Moving Target Defense Against Network Reconnaissance with Software Defined Networking

Li Wang and Dinghao Wu

College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
{lzw158,dwu}@ist.psu.edu

Abstract. Online hosts and networks are easy targets of network attacks due to their static nature, which creates an information asymmetry and makes them easy to attack and hard to defend. To break the asymmetry, Moving Target Defense was proposed to bring uncertainties to computer systems. It can be applied to all levels of protections, covering applications, system software, operating systems, and networks. In this paper, we present, Sniffer Reflector, a new method to practice Moving Target Defense against network reconnaissance, which is usually considered as the very first step of most attacks. Sniffer Reflector employs Software-Defined Networking to disturb network reconnaissance. We use virtualization to provide an obfuscated reconnaissance result for attackers. Our method can be easily combined with existing security tools for network forensics as well. We have developed a prototype in a virtual local area network. Our experiment results show that Sniffer Reflector is effective and efficient in blurring various network reconnaissance.

Keywords: Network reconnaissance, Network reflector, Software-Defined Networking, Moving Target Defense, Shadow networks

1 Introduction

Online hosts and networks are easy targets of various attacks due to their static nature. Under the current Internet architecture, it is not easy for networked computer systems to change their network parameters once being established. Most networked services are deployed with a set of well-devised computing infrastructure and serve in a stable network environment. For example, typically, a web server open to public visit will be deployed with a fixed domain name and connected to a physical network device, router or switch, and assigned a public IP address locatable on the Internet. Once deployed, the web server's network parameters will not be changed frequently. This is a good practice because users can easily get online service through the server's domain name or IP address. However, this also exposes valuable network information to attackers for malicious use. Theoretically, attackers have unlimited time to study the server's

network environment and find out a method to finally take over it. Although existing security tools, like firewalls and intrusion detection systems, can prevent most common attacks, these protections are essentially static and cannot change the static nature of the online servers. The static nature of online hosts and networks leads to an asymmetry between the attackers and defenders, and the attack and defense game is always unfair.

To change the attack and defense game, Moving Target Defense (MTD) was proposed to break the asymmetry between the attacker and defender. By introducing uncertainties and diversifications, MTD provides a dynamic defense environment for adversaries. Adversaries are forced to reprobe, reassess and restudy the protected environment. For example, an MTD strategy can periodically change a part of an operating system that makes it harder for attackers to find a known vulnerability for a specific OS. With MTD, it is hard for adversaries to decide and verify the authenticity of the obtained information. Therefore, adversaries can hardly start effective attacks in an MTD protected environment. Inspired by the promising defense philosophy of MTD, a lot of research has been proposed [8] to seek adaptive, dynamic, and practical security solutions for modern computer systems.

Software-Defined Networking (SDN) is a rising network technology which offers sufficient control flexibilities for users to modify network behaviors on the fly. Consider the fact that more and more critical services are moved from offline to online, network naturally becomes the top attack vector in most attack scenarios. Attackers have to make use of network infrastructures to achieve their attack goals. Typically, there are four attack phases for an attack over network [13]: network reconnaissance[1], targeting vulnerable machines, finding exploits, and conducting effective attacks. Each phase represents an attack step. For example, network reconnaissance is used to collect effective information in a target network, like how many nodes are alive, what are the versions, and so on. It provides valuable information for an attacker to conduct the rest attack steps. Existing research results show that, port scan should be regarded as the initial step of the cyber attack routine [2, 6], and more than 70% of the network scans [16] are connected with attack activities. We manage to use MTD to mitigate network reconnaissance with the SDN technologies.

Scan is a special network activity, which can be used both for protectors and attackers. Protectors, like security engineers and network administrators, use network scans to assess the security status of a target network. In most cases, scan activity is regarded highly dangerous in real networks. In our work, unless with a special mention, we recognize network scan as an attack activity.

In this paper, we present Sniffer Reflector, a new MTD method against network scans with SDN. Different from traditional protection mechanisms, Sniffer Reflector does not block or drop network scan traffic. Instead, it reflects scan traffic to a shadow network where scan replies are generated and obfuscated. Shadow network can simulate arbitrary network structures and services with an

[1] Technically, the terms *network reconnaissance* and *network scan* are exchangeable when describing network probe activities. We use them equally in this paper.

acceptable overhead. Therefore, attackers can only obtain obfuscated network views from a shadow network instead of a target network. With our method, attackers can no longer collect effective network information through network reconnaissance. Consequently, it is hard for attackers to continue the rest three attack phases and finish desired attacks.

