IEICE TRANS. COMMUN., VOL.E105–B, NO.11 NOVEMBER 2022

PAPER *Special Section on Towards Management for Future Communications and Services in Conjunction with Main Topics of APNOMS2021*

An SDN-Based Moving Target Defense as a Countermeasure to Prevent Network Scans

Shoya CHIBA[†], **Luis GUILLEN**[††], ***Nonmembers***, **Satoru IZUMI**[†††a)], **Toru ABE**[†,††], ***and*** **Takuo SUGANUMA**[†,††], ***Members***

SUMMARY This paper proposes a Software-Defined Network (SDN)-based Moving Target Defense (MTD) to protect the network from potential scans in a compromised network. As a unique feature, contrary to traditional MTDs, the proposed MTD can work alongside other tools and countermeasures already deployed in the network (e.g., Intrusion Protection and Detection Systems) without affecting its behavior. Through extensive evaluation, we showed the effectiveness of the proposed mechanism compared to existing solutions in preventing scans of different rates without affecting the network and controller performance.

key words: *MTD, IPS, network scan, SDN*

1. Introduction

Targeted attacks against companies, especially Advanced Persistent Threat (APT) attacks such as ransomware [1], are causing havoc around the world. Attackers often send malware via email to targeted PCs, and once compromised, these infected PCs start deploying different operations to spread the infection. For example, Wanna-cry [2], one of the most well-known ransomware, performs a reconnaissance of the network infrastructure after the initial infection. The main goal of the reconnaissance is to collect information such as, among others, network topology, IP addresses, and open ports. As a countermeasure, companies usually implement different defense technologies such as Intrusion Prevention Systems (IPS), Intrusion Detection Systems (IDS), or Firewalls (FW), which are implemented with a set of rules to detect/protect from attacks. For instance, these countermeasure systems might block IPs from sources trying to perform an IP or Port scan within a short given time (i.e., high-rate scans). However, it is challenging for IPS/IDSs and FWs to detect an attack if it is performed using low rate scans or when the scan interval is not periodical [3].

Therefore, Moving Target Defense (MTD) [4], has been recently used as a complementary mechanism to deal with this problem. As its name imply, in MTD, the idea is to protect the target by frequently moving/changing it. For instance, if an attacker is performing an IP scan, the MTD would periodically change the IP address from hosts so that the information collected by the attacker is not valid after a specific time. Thus, this mechanism can be ideal for preventing low-rate reconnaissance attacks.

On the other hand, Software Defined Network (SDN) [5] has been used to deploy MTDs. The core idea of SDN is to separate the control from the data plane, which allows innovative and flexible network management. In the particular case of MTD, SDN can manage the moving of the targets (e.g., PCs, Servers) directly by handling the network traffic instead of an actual physical change of the target (e.g., by changing the IP addresses). The implementation of SDN-based MTDs has proven to be adequate to protect from attacks [9], [10]. However, since MTD is deployed alongside other countermeasure systems (e.g., IPS, FW), there is a mismatch of countermeasures due to the lack of coordination, which ultimately disrupts the user experience and makes it more vulnerable to attacks. For instance, if the IPS/IDSs or FWs are programmed to protect specific servers using the IP address, if this IP is changed periodically as an MTD policy, it becomes increasingly challenging for both to protect the target effectively.

Therefore, in this study, we propose a mechanism to allow both MTD and IPSs to work transparently with each other. In a preliminary version of this study [6], we showed the feasibility of its implementation by designing an SDN-based MTD which considers the interaction with an IPS. This paper extends our preliminary study by an updated literature review, focusing on the related work on SDN-based MTDs for IP defense. Moreover, we also conducted an extensive evaluation of different setups and parameters to test the proposal's effectiveness in both high and low-rate scans compared to using individual countermeasures.

The remainder of this paper is organized as follows. Section 2 presents a thorough overview of the related work. Then, Section 3 introduces the proposed mechanism, which is then extensively evaluated in Sect. 4. Finally, Sect. 5 concludes this paper with some final thoughts and future work.

2. Related Work

MTD has been vastly studied in different areas where the core idea is to defend the target by making it difficult to launch attacks. There are different properties of the target that can be "moved;" however, three factors must be considered:

Manuscript received December 10, 2021.
Manuscript revised March 31, 2022.
Manuscript publicized May 27, 2022.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-8579 Japan.
[††]The authors are with the Cyberscience Center, Tohoku University, Sendai-shi, 980-8578 Japan.
[†††]The author is with the National Institute of Technology, Sendai College, Sendai-shi, 989-3128 Japan.
a) E-mail: izumi@sendai-nct.ac.jp

DOI: 10.1587/transcom.2021TMP0020

what would be the target, *when* moving it, and *how* to do it [7]. In that regard, the most common form of MTD, and also the focus of this paper, is the so-called Random Host Mutation (RHM), which consists of changing the IP, port, host identity, or instruction set from the target [8]. Moreover, SDN makes the ideal candidate perform the RHM operations as it can flexibly control the communication in the network. Therefore, in this section, we focus on SDN-based RHM implementations.

