EAI Endorsed Transactions
on Security and Safety

Research Article **EAI.EU**

Shoal: A Network Level Moving Target Defense Engine with Software Defined Networking

Li Wang[*]

College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA, USA

Abstract

Moving Target Defense (MTD) was proposed as a promising defense paradigm to introduce various uncertainties into computer systems, which can greatly raise the bar for the attackers. Currently, there are two classes of MTD research over computer system, system level MTD and network level MTD. System level MTD research introduces uncertainties to various aspects of computer systems; while network level MTD research brings unpredictability of network properties to the target network. A lot of network level MTD research has been proposed, which covers various aspects of computer network. However, the existing MTD approaches usually target on one aspect of computer network, and most of them are designed against a certain network security threat. They can hardly defend against complex attacks or provide complicated protections. In this paper, we propose Shoal, a Moving Target Defense engine with multiple MTD strategies over SDN networks. By applying hybrid and multiple network level MTD methods, Shoal is capable of providing complicated protections and defending advanced attacks. We evaluate Shoal in two advanced protection scenarios, moving target surface and Crossfire attack. The evaluation results, in term of security effectiveness and performance cost, show the protection provided by Shoal's hybrid MTD methods is effective and the performance cost is relatively low.

Received on 25 March 2021; accepted on 09 May 2021; published on 01 June 2021

Keywords: Moving Target Defense, SDN, security strategy, network security

doi:10.4108/eai.1-6-2021.170011

1. Introduction

Current computer systems are usually built in a static nature. The static nature means the running environment of a computer system will not be changed once established. Generally, a computer system's running environment includes an operating system, a set of software stacks, a fixed IP address (if the computer system is on the Internet), the related configurations, and the corresponding services. For example, to run a web server, we need an operating system, Linux or Windows, to help manage computer hardware; we need a set of software stacks, like C standard library [1] and other intermediate APIs, to provide low-level software supports to run web applications; we need a web server program, like httpd and HTML pages, to provide web service content. Also, we need a fixed IP address for the web server, which guarantees the server is reachable through the Internet. When the web server is running, we can hardly change its operating system or C libraries because the upper-level web server program is relying on the running environment to provide stable services to users. Similarly, under the current computer network architecture, we are not able to reconfigure the network parameters of the web server due to users rely on the parameters to visit its service. In summary, the static nature makes computer systems easy to operate and manage.

However, the static nature makes current computer systems easy targets of cyber attack as well [2, 3]. Since computer systems will not be changed at run time, attackers are able to spend as much time as they can to find an effective way to compromise a target machine. There are a lot of publicly-available tools for attackers to identify the vulnerabilities of a target system. For instance, an attacker can use Nmap [4] to scan the target network. From the scan results, the attacker is

[*]Corresponding author. Email: lzw158@ist.psu.edu

able to figure out how many live nodes in the network, what services are running, what are the service versions, and so on. With the valuable information, the attacker is capable of inferring the vulnerabilities information and make a detailed plan for a future attack. Although existing security tools are created to provide protections on various attack activities, attackers can still detect, study, and finally bypass them because the current security tools are not faultless.

Consequently, Moving Target Defense (MTD) [5] was proposed as a promising defense paradigm to brake the static nature of current computer systems. MTD tries to introduce diverse uncertainties to make a computer system's running environment dynamic and unpredictable. In an MTD protected environment, a target computer system is able to provide different behaviors and responses when facing suspicious visits [6]. It is not easy for an attacker to collect effective information for attacking use. Therefore, the attacker may need to assess the target system over and over again, which greatly increases the bar for attackers. The existing MTD research can be categorized into two classes, system level MTD and network level MTD [7]. System level MTD research introduces uncertainties to various aspects of computer systems; while network level MTD research brings unpredictability of network properties to the target network.

Network level Moving Target Defense (MTD) methods have been proposed to bring uncertainties to the network properties. However, the existing MTD approaches usually target on one aspect of the network [8, 9], and most of them are designed against a certain network security threat. For example, to mitigate the network reconnaissance attack, host address randomization [10] is proposed to transparently change the host's IP address. By dynamically changing the host IP address, host randomization MTD method tries to achieve a high unpredictability rate for the attackers to identify targets. Host randomization can help to delay the network intelligence collection phrase in an attack, which is an early stage in an attack. DNS service mutation is proposed [11] [12] to add the authentication and authorization process to the protected servers, which brings an additional mapping layer between the DNS server and target servers to help reducing DDoS attacks. DNS service mutation can help to mitigate the simple DDoS attacks, but it may not be able to help advanced DDoS attacks, such as Crossfire attack [13]. Unlike simple DDoS attacks, Crossfire attack does not send attack traffic to the target network directly. Instead, the attack traffic is sent to the related network area, such as neighbor network, and the attack traffic will aggregate and congest at target links, which are critical to the target network. As a result, the target network communication will be affected. Since Crossfire attack does not attack the target network directly, the protections deployed at the target network usually cannot detect it. Most existing MTD methods can help defending relatively simple security threats. However, they can hardly defend against more complex attacks or provide complicated protections. Besides, the existing MTD methods are evaluated individually. Seldom is there specific work explaining how the individual MTD schemes can be combined together against the advanced attacks and what are the effectiveness and performance results of applying the hybrid MTD methods.

Consider the limitation of the existing MTD work, we present Shoal, a Moving Target Defense engine with SDN, to provide investigations on hybrid MTD methods. SDN is a new network paradigm, which enables the flexible programming capability of network devices. Through SDN technologies, network researchers and users are able to program network behaviors on the fly. More details of SDN will be given in the next section. Shoal provides nine individual network level MTD strategies, which can be combined together to provide different defense choices. We want to achieve a comprehensive Moving Target Defense protection which is capable of defending complicated and advanced security threats. Our Contributions are listed as follows:

- *Propose a practical Moving Target Defense Engine with SDN.* To the best of our knowledge, Shoal is the first comprehensive network-level MTD engine over virtualized networks, which provides a batch of MTD strategies for different aspects of the target network. These MTD strategies can be used together to provide multiple defense choices and satisfy different protection purposes.

- *Provide a batch of network-level MTD strategies for defense choices.* Based on the existing MTD philosophies, we provide a series of network-level MTD strategies. including *network configuration mutation*, *network route mutation*, *network topology mutation*, *network address shuffling*, *network traffic reflection*, *network traffic manipulation*, *network diversification*, *network elements migration*, and *DNS service mutation*. These strategies are summarized from real attack scenarios, which can help to disrupt attacking activities over networks. They can be used individually or together for different protection purposes.

- *Build up a Shoal prototype to show the effectiveness of our method.* We build up a Shoal prototype with the provided MTD strategies in a virtual network environment. The MTD strategies are demonstrated in different security scenarios, which show how the provided MTD strategies can be applied in real protections.

- *Investigate using hybrid MTD strategies to provide complicated protections and defend complex attacks.* We evaluate Shoal in real protection scenarios. The evaluation result includes security effectiveness and performance cost. We use hybrid MTD strategies provided by Shoal to practice Moving Target Surface [14] [15] and defend Crossfire attacks [13]. The evaluation result shows the hybrid MDT strategies are effective and useful in the above scenarios.

Unlike existing network-level MTD research, Shoal offers a set of comprehensive network-level MTD strategies and provides universal protection for the target networks. With different MTD strategies, defenders can dynamically achieve different protection purposes and protect network properties from various security threats. It is designed to fit diverse security protection scenarios against various attacks over different platforms, including cloud, data center, and other virtualization environments. We employ the combined MTD strategies of Shoal to provide complicated protections against complex attacks.

The remaining of the paper is organized as follows. We introduce the Moving Target Defence background in Section 2. In Section 3, we present the system design of Shoal. The network-level MTD strategies will be given in Section 4. The Shoal prototype will be illustrated in Section 5. We talk about the evaluation in Section 6. Section 7 presents the possible costs introduced by various MTD strategies. We conclude the paper in Section 8.