In summary, the main contributions of Sniffer Reflector are as follows:

- *Provide a new MTD method against network reconnaissance.* As far as we know, it is the first work that employs MTD on network scan. We try to prevent attackers from collecting effective network information through network scan. Consider attackers rely on scan responses to collect vulnerability information, our method fail attackers by obfuscating the returned scan responses.
- *Obfuscate attackers' view with shadow network.* We use shadow network to provide forged responses to scan traffic. Shadow network can establish any desired network environment to obfuscate attackers' view, which is an invisible and isolated environment, and implemented with small overhead through the virtualization technologies.
- *Achieve stealthy protection.* Sniffer Reflector provides "stealthy" protection against network scan. Here, "stealthy" means an attacker cannot detect the fact that the responses he obtained are from a shadow network instead of the target network. Our method finishes at link layer and does not give any hints about its existence. As a result, Sniffer Reflector is invisible to the network layer and above.

Sniffer Reflector can be easily combined with other security tools for network forensic purposes. We have developed a Sniffer Reflector prototype by using virtualization technologies. It mainly has three components, Scan Sensor, Reflector and Shadow Network, and the three components cooperate together to protect a target network. Our implementation can be easily deployed in real productive network environments. The experimental results show that Sniffer Reflector is effective and efficient in defending various network scans. We tested our prototype in a local area network. The experiment results show attackers can only receive obfuscated scan responses from the shadow network.

2 Background

2.1 Moving Target Defense

For a long time, the cyber defenses are mainly static. Security analysts follow the conventional process to deploy protections in a productive network, which includes accessing information properties, planning defense strategies, deploying defense technologies and conducting penetration tests. Once the defense is established, it will keep running statically as deployed for a long time. Consequently, adversaries can take time to systematically study the network environment, plan malicious activities, find out a break point and conquer the protected system finally.

Moving Target Defense tries to increase attack bar for adversaries by introducing uncertainties and diversifications to computer systems. It forces adversaries to reprobe, reassess and restudy the target systems. Providing dynamic defense makes MTD a promising research topic in academia. The existing MTD researches can be categorized at two levels: system level and network level. At system level [5], researchers tried MTD on operating system, processor architecture, program runtime environment, application source code and binary code, and so on. At network level [1, 21], MTD was practiced with frequent IP address reshuffling, network port remapping, network configurations adaptation, network topology mutation, dynamic changes on routing information and IP hopping. More MTD research details will be discussed in Section 6.

2.2 Network Scan

Network scan is composed of a set of network activities systematically collecting network information from a target network. Based on different purposes, network scan can be divided into three phases, which are illustrated as follows:

Host Detection The first phase of network scan is host detection. Host detection tries to determine the accessible hosts and their IP addresses in an unknown network. The most used method for host detection is sending an ICMP echo request. If the remote host is alive and the request arrives unblocked, an ICMP echo reply will be answered. When the ICMP reply is received, we know the remote host is on.

Port Discovery Port discovery is the second phase of a network scan, which puts efforts on searching all open network ports of a live host. In general, there are two scans to discover open ports of a host, UDP scan and TCP scan. The most dominant one is TCP scan because most valuable services are implemented in TCP protocol. In this paper, we mainly introduce TCP scan. There are several ways to perform TCP scan: 1) *Full TCP Scan.* scanning host tries to finish the classic TCP *three-way handshake* and establish a full TCP connection with the target host; 2) *Half-Open TCP Scan.* Scanning host sends a SYN segment to a selected port on the target host. If a SYN/ACK reply is received, scanning host knows the selected port is open and the target host is on; if a RST reply is received, that means the port is closed. It does not establish a complete TCP connection; 3) *Stealth Scan.* The word "illegal" means the standard TCP three-way handshake does not consist any of these segments. Attackers forge illegal TCP segments (mainly on control bits) to start a *Stealth Scan* [20], which includes FIN scan, Xmas scan, NULL scan and so on.