One of the most representative SDN-based MTDs that implement RHM is OF-RHM [9], which changes the IP address at regular intervals to obfuscate the scans. OF-RHM is designed for OpenFlow (OF) [12] networks, which is the de facto standard for SDN, and its main feature consists of rewriting the IP address of the packet header received on the switches. To do so, the authors modify the real IP address (*rIP*) to a different virtual IP address (*vIP*). Moreover, the mapping between *vIP* and *rIP* is updated at regular intervals, making it difficult for low-rate reconnaissance attacks. Although OF-RHM showed an effective defense mechanism transparent to the end-user, it notifies the *vIP*s using a DNS Server, which becomes the attacker's target.

CHAOS [10] presents an improvement by using IDS to identify suspicious hosts while at the same time randomizing the IP addresses. The role of the IDS is to block only the ports of suspicious hosts when a high-rate scan is detected. The IP randomization is performed using SDN and a hierarchical structure to make it more unpredictable and flexible. Although CHAOS is capable of decreasing information disclosure; however, it heavily relies on the assignment of obfuscation and random index values, which vary depending on the security level of each host.

FRVP [11] is another interesting approach that uses an SDN-based RHM. In FRVP, the core idea is to flexibly and randomly multiplex virtual IPs; the main difference is the use of variable time to map *M rIP* to *N vIP*s. The authors also derive a probabilistic model from evaluating the effectiveness of their proposal. However, it might be challenging to find the optimal balance of security and performance when relying on a single countermeasure. Moreover, since *vIP*s are assigned to each service, multiple *vIP*s are assigned to servers running multiple services, which will make it difficult to monitor for the already deployed mechanisms [13].

From the above-mentioned related work, we can conclude that although they present innovative and relatively effective countermeasures to scan attacks, they did not consider their effect on other implemented defense mechanisms, such as IPS. The main issue is that when both MTD and IPS are deployed in parallel, it is necessary to determine the optimal change interval while setting a scan detection time according to the attacker's strategy, especially the scan rate. Therefore, this paper fills the gap in those works by presenting a mechanism that allows both RHM and IPS to protect from different rates of scan attacks conjointly.

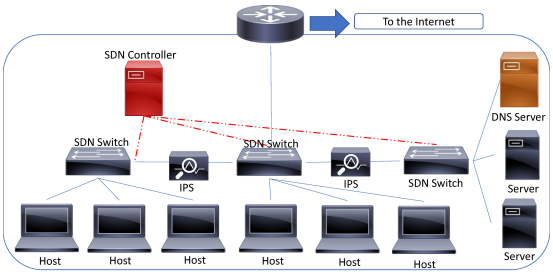

Fig. 1 Assumed environment.

3. Proposed Anti-Scan MTD Mechanism

3.1 Assumed Environment

Before delving into the proposed MTD, Fig. 1 depicts the assumed environment in this study. As observed, we assumed a Local Network whose core infrastructure is SDN-enabled through OF-switches that handle both the internal and external traffic. Moreover, all users share the same subnet and connect to various servers (e.g., file servers, Virtual Machine servers), limited to local usage, while other servers (i.e., mail, web) can be accessed from outside the network. Finally, note that a DNS server performs the address translation to the local servers and the Web server, accessing the Internet.

As security measures, a FW is placed on the edge of the network, and an in-line IPS is installed to protect the gateway from external attacks. The MTD is implemented as a function of the SDN controller. Note that, in this research, to avoid affecting the services, the public server in the Demilitarized Zone (DMZ) is not subject to MTD.

We assume that the attacker has already successfully infected a host and performed the initial intrusion. For instance, in a targeted attack, the user opened an email with malware that executed malicious batches. Since the malware was sent via (encrypted) email, the FW and IPS were easily breached; thus, the next step in the privilege escalation is reconnaissance via an IP scan.

3.2 Overview of the Proposed MTD

Having a single countermeasure (i.e., MTD, IPS, FW) is ineffective in preventing the attack scenario presented in the preceding section. Therefore, a set of them need to be implemented; however, the mentioned in Sect. 2 the inherent properties of MTD mechanisms (i.e., changing values) may adversely affect this interaction. Therefore, this study proposes a mechanism where an MTD can seamlessly work with other attack countermeasures (i.e., IPS).