2. Background and Related Work

Software-defined networking (SDN) is a new network paradigm, which decouples the network control plane from the data forwarding plane. It has gained much attention in both academia and industry [16]. By decoupling the control logic from the closed and predesigned network devices, SDN enables the flexible programming capability of network devices. Through SDN technologies, network researchers and users are able to program network behaviors on the fly. In conventional network architecture, network devices can only work as they were manufactured by network equipment vendors. Once distributed by the vendors, all the traffic control and forwarding functions on the network devices are not changeable. With SDN, the traffic control functions and traffic forwarding functions are divided as *control plane* and *data plane* logically, which could be put onto different network entities. In SDN networks, the data plane is still embedded in network devices, while the control plane is fulfilled on a different network entity, called *SDN controller*. SDN controllers are responsible for maintaining the network flow rules and managing the network running behaviors. There are two kinds of programming interfaces in SDN architecture, northbound interfaces, and southbound interfaces. Northbound interfaces provide the upper level network applications to communicate with SDN controllers, while the southbound interfaces are used for communications between SDN controllers and network devices. The separation of *control plane* and *data plane* provides an unprecedentedly powerful and flexible network infrastructure for network behavior control. A lot of network innovations have been created with SDN technologies.

Many security researchers tried to take advantage of SDN technologies to devise new network security solutions. A lot of efforts [17–21] have been put on providing better network security services over software-defined networks. The existing security solutions are either implemented at the centralized controllers or the distributed inline network devices. For example, FRESCO [20] is a modular security service in software-defined networks. Aided by OpenFlow protocol, FRESCO achieves a security application framework that enables security researchers to establish a security application with several composable modules. Similarly, some researchers tried to put some security functions on network devices for performance considerations. OpenFlow Extension Framework (OFX) [21] uses the processing power of network switches to run security applications, which means the security functions will be deployed on network switches directly.

SDN technologies are also used in many network level MTD researches. The solutions [10, 11, 22, 23] to deploy MTD over network include IPv6 based network methods, network configuration randomization methods, and software defined networking methods. A framework MUTE [23] was brought up by Ehab et al. to mutate network configuration randomly and dynamically. MUTE is a comprehensive method for network MTD solution, which not only makes use of IP randomization strategy but also employs network configuration information. MUTE can dynamically change the network configuration to diversify the network behavior. At the same time, normal network communication can still flow fluently. Jafarian and Duan [10] employed Software Defined Network (SDN) OpenFlow protocol to reshuffle IP addresses frequently for online servers. Their reshuffling method is claimed to reach a high unpredictability rate and fast transparent IP mutation. By hiding the authentic IP address of the protected server, attackers can hardly initialize an attack from an outside network environment. Kewley et al. [24] proposed to reduce network reconnaissance attacks by obfuscating network packet headers. When attackers receive network sniffing responses, the network properties obtained from packets are not correct. Using the fake network properties will lead to an unsuccessful attack. A live IPv6 version MTD [25] was implemented by Dunlop et al. 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. Similarly, IPv6 is also used in Sherburne's method [22], which deploys moving target IPv6 defense to protect Low-Powered Wireless Personal Area Networks (6LoWPAN). Peng and Zou [8] proposed an MTD method to diversify the attack surface for cloud infrastructure. With the dynamic nature of MTD, cloud infrastructure presents a heterogeneous and dynamic attacking surface, which tremendously increases attacking difficulties for attackers. Besides, the overlay network solution is employed for MTD as well. Li and Reiher et al. devised a method to break the malware spreading vector by using a redundant overlay network [26] to distribute security updates dynamically. However the existing MTD methods are designed against a certain network security threat or only target on one aspect of the network. They can hardly defend against the advanced attacks or provide complicated protections. Virtual network adaptation had been proposed in the distributed computing environment [27] for virtual network management uses. VNET [28] is a simple layer two virtual network tool which supports arbitrary topologies and routing and adaptive control of the overlay network. VNET claims to provide arbitrary topology and forward rules on the fly. However, VNET is relying on an overlay network and built on top of the ethernet. The implementation of VNET decides it is not a "real" network adaptive engine and cannot be applied in a large scale network. For example, VNET provides little support to modify network attributes and manipulate network traffic. Besides, VNET is designed for network management use instead of security use. Compared with VNET, Shoal is a universal network adaptive engine and supports various security protection policies.

However, the existing network-level MTD methods are limited in its functions of bringing uncertainties to a network. Very few of them can cover all the aspects of a network. Shoal is able to provide a comprehensive MTD protection on the target network. And, Shoal provides a set of MTD strategies to defend various network attacks, which provide enough flexibilities for users to choose in different scenarios.

3. Design and Architecture

3.1. Overview

Figure. 1 shows the Shoal architecture. As can be seen from the figure, there are mainly three kinds of components in Shoal architecture, *MTD Strategy Center*, *Enforcement Agent*, and *Target Network*. The three components cooperate together to provide a comprehensive MTD protection for the target network. From top to bottom, the first component is MTD Strategy Center. MTD strategy center is in charge of storing and distributing the MTD strategies, and the specific MTD strategies represent how the target network properties can be adjusted to defend against the possible threats. In each protected target network, there is an Enforcement Agent who is responsible for communicating between the MTD Strategy Center and Target Network. For one side, the Enforcement Agent is responsible for receiving the specific MTD strategies; for the other side, the Enforcement Agent is responsible for translating the MTD strategies into concrete MTD rules, which can be used to instruct the target network to adjust its network behaviors. The last level is the Target Network, which represents the protected network properties. These protected networks include various network nodes, shadow networks, DNS server (used in MTD strategies involving DNS service), DHCP server (used for IP reshuffling service), and network reflector (used for network reflecting service).

3.2. MTD Strategy Center

MTD Strategy Center is responsible for driving the entire engine. It manages and stores all the Moving Target Defense strategies, which record how the target networks can be protected with specific network level adjustments. The existing MTD adaptation models include the proactive adaptation model and reactive adaptation model [29]. Depending on the specific adaptation model, the MTD Strategy Center initializes an MTD defense activity periodically or based on an attack alert. For example, a security analyst may sniff an unexpected network scan in the target network and send out a malicious network scan alert to the MTD Strategy Center. As a result, MTD Strategy Center may choose a corresponding scan mitigation strategy, such as IP reshuffling strategy or network reflecting strategy, and distribute it to the Enforcement Agent. The MTD Strategy Center is communicating with multiple Enforcement Agents for different target networks.

3.3. Enforcement Agent

The Enforcement Agent is in charge of translating and distributing MTD strategies and have these strategies enforced at individual network devices and nodes. When the Enforcement Agent receives an MTD strategy from MTD Strategy Center, it retrieves the target network information and interprets the MTD strategies into fine-grained MTD rules. Each MTD strategy may involve multiple network devices and nodes, and the MTD rules will be executed at the related network devices and nodes. For example, to mitigate a Crossfire Attack [13] on certain vulnerable links, the network

