A Moving Target Defense Approach to Disrupting Stealthy Botnets. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, Vienna, Austria, 24 October 2016; pp. 37–46. [[CrossRef](#)]
108. MacFarland, D.; Shue, C. The SDN Shuffle: Creating a Moving-Target Defense using Host-based Software-Defined Networking. In Proceedings of the Second ACM Workshop on Moving Target Defense, Denver, CO, USA, 12 October 2015; pp. 37–41. [[CrossRef](#)]
109. Jafarian, J.; Al-Shaer, E.; Duan, Q. An Effective Address Mutation Approach for Disrupting Reconnaissance Attacks. *Inf. Forensics Secur. IEEE Trans.* **2015**, *10*, 2562–2577. [[CrossRef](#)]
110. Luo, Y.B.; Wang, B.S.; Wang, X.F.; Hu, X.F.; Cai, G.L.; Sun, H. RPAH: Random Port and Address Hopping for Thwarting Internal and External Adversaries. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; pp. 263–270. [[CrossRef](#)]
111. Clark, A.; Sun, K.; Bushnell, L.; Poovendran, R. A Game-Theoretic Approach to IP Address Randomization in Decoy-Based Cyber Defense. In Proceedings of the 6th International Conference, GameSec 2015, London, UK, 4–5 November 2015; pp. 3–21. [[CrossRef](#)]
112. Jafarian, J.; Al-Shaer, E.; Duan, Q. Spatio-temporal Address Mutation for Proactive Cyber Agility against Sophisticated Attackers. In Proceedings of the First ACM Workshop on Moving Target Defense, Scottsdale, AZ, USA, 7 November 2014; Volume 2014, pp. 69–78. [[CrossRef](#)]
113. Jia, Q.; Wang, H.; Fleck, D.; Li, F.; Stavrou, A.; Powell, W. Catch me if you can: A cloud-enabled DDoS defense. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 23–26 June 2014; pp. 264–275. [[CrossRef](#)]
114. Peng, W.; Li, F.; Huang, C.T.; Zou, X. A moving-target defense strategy for Cloud-based services with heterogeneous and dynamic attack surfaces. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 804–809. [[CrossRef](#)]
115. Jia, Q.; Sun, K.; Stavrou, A. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. In Proceedings of the 2013 22nd International Conference on Computer Communication and Networks (ICCCN), Nassau, Bahamas, 30 July–2 August 2013; pp. 1–9. [[CrossRef](#)]
116. Clark, A.; Sun, K.; Poovendran, R. Effectiveness of IP address randomization in decoy-based moving target defense. In Proceedings of the 52nd IEEE Conference on Decision and Control, Firenze, Italy, 10–13 December 2013; pp. 678–685. [[CrossRef](#)]
117. Huang, Y.; Ghosh, A. Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web Services. In *Moving Target Defense*; Springer: New York, NY, USA, 2011; pp. 131–151. [[CrossRef](#)]
118. Dunlop, M.; Groat, S.; Urbanski, W.; Marchany, R.; Tront, J. MT6D: A moving target IPv6 defense. In Proceedings of the MILCOM 2011 Military Communications Conference, Baltimore, MD, USA, 7–10 November 2011; pp. 1321–1326. [[CrossRef](#)]
119. Azab, M.; Hassan, R.; Eltoweissy, M. ChameleonSoft: A Moving Target Defense System. In Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Orlando, FL, USA, 15–18 October 2011; pp. 241–250. [[CrossRef](#)]
120. Narantuya, J.; Yoon, S.; Lim, H.; Cho, J.H.; Kim, D.; Moore, T.; Nelson, F. SDN-Based IP Shuffling Moving Target Defense with Multiple SDN Controllers. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Supplemental Volume (DSN-S), Portland, OR, USA, 24–27 June 2019; pp. 15–16. [[CrossRef](#)]
121. MATLAB. 9.11 (R2021b); The MathWorks Inc.: Natick, MA, USA, 2021.
122. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
123. Spring, N.; Mahajan, R.; Wetherall, D.; Anderson, T. Measuring ISP Topologies with Rocketfuel. *Netw. IEEE/ACM Trans.* **2004**, *12*, 2–16. [[CrossRef](#)]
124. Mininet: An Instant Virtual Network on your Laptop (or other PC). Available online: <http://mininet.org/> (accessed on 14 October 2021).
125. POX, a Networking Software Platform Written in Python. Available online: <https://github.com/noxrepo/pox> (accessed on 14 October 2021).
126. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an operating system for networks. *Comput. Commun. Rev.* **2008**, *38*, 105–110. [[CrossRef](#)]
127. Floodlight, a Community-Developed, Open Source, Java OpenFlow Controller. Available online: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview> (accessed on 14 October 2021).