As in the case of the related work in Sect. 2, the proposed mechanism also implements RHM using *rIP*s and *vIP*s. In particular, we perform address randomization to end hosts using *vIP*s, while keeping these changes transparent to IPS. Hence, the IPS can monitor the potential attacks based on

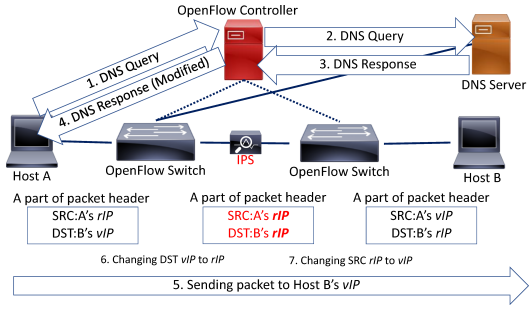

Fig. 2 Overview of the proposed method.

the *rIP* which does not change over time.

Figure 2 shows an overview of how the proposed MTD handles the end-to-end communication. As observed, when the source packet reaches the edge switch, it is forwarded as an OF *packet-in* message to the controller. Note that a packet-in message will only be created if no rules already match the header fields. When Host A (source) sends a packet to Host B (destination), the source has to get the destination's *vIP*. Therefore the source host sends a DNS query to the DNS server through the controller. Initially, the controller verifies whether the source host is authorized. This procedure is coordinated with the existing IPS, FW, or other security mechanisms (in the assumed environment, we use an IPS). The controller then sends the DNS query, which replies with the destination's *vIP* to install the corresponding rules to the switches. The specific steps to install the rules will be detailed in Sect. 3.3. Finally, the traffic can resume until the MTD changing interval time expires.

Note that, in existing methods (i.e., [9], [11]), the packets traverse between switches based on their *vIP*s. However, when *vIP*s change frequently, the other countermeasures cannot monitor the traffic effectively. Thus, in the proposed MTD, the traffic between switches is routed using the *rIP*s, relying on the use of *vIP*s only between end hosts. This makes it transparent to IPSs or other attack countermeasures to continue performing their duties without being affected by the MTD strategy.

Therefore, the IPS can, for instance, detect high-rate scans, which can be completed within the MTD interval and block compromised hosts as soon as it is detected, while the MTD can handle the low-rate scan threats. For example, the MTD can set up a policy using the number of attempts ($n = host/T$) to access a local resource ($host$) within a specific time (T). If n is above a set value, it can be concluded to attempt to perform a low-rate scan. By using this strategy, can even support time-varying intervals without interfering with the IPS/IDS policies.

3.3 System Design and Configuration

The proposed MTD implements the following features in the OpenFlow Controller to handle the defense without interfering with other countermeasures:

- ARP and IP packet handler
- End-to-end address translation
- Scan prevention

The following sections present details of each of them.

3.3.1 ARP and IP Packet Handler

The packet handler manages the Packet-In messages issued to the OpenFlow controller once an unknown packet arrives at the switches. This module rewrites the packet headers of ARP and IP packets. Then, when the source host tries to obtain the MAC address of the corresponding destination *vIP* using ARP, the controller inspects the ARP packet and installs a flow rule in each switch involved in the destination path. To do so, the destination *vIP* is changed to its *rIP*, while the source *rIP* to its *vIP* when both are connected on the same switch. However, if they are not connected, the packet is forwarded to the next switch with the source and destination *rIP*. This process is repeated until the destination host receives the ARP request and uses its *rIP* to reply to the source *vIP* to complete the request. As a result, each host can obtain and communicate with the MAC address corresponding to *vIP* without knowing the *rIP* of the destination host.

Next, the switches receive IP packets. Like ARP packet processing, if the destination and source are connected to the same switch, the destination *rIP* is rewritten to *vIP*, and the source *vIP* is rewritten to *rIP*. Otherwise, the packet's destination *vIP* is rewritten to *rIP* and forwarded to the switch connected to the destination host. Finally, the switch rewrites the source *rIP* to *vIP* and forwards the packet to the destination host. Finally, the controller installs the rules to handle the ARP and IP packets on each switch until the following address change. The same operation is performed when the destination and source hosts are switched.

Also, as part of the IP packet handler, the controller intercepts the DNS translation, rewrites the *rIP* in the DNS record to *vIP*, and replies.

The mapping between *rIP*-*vIP* is done using a host-switch connection table which contains a list of the switches and their corresponding connected hosts. Moreover, the controller also has an *rIP*-*vIP* mapping table updated after the address change interval elapses.

Since the transit source and destination between the switches uses the *rIP*, it is possible to collaborate transparently with the existing IPS. The scan rate set at the IPS can be based on the address change interval set in the controller so that if a high-rate scan happens after that interval, it can still be detected and blocked.

For broadcast packets, we did not include a specific processing in the current implementation, as we used the default functionality provided by the controller to handle those packets. In particular, those packets are dropped by the switch.

3.3.2 End-to-End Address Translation

This feature is also implemented as part of the Packet-In

handler at the controller. The main goal is to act as a mediator between end-hosts and the DNS server. To do so, when a host in the network attempts a name resolution, the switch receives a packet containing a DNS query, which is then forwarded to the controller.