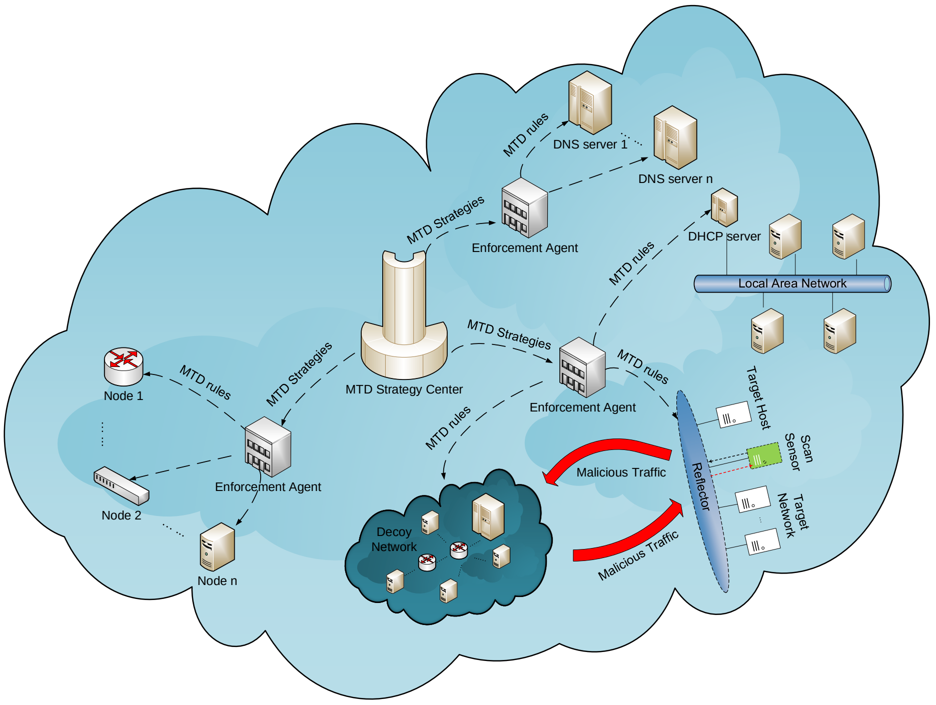

Figure 1. Shoal design and architecture

operators can choose alternative routing paths to adjust the excessive network traffic on certain vulnerable links. Depending on the under layer network structure, the related network boxes will be picked up and their routing rules will be changed. Consequently, the excessive network traffic flowing through the vulnerable links will be guided to take other links.

3.4. Target Network

Target Network is a network area that is being protected within the Shoal architecture. It is connected to the Enforcement Agent and equipped with various adapted network services, such as SDN controller services and network management services. Through the adapted network services, Target Network receives and executes MTD rules from Enforcement Agent to adjust the corresponding defense strategies. Depending on the protection goal, a Target Network could be a public subnetwork, an entire enterprise network, or an individual network application server. For example, assume we want to protect a subnetwork from being attacked by network reconnaissance attacks, we can deploy the protected subnetwork as a Target Network. The protected subnetwork can be connected with SDN compatible network devices or adapted network management services, which provide network reflector defense or network address randomization defense against unexpected network reconnaissance activities. Similarly, an individual network application server can also be protected in a target network. More details will be given in the MTD strategies section.

3.5. MTD Rules

MTD rules are the specific instructions enforced at the Target Network when adjusting the MTD defense strategies. For example, if the Target Network is being protected with IP randomization strategy, the MTD rules could be the related DHCP commands to flush the IP addresses assigned in the Target Network. Also, when protecting the Target Network with shadow networks, the MTD rules will be the instructions to modify the shadow network, which could disturb the potential attackers' view and hide the valuable protected network nodes. More details can be located in the following sections.

4. MTD Strategies

In this section, we present a series of network-level MTD strategies, including *network configuration mutation*, *network route mutation*, *network topology mutation*, *network address shuffling*, *network traffic reflection*, *network traffic manipulation*, *network diversification*, *network elements migration*, and *DNS service mutation*. These MTD strategies decide how we are going to adjust the network behaviors to fight against attacking activities. Our MTD strategies cover almost all the aspects of a network or network nodes. Taking advantage of these strategies, defenders are given multiple choices to deal with various attacking scenarios. Moreover, these strategies can be easily implemented over virtualized networks. Next, we explain each strategy in detail and illustrate its applications under specific attacking scenarios.

4.1. Strategies Introduction

Network Configuration Mutation. Network configuration includes network address, access control permissions, network node configurations, and so on. Network configurations are effective information for attackers. Assume an attacker wants to compromise a target system, he needs to obtain at least its network address and access permissions. If we periodically change a system's IP address (in this case, the system should be able to notify normal users of the newest IP addresses), the attacker cannot obtain a guaranteed visit on it, which obviously creates difficulties for the attacker. Network Configuration Mutation can be used to protect target-based attacks, network scans, and large-scale malware propagations.

Network Route Mutation. Network route is a network path through which the network packets travel through the networks. The route includes all the network devices the network packets may pass by. Here, the network device is defined as any network box which are able to transfer and handle the network packets from source to destination. Through network route information, the attacker is able to figure out the network topology information and other valuable network intelligence. Consider the network route may expose critical information to the attackers, we can use Network Route Mutation to change the network route information to defend any network route related attacks.

Network Topology Mutation. Network Topology Mutation is used to disturb an attacker's view on a target network. As an important network character, network topology records the physical "shape" of the network. Through a network's topology, an attacker can easily infer valuable information for planning an invasion. For example, given a network with a *star* topology, the attacker will probably choose to take over the "hub" node first, for it has the most flexible connections with all other nodes. We can use Network Topology Mutation in defending network reconnaissance, stepping stone attacks, and large-scale malware propagations.

Network Traffic Manipulation. Network Traffic Manipulation is used to investigate and modify the network traffic content. We can modify a packet's content during its transmission for various protection purposes. For example, we can modify the *dest* IP field in a packet header to change its destination. Or we can modify the TCP flag fields to disturb a suspicious TCP connection. Through Network Traffic Manipulation, we can achieve network reflection, network honeypot, and other special protections.

Network Diversification. Network Diversification means we adopt a target network to present an obfuscated and dynamic look to the outside. Consider virtualization technologies give us the power to simulate numerous virtual nodes on one physical machine, we can obfuscate the target network with virtual nodes, which may provide different network and system views to the attackers. For example, the current target network contains ten real nodes running Linux operating system. To provide obfuscated network view, we can put ninety virtual nodes in the target network, and the ninety virtual nodes are running with Windows, FreeBSD, and Linux operating systems. When an attacker scans the network, he will get a result of one hundred nodes, which provides a combined network view containing Linux, Windows, and FreeBSD systems. It can greatly increases the workload for the attacker. We can use Network Diversification [14] to defend against network scan, advanced persistent threat, malware propagation, and vulnerability specific attacks.

Network Elements Migration. Network Elements Migration means dynamically rearrange the network resources in a virtual network environment. It can be used to change network structure and adjust the network configuration. The migrated network elements cover network nodes and network devices. The network node migration is supported by the virtualization platform, which can transfer virtual machines from one physical machine to another physical machine during runtime. The network devices migration, such as virtual switches and routers, is achieved by changing their positions in the virtual network. The configuration information in the network devices can also be adjusted during the migration process. We can use Network Elements Migration to defend botnet, reduce the security risk brought by an infected node, and malware propagations.

Network Address Shuffling. Network Address Shuffling can help the protected network from being disturbed by stealthy scanning, worm propagation, and scanning related attack. We employ IP address shuffling to protect the target network. There are several ways to implement IP shuffling. We take Jafarian's method [10] to deploy the real IP and virtual IP mapping at the SDN controllers. With the mapping at SDN controllers, we may avoid frequently changing the host network configurations and services. Also, we can instruct the IP reshuffling time cycle at the SDN controller and dynamically change the virtual IP addresses of the target network.

Network Traffic Reflection. Network traffic reflection is an effective MTD strategy to disturb the malicious network reconnaissance activities. We use sniffer reflector [30] to provide an obfuscated network view to the attackers. With the mutable shadow network, we can provide arbitrary network structures and services to disturb attackers. With Sniffer Reflector, attackers can only collect the network information from the shadow network instead of the Target Network. As a result, attackers can hardly initialize an effective attack on the target network.

DNS Service Mutation. DNS service can also be used to provide different MTD choices. We can make use of DNS service to provide a middle layer for the visitors of the protected servers [11] [12] [31]. For example, the same domain name can be resolved to different IP addresses through DNS services, and this can help direct and balance the coming visits to the protected services. Also, DNS service can also work with proxy servers [11], which provide a dynamic and distributed relay service to the protected services. In other scenarios, DNS service can also work with SDN controllers to provide alternative IP addresses and hopping rules for the protected servers. We can use DNS service changing strategy to defend DDoS attacks and change the traffic pattern on the protected servers.