Vulnerability Assessment Vulnerability assessment is the last phase of network scan. After identifying the live hosts and open ports, an attacker needs to know further details about a target system, such as OS version, service version,

and configuration, to make further attack strategy. Due to most security vulnerabilities are OS dependent, it is necessary to fingerprint the OS information. Also, the protocol and service versions are valuable to attackers as well, for choosing attack exploits. By figuring out the specific OS and protocol information, the attacker can find out corresponding vulnerabilities and start effective attacks.

2.3 Software-Defined Networking

Software-Defined Networking (SDN) is a hot topic both in academia and industry. The key innovation of SDN is changing the static nature of network devices and making them programmable [11]. SDN separates the control function and forwarding function of a network device, which provides network users abilities to change network behaviors dynamically. Prior to SDN, network devices are designed like "dead" boxes. Each network device will be installed with the same chips and firmware in the factory. Once delivered, network devices can only work in a predefined way. If users want to make some changes to their networks, they have to start over and rebuild the physical network environment, which proves a time-consuming task. With SDN, users can easily modify network packets, change traffic flow directions, form a new network topology, and so on.

Inspired by the defense philosophy of MTD, we consider putting MTD practice into network reconnaissance protection. Since SDN provides the power to modify network behaviors during running, we try to make use of SDN's flexibilities to modify scan traffic flow and provide obfuscated scan replies. If network scan is obfuscated, the following attack steps will not succeed. We believe our method can greatly raise the attack bar for attackers.

3 System Architecture

In this section, we present Sniffer Reflector architecture, which seeks to employ SDN technologies and shadow networks to provide forged scan responses to attackers. Fig. 1 shows the Sniffer Reflector architecture.

As can be seen in Fig. 1, Sniffer Reflector has three main components, *Scan Sensor*, *Reflector*, *Shadow Network*. The wholly protected network environment is called a protection domain. Each protection domain will have one or more protected objects. Each object is connected at least one Reflector, and each Reflector has one or more protected objects connected. In the figure, we have two protected objects in a protection domain, one target host and one target network. The green node stands for a Scan Sensor, which is responsible for monitoring network traffic. The red node represents an attacker node and sends out scan traffic, which is represented by a red curved arrow. The dash line in red shows the scan traffic is visible to the Scan Sensor when going through the Reflector. If scan traffic is detected, the Scan Sensor will send out a reflection message to the Reflector. After receiving the reflection message, the Reflector will manipulate the scan traffic and reflect it to a Shadow Network, where scan responses will be generated. The working process for the Sniffer Reflector architecture could

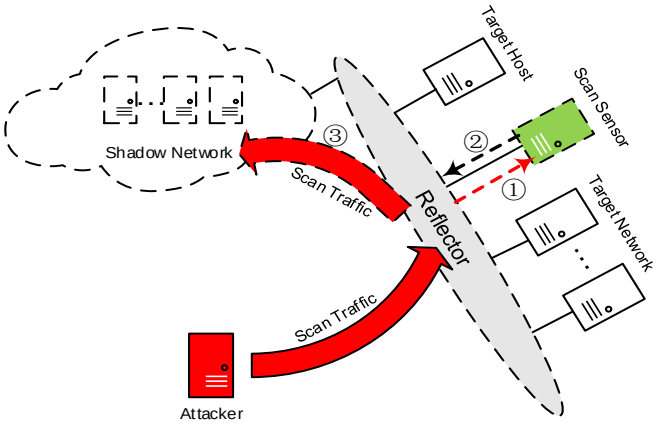

Fig. 1. The Sniffer Reflector Architecture

be illustrated as following steps: (1) Scan Sensor will be monitoring the coming traffic for target hosts and target networks. (2) Once scan traffic is detected, Scan Sensor will send a reflection message to the corresponding Reflector, and the Reflector will follow the message and redirect the scan traffic to Shadow Network. (3) Shadow Network receives redirected traffic from Reflector and disturbs attackers' view by making obfuscated scan responses.