The query resolves the hostname from its *vIP*; therefore, the controller must perform a mapping between the *rIP* and *vIP*, which rewrites the headers and forwards them to the DNS server.

The DNS server replies with a record of the corresponding hostname. With all this information, the controller can build the rule and install it to the corresponding switch. Note that no flow rule is installed to facilitate implementation to address resolution to the DNS server. Thus, the Packet-In message is sent every time a new translation is required. However, a record representing the correspondence and mapping is registered.

By this mechanism, hosts can connect to the destination using the hostname instead of the time-varying *vIP*.

The overall process of these functions is detailed in Algorithms 1. As observed, upon receiving a Packet-In message, the controller evaluates the type of the packet (i.e., ARP, IP), then it checks if the destination IP is *vIP* and its source host is directly connected. Then, it installs a flow rule that rewrites the destination *vIP* to *rIP*, the source *rIP* to *vIP*, and then forwards the packet. Otherwise, it rewrites the destination *vIP* to *rIP* and installs a flow rule to forward to another switch.

When the switches receive packets from other switches that are not directly connected to the source or destination, they relay the packet. Note that the controller verifies if both the destination and source addresses are *rIP*. When the destination host is connected, the source *rIP* is rewritten to *vIP* and forwarded to the destination. We drop the packet if the destination is connected to the same switch as the source host. This is because, in principle, only if the destination host is directly connected and the source is using its *vIP*, the packet is forwarded to the destination host. However, if the destination host is not connected, the address is not rewritten, and the packet is forwarded to the next switch.

Algorithm 1 Packet-In Handler

```
1:  procedure PACKET-IN-HANDLER(pkt)
2:      if pkt is arp_packet then
3:          if isrIP(pkt.arpdst) and isvIP(pkt.arpdst) then
4:              if isConnected(pkt.arpdst) and isConnected(pkt.arpsrc)
    then
5:                  v2r(pkt.arpdst)
6:                  r2v(pkt.arpsrc)
7:          else
8:              v2r(pkt.arpdst)
9:          if isrIP(pkt.arpdst) and isrIP(pkt.arpsrc) then
10:             if isConnected(pkt.arpdst) and isConnected(pkt.arpsrc)
    then
11:                 drop(pkt)
12:             else if isConnected(pkt.arpdst) then
13:                 r2v(pkt.arpsrc)
14:             else if isConnected(pkt.arpsrc) then
15:                 drop(pkt)
16:             else
17:                 None
18:         else if isvIP(pkt.arpdst) then
19:             None
20:         else
21:             drop(pkt)
22:     else if pkt is ip_packet then
23:         if pkt is dns_response then
24:             r2v(pkt.record.targetip)
25:         if isrIP(pkt.ipsrc) and isvIP(pkt.ipdst) then
26:             if isConnected(pkt.ipdst) isConnected(pkt.ipsrc) then
27:                 v2r(pkt.ipdst)
28:                 r2v(pkt.ipsrc)
29:             else
30:                 v2r(ipdst)
31:         if isrIP(pkt.ipdst) and isrIP(pkt.ipsrc) then
32:             if isConnected(pkt.ipdst) and isConnected(pkt.ipsrc) then
33:                 drop(pkt)
34:             else if isConnected(pkt.ipdst) then
35:                 r2v(pkt.ipsrc)
36:             else if isConnected(pkt.ipsrc) then
37:                 drop(pkt)
38:             else
39:                 None
40:         else if isvIP(pkt.ipdst) then
41:             None
42:         else
43:             drop(pkt)
44:     installRules(pkt)
```

3.3.3 Scan Prevention

The scan prevention feature verifies that only a specific host can communicate via *rIP*. To do so, the controller checks whether the destination host is in the list of allowed to connect using its *rIP*. If it is on the list, the address is not changed and is forwarded (as a standard L2 switch would do). Therefore, initially, each host uses the *rIP* of the DNS server as the destination so that it does not interfere with the communication with the statically configured gateway or DNS server. This mechanism can also be applied to some hosts with higher privileges (e.g., network administrators).

The scanning attack is prevented by coordinating all these elements while keeping its roles independent and transparent to each other. In particular, the MTD relies on the address change interval (T), which updates the mapping table (*rv*MAP) every T [s], to prevent low-rate scanning. While the IPS relies on the Detection Scan Rate (DSR) threshold to block the high-rate scans. Therefore, if the DSR of a host is less than the set value, the activity is deemed normal. However, if it is higher, it will be considered a network scan and, consequently, blocked. Even after the address change, the combination of these features can continuously isolate the attackers.