The above network-level MTD strategies provide various adaptation choices on a protected network, which makes the protected nodes and network area dynamic and unpredictable. Several strategies can be easily applied on the mainstream network platforms, such as changing network attributes and network traffic manipulation. Some research has been tried on network migration [32], network diversification [15], and network diversification [30], which reveals the potential benefits and influence brought to the network. These research results can be referred to when implementing the related MTD strategies. However, some strategies

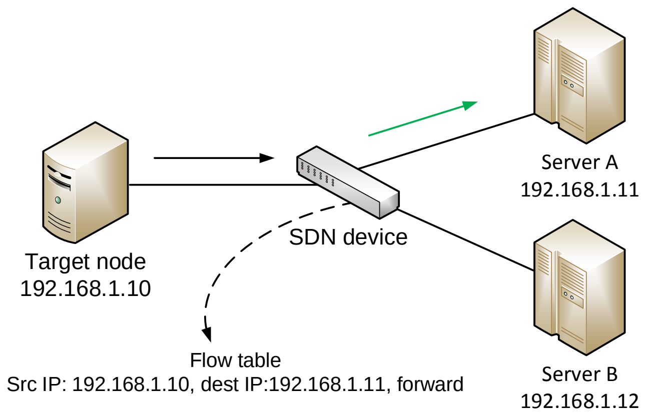

Figure 2. Network Configuration Mutation

need more support from the underlying network infrastructures. For example, changing network topology and network reflection strategies need some programming capability on the underlying network devices. Depending on the protected network environment and protection goals, the security engineers may choose the corresponding MTD strategies to implement in their specific production network environment. In this paper, we design and implement Shoal in SDN compatible network environment.

5. Prototype

We build up a Shoal prototype in an SDN network with virtualization technologies. The entire virtual network environment is running on a CentOS 7.0 host with kernel 3.10 x86_64, which is equipped with Intel Core i7-3370 3.4Ghz and 16GB RAM. All the network nodes and devices are running with virtualization technologies. The network nodes are running as KVM virtual machines and connected with Linux bridge, TUN/TAP devices, and virtual network devices. To provide the MTD strategies designed with SDN, we use SDN controller Ryu v4.24 [33] and Open vSwitch v2.5.0. Also, we customized the DNS service to provide IP reshuffling [10] for the protected network nodes. We used the previous work sniffer reflector [30] to provide traffic reflection to the protected network. In this section, we will demonstrate the eight MTD strategies, including Network Configuration Mutation, Network Route mutation, Network Topology Mutation, Network Traffic Manipulation, Network Diversification, Network Elements Migration, Network Address Shuffling, and DNS Service Mutation. In this section, we will not introduce the implementation of Network Traffic Reflection strategy, for it has been illustrated in the previous Sniffer Reflector work. The details about Sniffer Reflector can be located in paper [30].

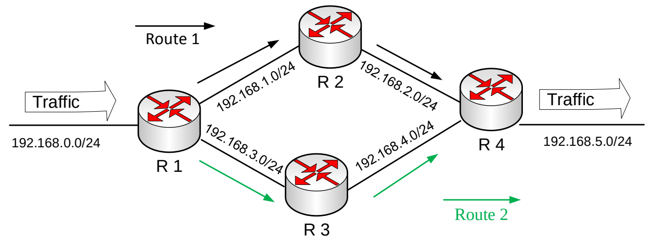

Figure 3. Network Route Mutation

5.1. Scenario 1: Network Configuration Mutation

In this scenario, we show how we use the Network Configuration Mutation strategy to change the network access control permission. We use Open vSwtich and SDN controller Ryu to implement the network access control functions. By dynamically changing the flow rules in the SDN device, the network access permission is turned on or off during the running time. Figure 2 shows the network access control on the SDN devices. In the figure, the arrow line stands for the network traffic flow from the target node to servers. The SDN device, which is implemented with Open vSwitch, maintains a flow table recording the flow rules among the connected network nodes. The network flows of the target node are controlled by the SDN devices. Initially, we assume the target node does not have the permission to visit server A, and we want to turn off the target node's network access permission to server A. By default, the OpenFlow device will not forward any packet of the target node to server A or server B. To configure the target node with network access rules to server A, we install a forwarding flow rule *(port num, src Mac, dest Mac, 0800, vlan1, 192.168.1.10, 192.168.1.11, 4, *, 80, port num of server A)* to the SDN device, which instructs the packet is allowed from the target node to server A. The rule allows the SDN device to forward the traffic from 192.168.1.10 to 192.168.1.11 at the specific port number.

5.2. Scenario 2: Network Route Mutation

Network route information is an important network characteristic for attackers. Through the network hopping information, the attackers are able to realize the knowledge of the passing routers and network structures. Dynamically changing the network route information can help invalidating the attackers' collected information. We practice the Network Route Mutation strategy by dynamically changing the routing rules among the connected network devices. Depending on the network deployment environment, the Network Route Mutation strategy can be implemented either on the SDN devices or the virtual routers. In this scenario, we implement the route mutation strategy in a virtual network environment connected with virtual routers. Figure 3 shows a simple example of route mutation process information. The black arrow line stands for the original traffic flow route, while the green arrow line stands for the mutated traffic flow route. There are six subnetworks in the figure, which include 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24, 192.168.4.0/24, and 192.168.5.0/24. In the original route, the traffic from subnetwork 192.168.0.0/24 to subnetwork 192.168.5.0/24 goes through R1 and R2. To mutate the route, we first remove the routing rule <dest 192.168.5.0/24, R2> of R1 and <dest 192.168.5.0/24, R4> of R2. Then we add the routing rule <dest 192.168.5.0/24, R3> to R1 and <dest 192.168.5.0/24, R4> to R3. The routing rules operation commands are executed as script files on the virtual routers. After the route mutation, we use the traceroute tool to verify the new route between R1 and R4, and the result shows the flow path is changed as expected.

5.3. Scenario 3: Network Topology Mutation

The dynamic network topology is able to raise the attacking bar for the attacks, as the collected network information may no longer be effective in the mutated network structure. We fulfill the Network Topology Mutation strategy in a network environment composed of five subnetworks. We use four virtual routers to connect the five subnets. For demonstration purposes, virtual machines are used to simulate virtual routers and connect them with Linux bridge and virtual TUN/TAP devices on the host machine. After the virtual machine router receives the notification from MTD Strategy Center, it will execute a predefined topology script to change the topology of the virtual network. Fig. 4 shows the topology mutating process. There are three phases in the mutation figure. The left side shows the original network topology before mutation, and the right side shows the changed network topology after mutation. The mutation process is demonstrated in the middle part.