3.1 Scan Sensor

Scan Sensor is a scan detection engine in our architecture, which is responsible for detecting scan activities and generating reflection messages. To perceive scan activities, it is required to observe all the traffic happening in a protection domain. Once scan activities are detected, Scan Sensor will notify Reflector, and Reflector and Shadow Network will cooperate together to reflect scan traffic. Scan Sensor finishes two functions: (1) Detecting scan traffic. Scan Sensor monitor the network traffic and detect the scan activities. It uses scan detection algorithms to find out possible scan traffic. (2) Generating reflection message. Once scan activities are confirmed, Scan Sensor needs to generate a reflection message to notify Reflector. The message contains scan type, scan source, scan target, port info, sensitive level, and action. An example message could be like this: *(TCP Xmas scan, source IP 192.168.2.12, target IP 192.168.1.105, target port 80, high density, reflection flag)*. Reflector will receive the message and execute the reflection action.

3.2 Reflector

Reflector is in charging of reflecting scan traffic. It communicates with the Scan Sensor and the Shadow Network, and executes reflection actions to redirect the scan traffic to the Shadow Network. In a protection domain, each Reflector

will have at lease one protected host or protected network connected. For these connected objects, Reflector should be invisible to them. When there is no scan activity, Reflector acts as a regular network device and provides traffic flow functions to the connected nodes. It maintains the packets switching and traffic management like most network devices do. Consider the scan types may cover ICMP, TCP, UDP and other protocols, when reflecting, Reflector should be able to conduct fine-grained traffic control to identify scan traffic by protocol. Besides, to reflect scan traffic, the Reflector needs to provide at least two routes to change the flow of scan traffic. One route is for normal traffic, which leads to the target host; the other route is for scan traffic, which goes to the Shadow Network.

3.3 Shadow Network

All the scan traffic will be redirected to a Shadow Network. The Shadow Network is an isolated and invisible network, which is composed of shadow nodes. Except for the Reflector, no node in the protection domain is aware of the existence of the Shadow Network. Unless receiving scan traffic from the Reflector, the Shadow Network does not create any network traffic to the protection domain. A protection domain may have multiple Shadow Networks. And, these Shadow Networks can cooperate together to simulate a complex network structure. A typical strategy could be to simulate a replica of the protection domain. The forged network environment has the same look of the protection domain, such as the same number of nodes, the same network topology, and configurations. It can be used to cheat attackers to believe the reached network is the target network. Shadow Network can provide further responses if attackers keep attacking.

4 Design and Implementation

4.1 Design Principles

We have three design principles. First, all three components of Sniffer Reflector should be trusted and not disturbed by attackers. Moreover, the communications within Sniffer Reflector should be invisible to attackers and target nodes. Second, in the protection domain, no network traffic can escape from being monitored, and Scan Sensor can observe any communication happened in the protection domain. Last, once being detected and reflected, attackers cannot bypass the reflection mechanisms of Sniffer Reflector.

4.2 Prototype Implementation

Based on our three design principles, we implemented a prototype of Sniffer Reflector to provide scan protections in a virtual network. The prototype runs a protection domain which consists of two parts, a target network and a Sniffer Reflector framework. As shown in Fig. 2, the target network has two virtual machine nodes and both of them are connected to Reflector. Accordingly, the

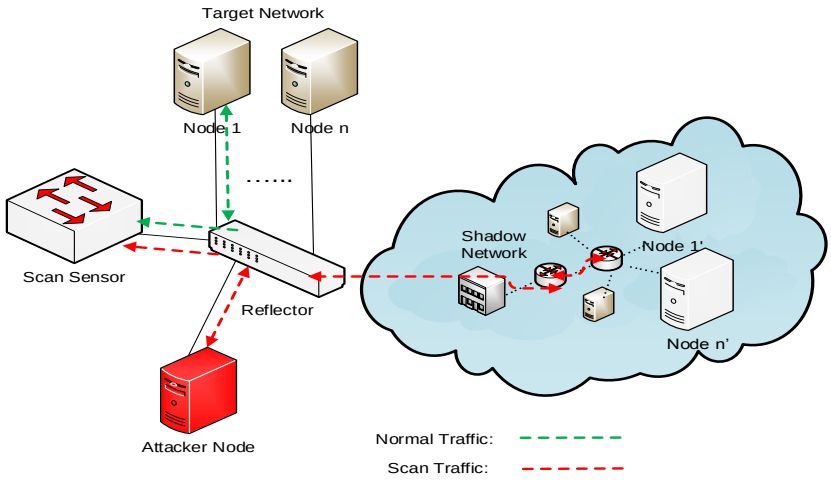

Fig. 2. A Prototype of Sniffer Reflector

Sniffer Reflector framework has three components, just as we discussed in Section 3, Scan Sensor, Reflector and Shadow Network.