4. Evaluation

To evaluate the proposed mechanism, we conducted various experiments to:

1. Measure the effectiveness as a scan countermeasure
2. Verify the correctness of other implemented tools (i.e.,

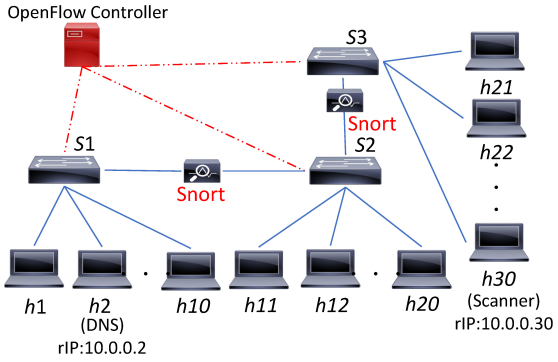

Fig. 3 Topology used in the experiments.

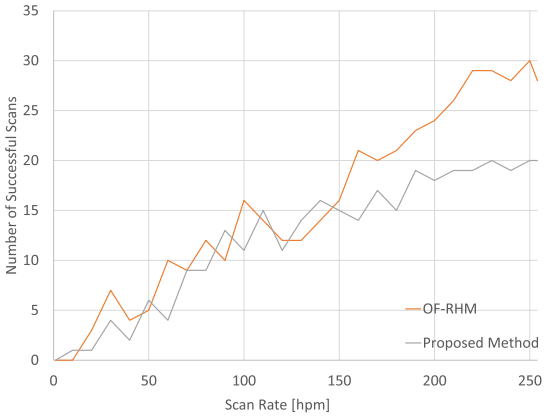

Fig. 4 Number of successful scans per scan rate in a single T from $h30$.

IPS) without being affected by the MTD
3. Evaluate the impact on the system performance

4.1 Experimental Setup

To conduct the experiments, we built the topology shown in Fig. 3 using the network emulator Mininet 2.3 [14], deployed in a physical machine with 16GB of RAM, a Core-i5 4 CORE processor, and running Ubuntu 20.04 LTS. As observed, this topology consists of 30 hosts, 2 IPSs (Snort [15]), 1 SDN controller (Ryu [16]) and 3 switches. All links are 1000 Mbps, and Snort is executed as an inline IPS located between $S1$ and $S2$, and between $S2$ and $S3$ in Fig. 3.

In this implementation, unused IP addresses in the subnet are used as *vIP*. The *rIP* for each host (hx) in the topology is "10.0.0.x.", so that the *rIP* range is from "10.0.0.1" to "10.0.0.30", while the *vIP* range is the rest of the subnet 10.0.0.0/24, from "10.0.0.31" to "10.0.0.254". In addition, Host $h2$ on the topology is running as DNS server; thus, in charge of the *vIP* notification by registering a correspondence between *rIP* and hostname. For simplicity, in the current implementation, the hostname of each host hx was set to $hostx.lo$.

4.2 Methodology

Initially, to measure the effectiveness as a scan countermeasure, we performed an ICMP scanning for two cases: short scans performed by a single host and long-term scans performed over multiple addresses changes.

Then, to verify the correctness of the implementation when other tools are deployed, we assumed three attack scenarios where the compromised host is in a different area of the network. For this experiment, we use a modified topology with more elements.

Finally, to evaluate the impact on the performance by applying the proposed mechanism, we measured the CPU usage in the system compared to an existing work [9].

4.3 Results

4.3.1 Anti-Scan Effectiveness Based on the Scan Rate

First, we evaluated the effects on short-term scans executed by a single host. To do so, we used $h30$ in the topology shown in Fig. 3, set the controller's address change interval $T = 120$, and the scan rate DSR to 10 [host]/120 [s] at the IPS. Moreover, the attack parameters were as follows: The attack range was the full subnet 10.0.0.0/24. In addition, the scans were performed for 120 s on each scan rate, which was progressively increased by 10 hpm in every run, except for the initial and final rates, which were incremented by one unit (i.e., Scan rate = 1–10, 20, 30, $\cdots$, 240, 250, 251–254).

Figure 4 shows results obtained. In this experiment, "the number of successful scans" refers to the number of times a (malicious) scanner could scan and obtain a response from hosts in the target network. We used this metric for evaluation, as the information obtained every time a scan is successful could mean a potential attack. As observed, the number of scans varies depending on the scan rate. For instance, when the scan rate is 150 hpm or less, the number of successful scans is relatively the same for both methods. However, the proposed mechanism has considerably fewer successful scans when the scan rate is above that value. This is because the IPS blocked the scans which were above the threshold. Note that, since this experiment was conducted within a single interval (T), the change of address was not performed.

Also, note that OF-RHM is used as the existing MTD in this experiment, which does not use IPS to block scanning, while the proposed method combines the MTD strategy and IPS functionality. Therefore, there is a difference between their performances.