As we can see in Figure 4, we plan to mutate the network topology by "moving" router R4 from R2 to R3. Before mutation, we have virtual router R1, R2, R3, and R4 to construct a connected virtual network. The five virtual LANs are 192.168.1.0, 192.168.2.0, 192.168.3.0, 192.168.4.0, and 192.168.5.0. Each router has a well-configured routing table through which it can reach all other networks. On each virtual machine, the routing information is configured to communicate with each other. For example, if R1 wants to send out a packet to R4, the first hop should be R2. R1's routing table (before and after mutation) is listed in the Fig. 4. To mutate the topology, we need to migrate the virtual router R4 from R2 to R3. The router migration

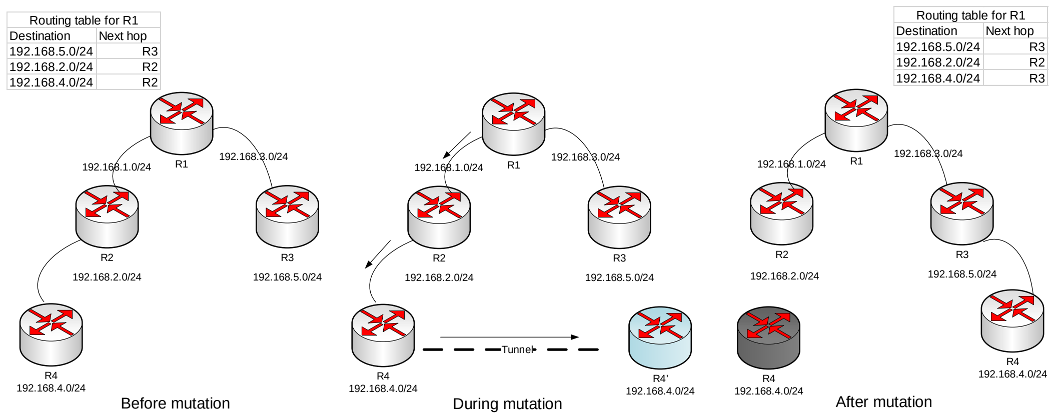

Figure 4. Network Topology Mutation

problem has been discussed in the previous research. Wang et al. [34] [35] proposed a migration method for migrating a virtual router among different physical routers, which can achieve a minimal packet loss and complex routing reconfiguration. We take Wang's method to achieve Network Topology Mutation. The following is the mutation process. We first create a new virtual router R4' in the new network location, and The forwarding rules and configurations of R4 are fully copied to R4'. Then we create a GRE tunnel between R4 and R4'. The tunnel is represented in the figure as dash lines between R4 and R4'. If there are network nodes connected to R4, the connected nodes will be migrated to the new network location one by one [36] [37]. When one node, for example, N1 finishes its migration, the traffic of N1 will be redirected from R4 to R4' through the GRE tunnel, where the new virtual router R4' will forward the traffic to the new node N1'. Although this method will greatly reduce the packet loss, there is still a network offline time from hundreds of milliseconds to seconds. After R4 and all its attached network nodes are migrated to the new network position, all the routing path to R4' will be adjusted and the new traffic will follow the adjusted routing path flowing to the new virtual router R4'. At this time, the original virtual router and GRE tunnel will be removed from the topology. After the topology mutation, we use traceroute command to verify the connectivity among any two routers and the results show we successfully change the topology of the virtual network.

5.4. Scenario 4: Network Traffic Manipulation

The Network Traffic Manipulation is a common network security strategy and can be used to investigate and modify the network packet content. Network traffic investigation is wildly used in firewalls and IDS/IPS,

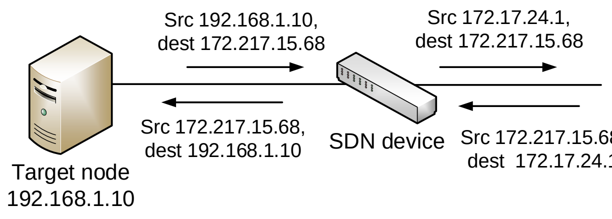

Figure 5. Network Traffic Manipulation

which covers deep packet inspection, signature-based intrusion detection, and so on. The modified network packet fields include packet header (source/destination IP address, source/destination port number, and source/destination MAC address), payload content (IP TTL values, protocol payload content, and so on), and other specific protocol-related content. In this scenario, we use the Network Traffic Manipulation strategy to implement a simple network address translation function. Figure 5 shows the address translation process. We employ Open vSwitch to implement the SDN device, and the SDN device is controlled by a Ryu controller. In the figure, the target node is located in a local area network 192.168.1.10, whose local IP address cannot be recognized in the public network. To visit the public network, the target node's local IP address needs to be mapped to a public IP address at the SDN devices. Similar to an NAT function, we implement the network address translation function with the OpenFlow rules. When the target node wants to visit the public web server at 80 port, the SDN device needs to translate the source IP 192.168.1.10 to the public IP 172.17.24.1; when the response traffic from the public network, the SDN device needs to translate the destination IP from 172.17.24.1 to 192.168.1.10. According to the translation rules, we install the corresponding flow

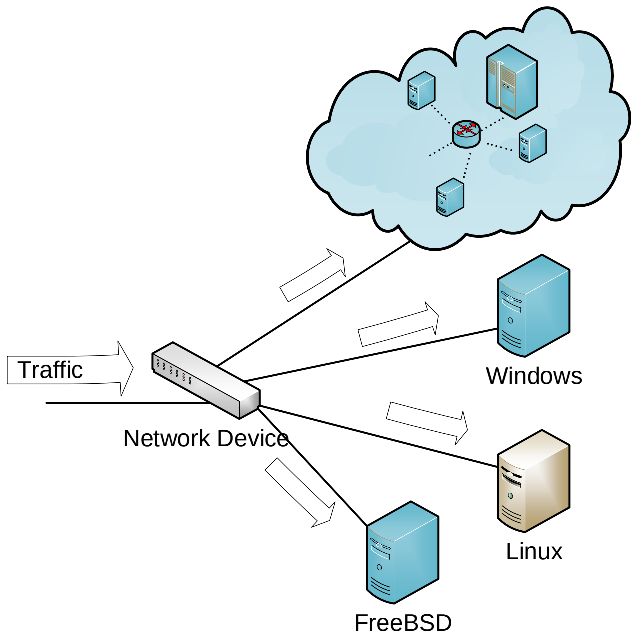

Figure 6. Network Diversification

rules *(*, *, *, 0800, *, 192.168.1.10, *, 4, *, 80, set source IP = 172.17.24.1)* and *(*, *, *, 0800, *, *, 172.17.24.1, 4, 80, *, set destination IP = 192.168.1.10)* to the SDN device. The two flow rules will have instructions to set the IP address fields. With the two flow rules, we verify that the target node is able to visit the web services.

5.5. Scenario 5: Network Diversification

The goal of Network Diversification is to increase the diversity of the target network. By adding additional network nodes running with different operating systems or software stack, we provide an obfuscated network view to the potential attackers, which can delay the attack activities of adversaries and invalidate the collected information. We use virtualization technologies and SDN devices to realize the Network Diversification strategy. We use KVM virtual machine and honeypot to run as additional nodes offering different network views. Similarly, the SDN controller and device are implemented with Open vSwitch and Ryu controller. Figure 6 demonstrates a view of the target network with the Network Diversification strategy. In the figure, the not-painted nodes are the real nodes in the target network. The nodes and network area in shadow blue are virtualized nodse and network area, which provide diverse systems and network views to the outside. When the probe traffics visit the diversified target network, it will get an obfuscated network view, including various virtual nodes running different OS and software stack. We can dynamically add or remove the virtual nodes in the target network. The virtual node creation and removal command are recorded in a script file, and the script file is executed when receiving a signal from the MTD strategy center. The Network Diversification strategy can help defending network reconnaissance attack and APT attack (the obfuscated network view will disturb the activation of APT attack). Besides, it also could be combined with other strategies to protect the target network. We verify the diversified network environment with scan tool, and the result returns the obfuscated network view.

5.6. Scenario 6: Network Elements Migration

Virtualization technologies provide many tangible benefits to the IT industry, such as reduced operating cost, simplification of data center management, and quick provisioning of system resources. The Network Element Migration strategy makes use of virtualization to dynamically rearrange the system resources at network level. It is quite helpful to optimize communication locations, achieve fault tolerance, and adjust network efficiency goals. We use KVM virtualizaiton platform to implement the Network Element Migration strategy [38]. There are two kinds of network element migrations in virtual network. One is virtual network node migration, while the other is virtual network device migration. Virtual network node migration changes the location of a network node, while virtual network device migration relocates the virtual network devices, such as virtual routers or switches, to a new network location. In this strategy, we introduce virtual network node migration. The virtual network device migration has been given in the Network Topology Mutation strategy, which covers virtual network device migration, topology reconfiguration, and so on. Network node migration is supported by the virtualization platforms, which migrate virtual machines among different physical hosts. Figure 7 shows the virtual machine migration process between two different connected physical machines. In the figure, the V-node m is migrated from host A to host B through the underlying network devices. We use the KVM command *<virsh migrate –live centos6.0 qemu+ssh://root@192.168.1.10/system>* to migrate the virtual node centos6.0 (virtual machine) to a new host 192.168.1.10 in the local area network 192.168.1.0/24. The migration time depends on the load and size of the migrated virtual node, for the copied source memory pages may be changed during migration and need to be copied multiple times.