We choose to implement the prototype on a virtualization platform, for more and more organizations are transplanting their services to cloud and data center. We employ the Kernel Virtual Machine (KVM) [10] virtualization platform. On KVM platform, 80% of instructions on guest machine can be directly executed on physical CPU, which provides a high efficiency of resource utilization. We customized a virtual switch, VDE_Switch, to run as Reflector. Both KVM guest machines and VDE_Switch are running as user processes at host OS. We illustrate the three components' implementation details as follows.

We modify Snort to implement a Scan Sensor. There are three working modes in Snort, sniffer mode, packet logger mode and NIDS mode. To fit our purpose, we take advantage of NIDS mode. Under NIDS mode, Snort allows security engineers install a set of security rules to detect suspicious events over network. When a suspicious event is detected, an alert information will be generated and written to the log file. We located the source code where the alert information is generated and inserted an additional module. The module is responsible for looking scan alert patterns and generating reflection messages. The message includes scan type, scan source, scan target, port info, sensitivity level and action. Moreover, we developed a communication module sending reflection messages to Reflector. More details about message content and scan detection policies will be given in Section 5.

The implementation of Reflector is based on a virtual switch, VDE_Switch [3]. The basic functionalities of VDE_Switch are receiving, processing and forwarding network packets for the connected nodes. Each connected node will be assigned a dedicated port number for packets switching purpose. We modified VDE_Switch and inserted it an extra layer for packet checking function. When

receiving reflection messages, VDE_Switch will translate the messages to reflection rules. Reflection rules contain the characteristics of scan traffic, such as scan source IP and protocol information. All the reflection rules will be stored in a *Refl_Rules_List*. All the packets going through VDE_Switch will be checked by the list. If any traffic got a match, it will be reflected. The modified VDE_Switch is capable of verifying packet headers from link layer to transport layer, which enables VDE_Switch to identify traffic flows by protocol. That helps us achieve a fine-grained traffic control. To implement secret channels for communications happened within Sniffer Reflector, we designed two reserved port numbers in VDE_Switch. These two reserved port numbers are dedicated to Scan Sensor and Decoy Network. For each reserved port, we set a switch flag. When there is no scan traffic, the switch flag will be set off and the reserved port is blocked; when there is scan traffic detected, the switch flag will turn on and scan traffic will be reflected through reserved ports. The detailed reflection decision is demonstrated in Algorithm 1.

Algorithm 1 Scan traffic reflection decision

Require: The current packet p; The switch flag sf; The set of reflection rules: $Refl\_Rules\_List$; The port used by Shadow Network: SN_port;

1: **for** every packet p **do**
2: **if** $((p.source\_IP, p.protocol) \in [Refl\_Rules\_List])$ && (sf is on) **then** send p to SN_port;
3: **end if**
4: **end for**

Shadow Network is implemented with virtualization technologies as well. We use KVM virtual machines to simulate nodes of Shadow Network. These virtual machines are connected with virtual network devices to construct a network structure. On each virtual machine, we deploy the corresponding OS and network services, which are intended to provide obfuscated scan responses to attackers. The virtual machines are in full control and can be configured with arbitrary network parameters to simulate desired network behaviors. The reason why we use real virtual machines instead of virtual honeypots/honeynets in Shadow Network is virtual machine can provide a fully responsible OS and TCP/IP stack for attackers. Some light-weight virtual honeynets, like honeyd [17], can be used to simulate a network, but their responses can be easily detected by attackers.

5 Evaluation

We evaluate our prototype in a virtual LAN 192.168.1.0/24. The entire virtual network facility is deployed on a physical machine, and all other network nodes and devices are running as virtual machines. Our testbed was deployed on Intel Core i7-3370 3.4Ghz processor with 16GB RAM. The host machine is running