Next, we measured the effectiveness of the anti-scan mechanism when the attack spans multiple address change intervals (T). For this experiment, we performed the attack for 1200 s from $h30$ using the same DSR (10 [host]/120 [s]) for the IPS as in the previous experiment. However, we used for different scan rates for $h30$, namely low-rate (4 hpm),

Table 1 Total number of successful scans per approach.

Rate [hpm]	IPS	OF-RHM	OF-RHM+IPS	Proposed Method
4	14	6	8	6
64	90	120	134	77
128	111	285	287	101
254	210	577	574	198

intermediate (64 hpm), high (128 hpm), and the highest (254 hpm). Also, for this experiment we compared the following approaches.

- Use IPS only (IPS)
- Use only existing MTD [9] (OF-RHM)
- Combined use of OF-RHM and IPS (OF-RHM + IPS)
- Proposed method (Proposed MTD + IPS)

Table 1 shows the total number of scans in 1200 s, when the attack uses a constant rate during the whole attack. The results were as follows:

- Slow-rate attack (4 hpm)
  The first row in Table 1 shows the results of the number of successful scans using a slow-rate attack (4 hpm). As observed, since the attack does not trigger the IPS' *DSR*, the IPS-only approach is the one that performs the worst, leading to ultimately 14 successful scans. On the other hand, the approaches that use MTDs had relatively the same effect since they all implement almost the same OF-RHM functionality. However, note that the OF-RHM+IPS approach was slightly affected by the combined use of those countermeasures. Moreover, as observed in Fig. 5, which shows the cumulative number of scans, since the IPS cannot prevent the attack to hosts connected to the same switch, the number of successful scans increases over time with either method. This is also applicable at all rates. However, the address changes at regular intervals when using an MTD; therefore, a host may be assigned to an already scanned address when the time elapses. Therefore, it will not be scanned again, successfully decreasing the attack's impact.
- Normal-rate attack (64 hpm)
  The second row of Table 1 shows the results of the normal-rate attack (64 hpm). As observed, at this rate the advantage of the proposed approach is bigger. Since the scan does not span the entire subnet, in the methods that use the combined strategy (MTD+IPS), the scans can be reduced to up to 60% compared to using a single countermeasure. Still, the proposed method has a higher advantage at this scan rate.
- High-rate attack (128 hpm)
  Next, the third row of Table 1 shows the results of a relatively high-rate scan (128 hpm). As observed, the difference between the single and the combined approach is significant but still between the 60% range. As it is a high-rate scan, the IPS can make early detection and blocking, which the MTD then enhances. Nevertheless is not enough to entirely prevent the scans.
- Highest-rate attack (254 hpm)

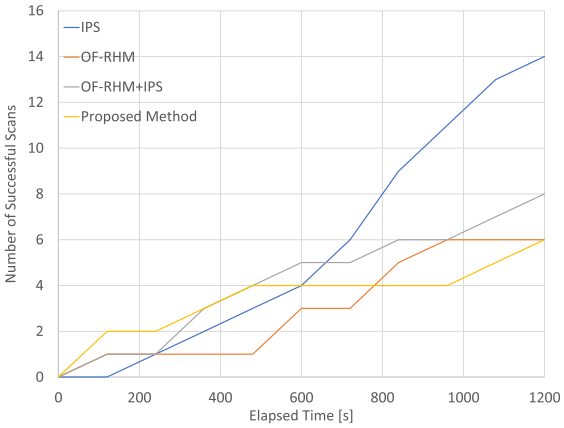

Fig. 5 Cumulative number of scans per approach at 4 hpm.

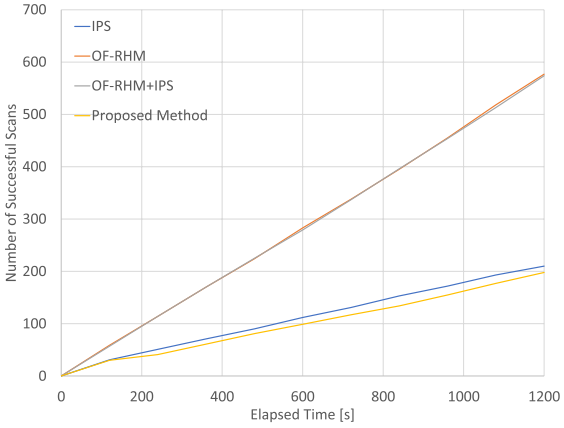

Fig. 6 Cumulative number of scans per approach at 254 hpm.

Finally, the last row of Table 1 shows the results of the highest rate (254 hpm). Note that, since it would be relatively easy for the IPS to detect and block an attack at this rate, the methods that use an IPS can considerably decrease the scan. As observed in Fig. 6, which shows the cumulative number of scans, the trend is almost the same as the case of the high rate. However, note that the proposed approach still has the advantage compared to the others. Of course, by fine-tuning the MTD interval and the IPS, this result can be even more significant without affecting their regular operation. Unlike slow scans, the rate of increase in the number of scans when using the OF-RHM is higher than when using IPS-only and the proposed method.