5.7. Scenario 7: Network Address Shuffling

Network Address Shuffling plays an important role at network-level moving target defense, which can be used to mitigate reconnaissance based attacks, scanning worms, and denial of service attacks. There

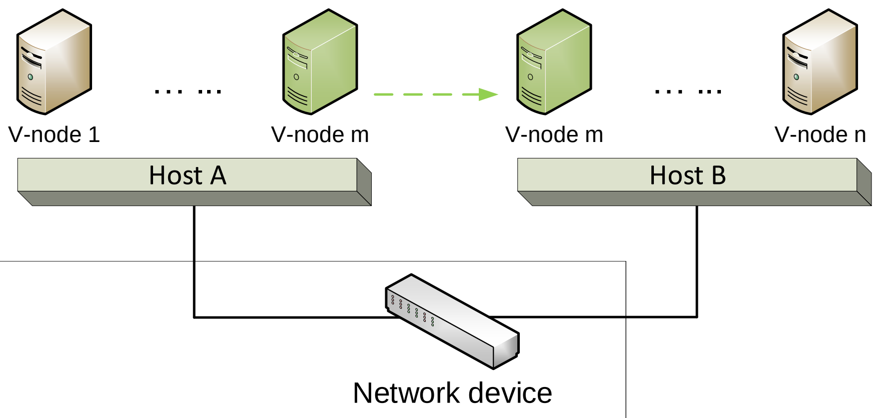

Figure 7. Network Element Migration

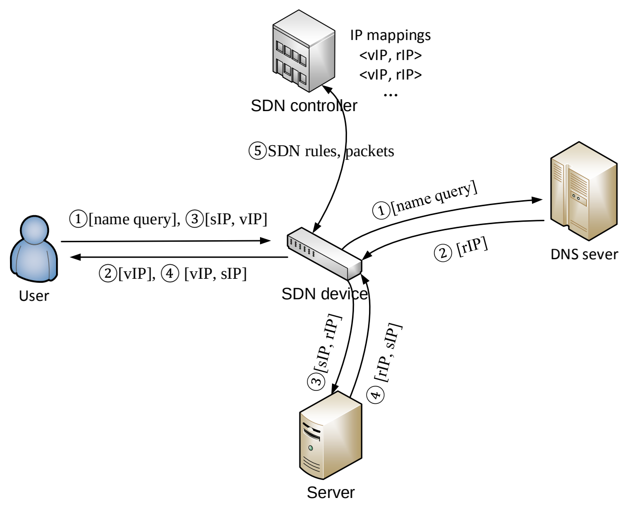

Figure 8. IP Address Shuffling

are different ways to implement this strategy. We take Jafarian¡¯s method [10] to implement the IP address shuffling strategy by making use of DNS service and the programmable SDN network device. In the protected network environment, the SDN controller is working as the Enforcement Agent and manages the SDN rules deployed in the SDN network devices. When the DNS server makes a response to a name query, the response will be intercepted by the SDN controller. The resolved real IP address will be replaced by a virtual IP address. After receiving the virtual IP address, the end user will use the virtual IP address to communicate with the server. When the end user traffic arrives at the SDN controller, the controller will indicate the SDN devices to translate the server's virtual IP address to the real IP address. Similarly, when the server sends back responses, the server's real IP address will be translated back to the virtual IP address. As a result, both the end user and the server has no awareness of the IP translation that happened in the communication. Also, to control the domain name query frequency, the resolved IP address can be set with a low TTL value, which decides how frequently the end user may resolve the domain name from DNS service. The defender may choose several ranges of virtual IP addresses and assign different virtual IP addresses to the protected servers.

Fig. 8 shows how the Network Address Reshuffling strategy works. We use Open vSwitch and SDN controller Ryu to build an SDN network with OpenFlow v1.3 protocol. The Ryu controller (as Enforcement Agent) is responsible for receiving the address reshuffling strategy and distributing the IP translation flow rules to the Open vSwitch. When the Ryu controller intercepts DNS response with real IP address, it will install the IP modification rules to the connected Open vSwitch. The server's real IP address will be replaced with a virtual IP address at Open vSwitch (shown in steps 1 and 2). As the figure shows in step 3 and 4, the server's IP address will be translated at the SDN device during the communication between the user and the server. Based on the fact that a scanned virtual IP has less chance to be scanned again, a counter may be used to record the scan times of the protected server. The defender may choose different action based on the counter value. Network Address Reshuffling can greatly reduce the accuracy of information gathering on the protected server.

5.8. Scenario 8: DNS Service Mutation

DNS service is designed to translate the domain name to IP addresses, which facilitates the users to locate the web resources more easily. DNS Service Mutation makes use of DNS service to provide an additional layer of uncertainties to the protected servers, and

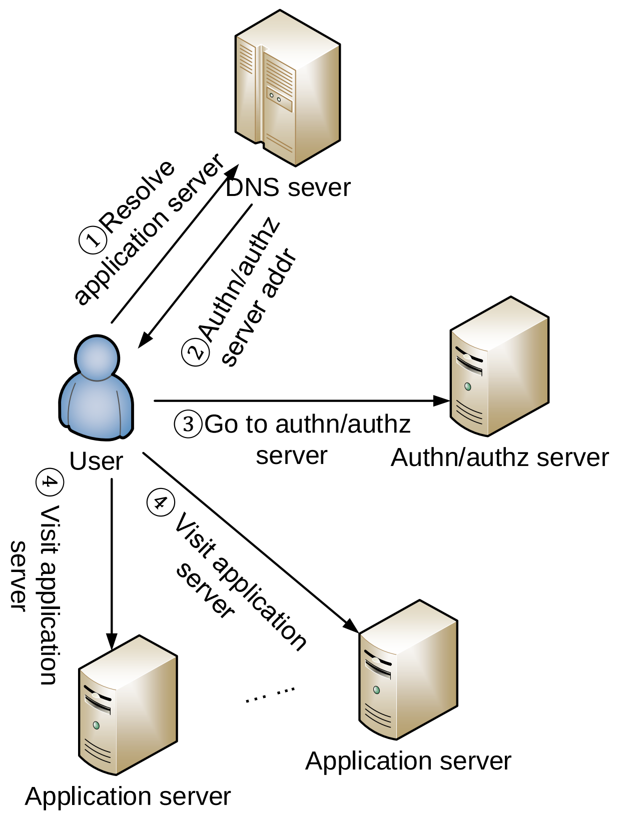
DNS sever
①Resolve application server
②Authn/authz server addr
User
③Go to authn/authz server
Authn/authz server
④ Visit application server
④ Visit application server
Application server
… …
Application server