128. Jikecloud Cloud Services. Available online: <https://www.jikecloud.net/> (accessed on 14 October 2021).
129. Chun, B.; Culler, D.; Roscoe, T.; Bavier, A.; Peterson, L.; Wawrzoniak, M.; Bowman, M. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Comput Commun Rev. Comput. Commun. Rev.* **2003**, *33*, 3–12. [[CrossRef](#)]
130. Berman, M.; Chase, J.; Landweber, L.; Nakao, A.; Ott, M.; Raychaudhuri, D.; Ricci, R.; Seskar, I. GENI: A Federated Testbed for Innovative Network Experiments. *Comput. Netw.* **2014**, *61*, 5–23. [[CrossRef](#)]
131. Chowdhary, A.; Dixit, V.H.; Tiwari, N.; Kyung, S.; Huang, D.; Ahn, G.J. Science DMZ: SDN based secured cloud testbed. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 1–2. [[CrossRef](#)]
132. Ricci, R.; Eide, E.; Team, C. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *Login Usenix Mag.* **2014**, *39*, 36–38.
133. NS3, a Discrete-Event Network Simulator for Internet Systems. Available online: <https://www.nsnam.org/> (accessed on 14 October 2021).
134. Green, M.; MacFarland, D.; Smestad, D.; Shue, C. Characterizing Network-Based Moving Target Defenses. In Proceedings of the Second ACM Workshop on Moving Target Defense, Denver, CO, USA, 12 October 2015; pp. 31–35. [[CrossRef](#)]
135. Dunlop, M.; Groat, S.; Marchany, R.C.; Tront, J.G. Implementing an IPv6 Moving Target Defense on a Live Network, 2012. Available online: <https://vtechworks.lib.vt.edu/bitstream/handle/10919/84190/DunlopIPv62012.pdf> (accessed on 1 September 2022).
136. Rathee, S.; Sinha, Y.; Haribabu, K. A survey: Hybrid SDN. *J. Netw. Comput. Appl.* **2017**, *100*, 35–55. [[CrossRef](#)]
137. Amin, R.; Reisslein, M.; Shah, N. Hybrid SDN Networks: A Survey of Existing Approaches. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3259–3306. [[CrossRef](#)]
138. Krishnan, V.; Serres, O.; Blocksome, M. COnfigurable Network Protocol Accelerator (COPA): An Integrated Networking/Accelerator Hardware/Software Framework. In Proceedings of the 2020 IEEE Symposium on High-Performance Interconnects (HOTI), Piscataway, NJ, USA, 19–21 August 2020; pp. 17–24. [[CrossRef](#)]
139. Tajbakhsh, H.; Parizotto, R.; Neves, M.; Schaeffer-Filho, A.; Haque, I. Accelerator-Aware In-Network Load Balancing for Improved Application Performance. In Proceedings of the 2022 IFIP Networking Conference (IFIP Networking), Catania, Italy, 13–16 June 2022; pp. 1–9. [[CrossRef](#)]
140. Burren, B.; Daly, D.; Debbage, M.; Louzoun, E.; Severns-Williams, C.; Sundar, N.; Turbovich, N.; Wolford, B.; Li, Y. Intel’s Hyperscale-Ready Infrastructure Processing Unit (IPU). In Proceedings of the 2021 IEEE Hot Chips 33 Symposium (HCS), Palo Alto, CA, USA, 22–24 August 2021; pp. 1–16. [[CrossRef](#)]
141. Intel, Explore the Power of Intel® Programmable Ethernet Switch Products, Intel. Available online: <https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html> (accessed on 31 July 2022).
142. Rossi Mafioletti, D.; Mello, R.; Ruffini, M.; Frascolla, V.; Martinello, M.; Ribeiro, M. Programmable Data Planes as the Next Frontier for Networked Robotics Security: A ROS Use Case. In Proceedings of the 2021 17th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 25–29 October 2021; pp. 160–165. [[CrossRef](#)]
143. Defining Insider Threats. Available online: <https://www.cisa.gov/defining-insider-threats> (accessed on 23 July 2022).
144. IBM Security X-Force Threat Intelligence Index 2022. Available online: <https://www.ibm.com/downloads/cas/ADLMYLAZ> (accessed on 23 July 2022).
145. Data Centre Networking: SmartNICs. Available online: <https://ubuntu.com/blog/data-centre-networking-smartnics> (accessed on 2 July 2022).
146. Azure Accelerated Networking: SmartNICs in the Public Cloud. Available online: https://www.usenix.org/sites/default/files/conference/protected-files/nsdi18_slides_firestone.pdf (accessed on 2 July 2022).
147. Miano, S.; Doriguzzi Corin, R.; Risso, F.; Siracusa, D.; Sommese, R. Introducing SmartNICs in Server-based Data Plane Processing: The DDoS Mitigation Use Case. *IEEE Access* **2019**, *7*, 107161–107170. [[CrossRef](#)]
148. Dimolianis, M.; Pavlidis, A.; Maglaris, V. A Multi-Feature DDoS Detection Schema on P4 Network Hardware. In Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 24–27 February 2020; pp. 1–6. [[CrossRef](#)]