CentOS 6.0 with kernel 2.6.32 x86_64 and qemu-kvm-0.15.1. Similarly, all other guest OSes are running CentOS Linux as well. The Scan Sensor virtual machine is running Snort 2.9.7.5 for scan detection. The Shadow Network is deployed with two virtual machines as shadow nodes to provide forged scan responses. The Reflector employs VDE_Switch 2.3.2.

```
alert tcp any any -> $HOME_NET any (msg:"TCP SYN"; flow:stateless; flags:S;
detection_filter:track by_dst, count 100, seconds 5; sid:1000001;rev:1)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL"; flow:stateless;
ack:0; flags:0; seq:0; reference:arachnids,4; classtype:attempted-recon; sid:623; rev:6;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";
flow:stateless; flags:SF,12; reference:arachnids,198; classtype:attempted-recon; sid:624; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN XMAS"; flow:stateless;
flags:SRAFPU,12; reference:arachnids,144; classtype:attempted-recon; sid:625; rev:7;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap XMAS";
flow:stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:1228;
rev:7;)
```

Fig. 3. Scan detection rules in Snort

As can be seen in Fig. 2, we have five nodes in prototype. The host is running with 192.168.1.107. Two other nodes, 192.168.1.153 and 198.168.1.154, run Linux OS as normal nodes of the protection domain. Node 192.168.1.153 is running with open ports 22, 23, 80, 111 and 443; node 192.168.1.154 is running with open ports 22, 80 and 111. Shadow Network has two shadow nodes, which both run Windows XP systems with the same IP addresses as normal nodes, 192.168.1.153 and 192.168.1.154. Both two shadow nodes are configured to open ports, 53, 135, 139, 445 and 3389. An attacker node runs with IP 192.168.2.1 in subnet 192.168.2.0/24. The attacker node is used to send scan traffic and collect scan results. Besides, Snort and VDE_Switch are running as user-space processes on host OS.

Scan detection rules were configured to detect TCP SYN flood and TCP stealth scans on Snort. The content of rules is shown in Fig. 3. The first rule detects TCP SYN flood scan, which is defined as X TCP SYN requests in Y time period. In our evaluation, we define TCP SYN flood as any TCP connection requests sent more than 100 times in 5 seconds (this assumption can be modified to accommodate different detection scenarios). Then, the following detection rules give the details of TCP stealth scans, including FIN scan, NULL scan, and XMAS scan.

We employ nmap to generate scan traffic. To demonstrate the effectiveness of our protection, we design three types of scans, SYN scan, Xmas scan and version detection scan. See the nmap commands as follows:

1. nmap -sS 192.168.1.0/24
2. nmap -sX 192.168.1.153/154
3. nmap -A -T4 -F 192.168.1.153/154

The first nmap command executes a TCP SYN scan for LAN 192.168.1.0/24. TCP SYN scan probes remote ports through a full TCP three-way handshake. Since TCP SYN scan is the most typical scan over the internet, most system logs will default capture TCP SYN scan. Then, the second scan command starts a TCP Xmas scan on two hosts 192.168.1.153/154. TCP Xmas scan is one of the TCP stealth scans, which is usually blocked by most firewalls. The third scan in our evaluation is a composite scan, which contains a series of scan activities. The -A option means the scan command will run OS detection, version detection, script scanning, and traceroute. Nmap will match the scan results with its fingerprint database and estimate the OSes and version information of the scanned hosts. We execute the scan commands in two rounds. The first scan round is executed without the protection of Sniffer Reflector; the second round is scanned with the protection of Sniffer Reflector. Then, we compared the scan results, which show our prototype is effective and efficient in defending various network scans.

Table 1. Scan results comparisons on 192.168.1.0/24

	Without Sniffer Reflector Protection	With Sniffer Reflector Protection
TCP SYN scan nmap -sS 192,168.1.0/24	192.168.1.107 is up 0.00015s latency Not shown: 998 closed ports Port State Service 22/tcp open ssh 111/tcp open rpcbind 192.168.1.153 is up 0.00029s latency Not shown: 995 closed ports Port State Service 22/tcp open ssh 23/tcp open telnet 80/tcp open http 111/tcp open rpcbind 443/tcp open https 192.168.1.154 is up 0.00023s latency Not shown: 997 closed ports Port State Service 22/tcp open ssh 80/tcp open http 111/tcp open rpcbind	192.168.1.107 is up 0.00032s latency Not shown: 998 closed ports Port State Service 22/tcp open ssh 111/tcp open rpcbind 192.168.1.153 is up 0.015s latency Not shown: 995 closed ports Port State Service 53/tcp open domain 135/tcp open msrpc 139/tcp open netbios-ssn 445/tcp open microsoft-ds 3389/tcp filtered ms-term-serv 192.168.1.154 is up 0.015s latency Not shown: 995 closed ports Port State Service 53/tcp open domain 135/tcp open msrpc 139/tcp open netbios-ssn 445/tcp open microsoft-ds 3389/tcp filtered ms-term-serv