Figure 9. DNS Service Mutation

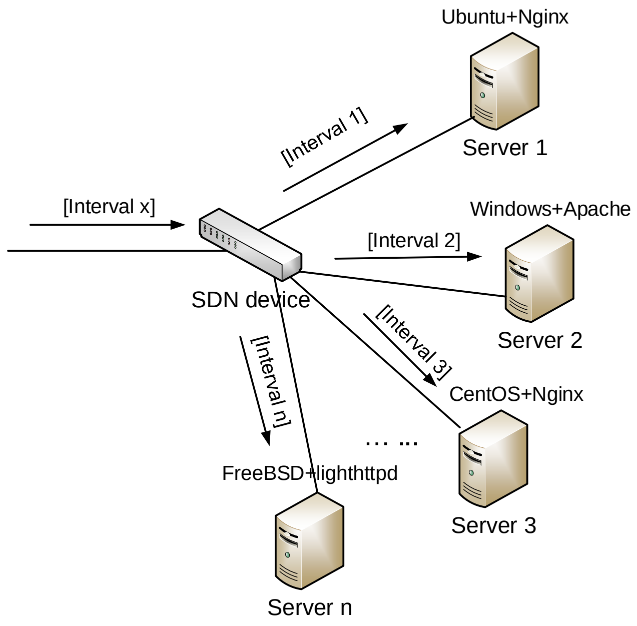

Figure 10. Moving Attack Surface with combined MTD strategies

it can be used to defend DDoS attack and adjust the server network traffic patterns. There are several methods proposed to realize this strategy [11] [31] [12]. One simple method proposed by Jia [11] is to map the protected application domain name with the authentication/authorization server's IP address We take Jia's method [11] to implement the DNS Service Mutation strategy. Figure 9 shows the simple DNS service mutation scenario. As the figure shows, there are four steps in the DNS service mutation strategy. Compared with the traditional DNS service, the users first go to the DNS service to obtain the IP address of the authentication/authorization servers (step 1 and 2). After passing the authentication/authorization process, the application servers are available to the users (step 3 and 4). As a result, the unauthenticated users will not be able to locate the application server's position and cannot compromise the application service.

6. Evaluation

In this section, we employ the combined MTD strategies provide by Shoal and evaluate the defense in terms of security effectiveness and performance cost. We first use Network Traffic Manipulation, Network Diversification, and Network Traffic Reflection to provide a Moving Attack Surface [14] [15] protection. Then, we use Network Traffic Manipulation, Network Route Mutation, and Network Topology Mutation strategies to mitigate the Crossfire attack [13].

6.1. Moving Attack Surface

Attack surface is the sum of the attack vectors in a software environment where an unauthorized user can explore to enter or extract data. Attack vector means any possible access point available for unauthorized users to intrude a system. For example, an attacker may make use of vulnerable network protocols, exploitable software vulnerabilities, and breakable interactive application interfaces to break into a software system. A lot of research has been conducted about the attack surface of the computer systems [15] [39] [40]. The existing methods tried to reduce, modify, or secure the attack surfaces. Unlike the existing methods, MTD is a new defense philosophy and introduces uncertainties and changes to the computer system. We try to use moving attack surface (MAS) to breaks the attacker assumptions that the surface is changed and the target server is always reachable. We use Shoal to establish a moving attack surface (MAS) to protect the network application servers.

To implement the MAS, we combine multiple MTD strategies, including Network Traffic Manipulation, Network Diversification, and Network Traffic Reflection, and apply them to the protected server. We use the Network Diversification strategy to introduce additional service nodes running different server software and operating systems to diversify the attack surface of the protected service. Every server node hosts the same service and has a different set of attack surface. Also, the Network Traffic Manipulation and Network Traffic Reflection strategies are used to detect and reflect the visiting traffic. Although the clients are visiting the same online service, the service may be provided by different server nodes. For example, assume we want to protect an http service, which may be initially deployed with Nginx on a Linux box, we may add Windows and FreeBSD boxes running different http servers, such as Apache, mini_httpd, and any other http servers. The inbound service request traffic will be transferred to the server nodes. Depending on the defense policy, the visiting traffic will go to a specific server node. Based on the server pickup algorithm, the serve nodes may serve simultaneously or alternatively. Figure 10 shows the moving target surface protection scenario. We use different operating systems and server software to diversify the attack surface of an http service. The SDN device is investigating the inbound traffic, and the http requests will be picked up and reflected to a specific server node. We use a time-based rotation algorithm to pick a server node. As can be seen in the figure, the visiting traffic arriving at different time intervals will go to a corresponding server node. The back-end server nodes will serve alternatively based on a time interval shift. For example, the requests arriving during time interval 1 will go to server 1, which is running an Ubuntu system and Nginx server; while the requests arriving during time interval 2 will go to another server node where a Windows system and Apache server are hosted. Due to the attack surface of the http service is variable under the MAS protection, an attacker has to initialize a successful attack within the given time interval, which greatly reduces the opportunities for the attackers to take over a server node. Consider the service shift among different server nodes introduces performance cost, a short shift interval may bring frequent server shift and high performance cost to the protected service. The time shift length can be decided by the security engineers based on the balance between performance and security.

We evaluate the MAS protection in the virtual network environment of Shoal. There are five guest virtual machines running as server nodes receiving http requests, and each VM is assigned with 1G RAM. The evaluation has two parts. One is security effectiveness, and the other is performance cost. We use the network scan tool Nmap to scan the http service during different time shifts, and the scan result shows the http service is provided by the different server software and operating systems. The result proves the MAS protection succeeds in providing diverse attack surfaces. We also evaluate the performance cost of the MAS protection. The evaluation is conducted with httperf, which is an http load generator for measuring web server performance. We first measure the maximum number of http requests the server node can handle per second when there is no server shift. The calculated result is around 280 to 300 requests per second. This number is low for the entire MAS prototype is deployed in a virtual environment, and all the server nodes and network devices are virtual and running on one host machine. Then, we execute httperf to generate http requests at an average 280 requests per second and obtained an average loss rate of 2.521 per server shift. The average loss rate is about 0.09%. We believe the performance cost brought by MAS protection is relatively low.

6.2. Crossfire Attack

Crossfire attack is first proposed by Kang et.al [13] in 2013. It is a powerful attack, which can degrade and often cut off network connections to a variety of selected server targets by flooding only a few network links with bots. Unlike other attacks, Crossfire attacks are undetectable by the targeted servers, for the targeted servers do not receive any attack traffic. What's more, the attack traffic is not detectable by the passing routers either, for the routers only receive low-intensity, individual flows that are not distinguishable from legitimate flows. The target area of the Crossfire attack could cover servers of an enterprise or the entire enterprise network. Before staring an attack to the target area, an attacker needs to have a set of decoy servers around the target area. The decoy servers are the publicly accessible servers surrounding the target area. The rule to choose decoy server is the route path from the attacker bots to the decoy server has a major intersection of the route path from the attack bots to the target area. To initialize a Crossfire attack on the target area, the first step is to construct the link-map between the attacker bots and the target area. The attacker makes use of route discovery tools, like traceroute, to discover the link-map between the attacker bots and the target area and decoy servers. Referring to the link-map, the attacker selects the target link for flooding. Based on the target link's bandwidth and position, the attacker assigns the individual flows to the corresponding bots, which keeps the flow rate low enough so that the network protection mechanisms will not detect them. When the traffic flows from all the bots exceed the bandwidth of the target link, the connectivity of the target area will be degraded or cut. More details about Crossfire attack can be referred in Kang's paper [13].