Since the scans usually are done sequentially, in the IPS-only scenario, the scan would go unnoticed for low-scan rates (as pointed out in Sect. 1). In contrast, the ones that include MTD, since the IPs are changed every interval (T), the information obtained would not be valid once the process renews the addresses. Therefore, in a single interval, the difference is marginal while in a relatively longer period the difference is more evident, as shown in Figs. 4–6.

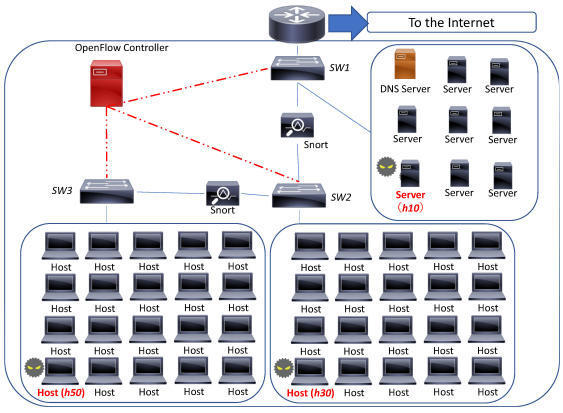

Fig. 7 Topology used for Experiment 2.

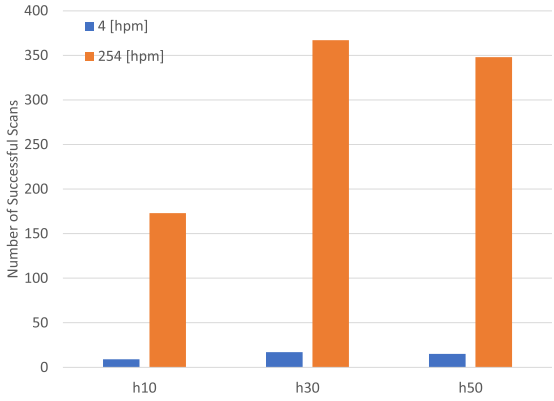

Fig. 8 Number of scans based on the location of the initial compromised host.

4.3.2 Evaluation Based on Attacker's Location

For this experiment, as shown in Fig. 7, we built a variation on the initial Mininet network topology comprised of 50 hosts, 3 OF switches, 2 IPSs, and a single controlled. Moreover, the address change interval (T) is set to 120 s.

All hosts have a default gateway, and the DNS server is configured in advance. From those 50 hosts, we assume that 8 are servers all connected to $SW1$ (as shown in Fig. 7). The other two switches ($SW2$ and $SW3$) have 20 hosts representing users in the LAN.

We also installed Snort between each switch with a detection scan rate set to 10 [hosts]/120 s. The *rIP* ranges from 10.0.0.1 to 10.0.0.50, while the *vIP* uses the IPs from .51 to .254.

This experiment investigates the number of successful scans per approach when the initial compromised host is located in different parts of the network. We assume three scenarios:

- Scenario A: $h10$ is assumed to be a server that might be running open-source software, and a backdoor was incorporated by, for instance, a supply chain attack.
- Scenario B: $h30$ is assumed to be host connected to $SW2$, which was affected by a backdoor malware.
- Scenario C: Similarly to Scenario B, in this case, $h50$ is the compromised host.

The attack was performed for 1200 s in all cases, using two different scan rates, i.e., low (4 hpm) and high (254 hpm).

Figure 8 shows the results of the number of scans depending on the location of the compromised host. As observed, the difference is considerable from scenario A to the others. The success in the scan is almost twice in scenarios B and C. The main factor that influences the result is the number of hosts connected to the same switch.

If the scan is performed within the same switch without going through IPS. The IPS cannot detect or block the attack. Therefore, additional measures would need to be implemented to solve this issue. Also, note a slight decrease from scenario C to B since the scan packet needs to go through one of the IPSs; some of those attempts were blocked.

5. Conclusion

This paper presents an MTD mechanism that prevents network resources from being scanned, allowing other security countermeasures (e.g., IPS, FW) to work transparently. Through extensive experimentation in different scenarios, we showed that using the proposed approach and an IPS, it is possible to decrease the disclosed information in both high- and low-rate attacks without affecting the overall performance. Furthermore, while the MTD is more effective for low-rate scans, the IPS can monitor/block high-rate ones.

In future work, we plan to perform more evaluations with more realistic examples, specifically, the impact on upper layers and the effect as a scan countermeasure. Moreover, since OpenFlow (OF) Random-Host Mutation (OF-RHM) can support the matching OF fields; we plan to study the effects of the proposed method on other fields (e.g., IP addresses, port numbers, MAC address, EthType).