Table 1 shows the scan results on subnetwork 192.168.1.0/24. The first column is the nmap scan command. The second and third columns show the scan

Table 2. Scan result comparisons on a single host

	Without Sniffer Reflector Protection		With Sniffer Reflector Protection	
	for 153	for 154	for 153	for 154
TCP Xmas scan nmap -sX 192.168.1.X	192.168.1.153 is up 0.00029s latency 995 closed ports Port State Service 22/tcp o/f ssh 23/tcp o/f telnet 80/tcp o/f http 111/tcp o/f rpcbin 443/tcp o/f https scanned in 14.26 sec	192.168.1.154 is up 0.00028s latency 997 closed ports Port State Service 22/tcp o/f ssh 80/tcp o/f http 111/tcp o/f rpcbin scanned in 14.25 sec	192.168.1.153 is up 0.015s latency 999 closed ports Port State Service 3389/tcp o/f m-t-s scanned in 14.28 sec	192.168.1.154 is up 0.0020 latency 999 closed ports Port State Service 3389/tcp o/f m-t-s scanned in 14.25 sec

results without and with the protection of Sniffer Reflector. As can be seen from the second column, scan results return all the live nodes in 192.168.1.0/24. For each live node, the results show scan latency, how many ports are closed, and a list of open port/state/service information. When there is no Sniffer Reflector, the scanner is able to receive network information from the protection domain. From the returned IP and open ports information, we can see the attacker can easily collect network information from the target network. The third column lists the scan results under the protection of Sniffer Reflector. The listed open ports match with the virtual machines we configured as shadow nodes, which shows the scanner actually obtains the network information from shadow network. Comparing the scan results of two columns, we can tell there is a difference in scan latency. With Sniffer Reflector, the average latency of simulated decoy service is 0.015s, which is higher than the latency of real nodes (0.00015s). The delay is probably because the reflection is implemented in software, and VDE_Switch needs to reflect scan traffic packet one by one.

Table 2 illustrates the TCP Xmas scan results on two hosts 192.168.1.153/154. The nmap scan command is in column one, and the rest columns demonstrate the scan results received with/without the protection of Sniffer Reflector. Different from SYN scan, the port state information in Xmas scan results shows as "o/f", representing "open or filed". The scan results show, when Sniffer Reflector is on, both two scans are returning simulated results from shadow nodes. That presents, the scan traffic is reflected and the scan results come from shadow network. And, we also got the scan results from the nmap composite scan. The results show that, with the protection of Sniffer Reflector, the composite scan can also be detected and reflected in our framework.

6 Related Work

MTD researches are quite popular in recent years. Many MTD methods have been proposed to mitigate security threats. The existing researches on MTD can be divided into two levels, system level and network level. The network level MTD solutions [1, 4] include IP address reshuffling, network configuration randomization, and so on. These methods all tried to obfuscate attackers at

network level. The system level MTD methods cover platform [15], runtime environment [19], and software applications [18]. Kewley et al. [9] proposed to reduce network attacks using dynamic network reconfigurations. They tried to force attackers to use the outdated network configurations to prepare an attack. A live IPv6 version MTD was implemented by Groat et al. [7]. By using DHCPv6 protocol, a hidden connection is built between IPv6 address and DHCP identity, which could be used to protect sensitive communications in government or confidential organizations. Unlike our method, these methods are still in the phase of prototype and far from mature for deployment.

Our work is also motivated by SDN researches. OpenFlow [14] is the most widely accepted SDN protocol. It is designed to dynamically change network behaviors by using controllers and switches. Controllers are responsible for managing the attached switches and deciding the flow tables of switches, and switches are using flow tables to forward network traffic. Lara et al. [12] provided a survey on innovations of OpenFlow, which has been used in network management, traffic analysis, fault tolerance, security, and many other areas. SnortFlow [22] is proposed to integrate intrusion prevention systems (IPS) with OpenFlow on a cloud. The basic idea is using the IPS alerts to change cloud behaviors. However, due to the heavy structure of cloud, It is impractical to provide timely reactions on security events. In comparison, Sniffer Reflector provides instant changes on reflection actions and can be deployed with no difficulties in a real network.

7 Conclusion

The relatively static nature of today's computing and network systems has led to an information asymmetry between the attackers and defenders. To break this asymmetry, in this paper, we present Sniffer Reflector, a new Moving Target Defense method against network reconnaissance with Software-Defined Networking, to obfuscate the attackers' view of the target network information. We have designed and implemented a prototype in a virtual local area network. Our experimental results with various scan activities show that Sniffer Reflector is effective and efficient.

Acknowledgements

This work was supported in part by the NSF Grants CCF-1320605 and CNS-1223710, and ONR Grants N00014-13-1-0175 and N00014-16-1-2265.

References

1. Al-Shaer, E.: Toward network configuration randomization for moving target defense. In: Moving Target Defense, pp. 153–159. Springer (2011)
2. Allman, M., Paxson, V., Terrell, J.: A brief history of scanning. In: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. pp. 77–82 (2007)

3. Davoli, R.: Vde: Virtual distributed ethernet. In: 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. pp. 213–220. IEEE (2005)
4. Dunlop, M., Groat, S., Urbanski, W., Marchany, R., Tront, J.: Mt6d: A moving target ipv6 defense. In: Military Communications Conference, 2011. IEEE (2011)
5. Evans, D., Nguyen-Tuong, A., Knight, J.: Effectiveness of moving target defenses. In: Moving Target Defense, pp. 29–48. Springer (2011)
6. Gadge, J., Patil, A.A.: Port scan detection. In: Networks, 2008. ICON 2008. 16th IEEE International Conference on. pp. 1–6. IEEE (2008)
7. Groat, S., Dunlop, M., Urbanksi, W., Marchany, R., Tront, J.: Using an ipv6 moving target defense to protect the smart grid. In: Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES. pp. 1–7 (Jan 2012)
8. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.): Moving Target Defense - Creating Asymmetric Uncertainty for Cyber Threats. Advances in Information Security, Springer (2011)
9. Kewley, D., Fink, R., Lowry, J., Dean, M.: Dynamic approaches to thwart adversary intelligence gathering. In: Proc. of DARPA Information Survivability Conference & Exposition II,. pp. 176–185 (2001)
10. Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A.: kvm: the linux virtual machine monitor. In: Proc. of the Linux symposium. vol. 1, pp. 225–230 (2007)
11. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. p. 19 (2010)
12. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using openflow: A survey. Communications Surveys & Tutorials, IEEE 16, 493–512 (2014)
13. Liao, H.J., Lin, C.H.R., Lin, Y.C.: Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications 36, 16–24 (2013)
14. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2), 69–74 (2008)
15. Okhravi, H., Comella, A., Robinson, E., Haines, J.: Creating a cyber moving target for critical infrastructure applications using platform diversity. International Journal of Critical Infrastructure Protection 5(1), 30–39 (2012)
16. Panjwani, S., Tan, S., Jarrin, K.M., Cukier, M.: An experimental evaluation to determine if port scans are precursors to an attack. In: Proc. International Conference on Dependable Systems and Networks, DSN 2005. pp. 602–611. IEEE (2005)
17. Provos, N.: Honeyd-a virtual honeypot daemon. In: 10th DFN-CERT Workshop, Hamburg, Germany. vol. 2, p. 4 (2003)
18. Rinard, M.: Manipulating program functionality to eliminate security vulnerabilities. In: Moving Target Defense. Springer (2011)
19. Shacham, H., Page, M., Pfaff, B., Goh, E.J., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: Proc. of the 11th ACM Conference on Computer and Communications Security. pp. 298–307 (2004)
20. Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical automated detection of stealthy portscans. Journal of Computer Security 10(1/2), 105–136 (2002)
21. Wang, H., Jia, Q., Fleck, D., Powell, W., Li, F., Stavrou, A.: A moving target ddos defense mechanism. Computer Communications 46, 10–21 (2014)
22. Xing, T., Huang, D., Xu, L., Chung, C.J., Khatkar, P.: Snortflow: A openflow-based intrusion prevention system in cloud environment. In: Research and Educational Experiment Workshop (GREE), Second GENI. pp. 89–92. IEEE (2013)