Also, we plan to investigate other various aspects of the proposed method. For instance, the address assignment strategy; the applicability to IPv6 and its effect on the Neighbor Discovery Protocol to perform address resolution; and the performance and controller load. Finally, it would also be interesting to study the effect of the address change interval on short- and long-term communication.

References

[1] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," IEEE Commun. Surveys Tuts., vol.21, no.2, pp.1851–1877, 2019.
[2] M. Akbanov, V.G. Vassilakis, and M.D. Logothetis, "WannaCry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms," J. Telecommunications and Information Technology, no.1, pp.113–124, 2019.
[3] NMAP.org, Timing and Performance — nmap Network Scanning, https://nmap.org/book/man-performance.html (Accessed on 05/20/2021).

[4] U.S. National Science and Technology Council, "Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program," Federal Cybersecurity: Strategy and Implementation for Research and Development, pp.69–99, 2011.
[5] Software-Defined Networking (SDN) Definition, Open Networking Foundation, https://opennetworking.org/sdn-definition/ (Accessed on 05/28/2021).
[6] S. Chiba, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "Design of a network scan defense method by combining an SDN-based MTD and IPS," Proc. 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS2021), pp.273–278, 2021.
[7] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," IEEE Commun. Surveys Tuts., vol.22, no.3, pp.1909–1941, 2020.
[8] Y. Yang and L. Cheng, "An SDN-based MTD model," Concurrency Comput. Pract. Exper., vol.31, no.21, e4897, pp.1–13, 2018.
[9] J.H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," Proc. 1st ACM International Workshop on Hot Topics in Software Defined Networks, pp.127–132, 2012.
[10] Y. Shi, H. Zhang, J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, F. Yan, and B. Zhao, "CHAOS: An SDN-based moving target defense system," Security and Communication Networks, vol.2017, pp.1–11, 2017.
[11] D.P. Sharma, D.S. Kim, S. Yoon, H. Lim, J.H. Cho, and T.J. Moore, "FRVM: Flexible random virtual IP multiplexing in software-defined networks," Proc. 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications, pp.579–587, 2018.
[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol.38, no.2, pp.69–74, 2008.
[13] B. Van Leeuwen, W.M.S. Stout, and V. Urias, "Operational cost of deploying moving target defenses defensive work factors," Proc. IEEE Military Communications Conference MILCOM, pp.966–971, 2015.
[14] Mininet: An Instant Virtual Network on Your Laptop (or other pc) -mininet, http://mininet.org/ (Accessed on 05/21/2021).
[15] Snort — Network Intrusion Detection & Prevention System, https://www.snort.org/ (Accessed on 05/20/2021).
[16] Ryu SDN Framework, https://ryu-sdn.org/ (Accessed on 05/21/2021).

Shoya Chiba received the B.S. from Tohoku Institute of Technology, and he is currently a master student at the Graduate School of Information Sciences at Tohoku University, Japan. His main research interests include cyber security and SDN. He is a member of the IPSJ.

Luis Guillen received his M.S. degree in Computer Science from Vrije Universiteit Brussel (Belgium) in 2012, and his Ph.D. in Information Science from Tohoku University in 2020. From 2020 to 2022 he worked as a Specially Appointed Assistant Professor with the Research Institute of Electrical Communication (Tohoku University), and he is currently an Assistant Professor with the Cyberscience Center, Tohoku University. His research interests include HCI, IoT, network management, resilient networks, SDN, green ICT, DASH, and QUIC. He is a member of IEEE, ComSoc, and SIB (Bolivia).

Satoru Izumi received the M.S. and Ph.D. degrees from Tohoku University, Japan, in 2009 and 2012, respectively. He is currently an Associate Professor with National Institute of Technology, Sendai College. He received the IEEE GCCE 2013 Excellent Paper Award. His main research interests include semantic Web, ontology engineering, green ICT, and SDN. He is a member of the IEICE and the IPSJ.

Toru Abe received the M.E. and Ph.D. degrees in information engineering from Tohoku University, in 1987 and 1990, respectively. From 1990 to 1993, he was a Research Associate with the Education Center for Information Processing, Tohoku University. From 1993 to 2001, he was an Associate Professor with the Graduate School of Information Science, Japan Advanced Institute of Science and Technology. He is currently an Associate Professor with the Cyberscience Center, Tohoku University. His research interests include image processing and knowledge engineering.

Takuo Suganuma received the M.S. and Ph.D. degrees in engineering from the Chiba Institute of Technology, Japan, in 1994 and 1997, respectively. He is currently a Professor and the Director with the Cyberscience Center, Tohoku University, Japan. His main research interests include agent-based computing, flexible network middleware, network management, symbiotic computing, green ICT, and Disaster-resistant communications. He is a member of the IEICE, the IPSJ, and the IEEE. He received the UIC-07 Outstanding Paper Award in 2007.