We combine Network Route Mutation, Network Topology Mutation, and Network Traffic Manipulation strategies to mitigate the Crossfire attack. The Network Route Mutation strategy is used to dynamically adjust

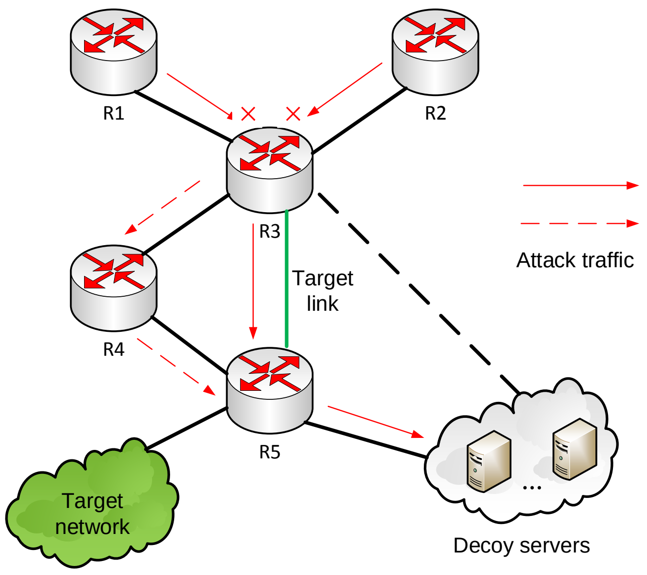

Figure 11. Defending Crossfire Attack

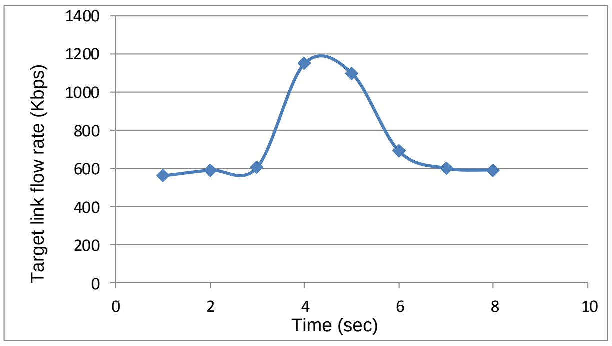

Figure 12. Target link flow rate

network routes among the connected network devices. Figure 11 shows the Crossfire attack defense scenario. We employ SDN controllers to monitor the target links' congestion statuses. SDN technologies provide fine-grained network monitoring functionalities, which collect real-time network statistics information over the SDN network. The controller will periodically probe the connected SDN devices and calculate the flow rates of the target links. The flow rate is calculated as *traffic_amount(t1) - traffic_amount(t0)/(t1 - t0)*, and the free bandwidth of the target link can be estimated as *link capability - current flow rate*. In Figure 11, the green solid line is the target link, and the red solid arrow lines stand for the attack traffic flows. When the target link's flow rate reaches to a threshold value, we know the target link is in congestion and the current flows on the target links will be rerouted to other links. The target links' flow rate will be reduced. As can be seen in Figure 11, the red dash lines represent the mutated network route of the attack traffic.

Although the target link's flow rate is reduced, the attacker may continue adding traffic pressures to the target link when the target network is still alive. The attacker may recalculate the link-map and re-pickup the target links. When the Network Route Mutation is not sufficient to defend Crossfire attack, we may use the Network Elements Migration strategy to migrate the decoy servers to a new network position, where their traffic flows will not affect the target links. As can be seen in the figure, the decoy servers are migrated from R5 to R3. Before migration, the route going to decoy servers will be *<R1, R3, R4, R5, decoy servers>*. After migration, the new route to go to decoy servers will be *<R1, R3, decoy servers>*, and the new route will not pass the target link. Consider the target link congestion status calculation time and route mutation time is far less than the time needed for the attacker to restart a new around Crossfire attack, the decoy server migration can successfully reduce the traffic flows on the target link.

However, the above two strategies may not be enough to eliminate the Crossfire attack. The attacker can still continue the Crossfire attack circle until sees a success, if the attacker owns enough bot nodes. Each time the target link congestion is relieved, the attacker may send another round of massive ICMP traffic and recalculate the link-map. There are several ways to identify the sources of the attack traffic [41] [42]. In this scenario, we use Wang's method [42] to catch the sources of the attack traffic. The massive ICMP traffic has two features, incremental TTL value and invalid destination port number. With these two features, we can make use of the Network Traffic Manipulation strategy to investigate and identify the ICMP traffic around the target network area. In each round of Crossfire attack, the source IP addresses of the massive ICMP traffic will be recorded. If an IP address shows more than once in the result, we conclude it is a bot node. In the future, the detected bot nodes traffic will be dropped. As the Figure 11 shows, the bot nodes traffic is identified and blocked at R3, and the block action is represented as cross mark.

We evaluate the Crossfire attack defense in the virtual network environment of Shoal. We use iPerf, a network performance measurement tool, to generate traffic to congest the target traffic. There are two kinds of traffic. One is normal traffic, and the other is attack. We first evaluate the Network Route Mutation strategy. We set the flow rate threshold for the target link as 1Mbps. When the target link's bandwidth exceeds 1Mbps, we regard the target link is in congestion. Once the

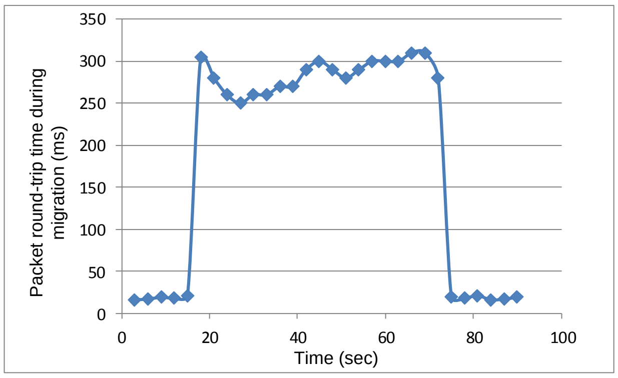

Figure 13. Packet round trip time

congestion happens, the decoy servers' traffic will be rerouted from route <R3, R5> to route <R3, R4, R5>. In our experiment, both normal traffic and attack traffic is sent at a rate of 600Kbps. Figure 12 show the flow rate variation of the target link. As can be seen from the figure, the target link flow rate exceeds 1Mbps around 4 second position. The flow rate starts falling after 4 second and gets back to 600 at 6 second, which shows the congestion is revolved in about 2 seconds. We use traceroute to verify the new route of the attack traffic and the new route is <R3, R4, R5>. We monitor the packet round-trip time during the decoy server mutation process. One decoy server and one switch is migrated from R5 to R3. We continuously send *ping* packets to the decoy server through R3 during the mutation process. Figure 13 shows the packet round-trip time during the migration. In the figure, we can see the transmission delay brought by the network tunnel during migration, and the packet round-trip time during migration is between 250ms to 310 ms. The delay is brought by the added network tunnel between R5 and the migrated new network device. Close to the end of the migration, there is a temporary network unavailability time 658ms, which is probably caused by the virtual machine migration implementation. We verify the route change after the migration, and the route path from R3 to decoy server changes. Also, the flow rate of the target link changes with the topology mutation. If the attacker continues starting a new round of Crossfire attack, we also employ Wang's method [42] to calculate the source of the flooding traffic and set the packet drop rules at R3, which practice the Network Traffic Manipulation strategy. The Crossfire attack is mitigated with the protection of Shoal.

7. Discussion

Although the MTD strategies are designed to bring benefits in terms of security, they also introduce various costs to the target network. The MTD cost includes the delayed network traffic transmission, consuming more computation and network resources, packet loss, temporary network unavailability, and so on. The Network Configuration Mutation may cause delayed traffic transmission. For example, assume a network node gets reconfigured with a new MAC address. It may need to rebroadcast its presence through the ARP protocol to communicate with other nodes. The reacquaint process may take several hundred milliseconds. If the node happens to have a upper-level TCP connection, the TCP traffic may be delayed during the reacquaint process. Besides, the Network Route Mutation may also delay the network traffic, for the new route path need to be added to all the bypassing network devices. Similarly, the Network Traffic Manipulation and Network Topology Mutation strategies may also delay the network traffic. There will be a low packet loss rate during the router migration. The lost packets need to be retransmission after all the migration finish. In order to introduce diversity to the target network, MTD philosophy requires more system and network resources for additional distinct nodes and presents a diversified network view. The additional nodes consume more computation and network resources. The Network Diversification uses virtual machines and honeypots to simulate numerous virtual nodes in a target network. Both of them will consume host memory, network bandwidth, and other system resources. The Network Address Shuffling strategy may occupy the unused network resources (free IP addresses) in the target network to exchange security benefits. By contrast, some MTD strategies may cause temporary service unavailability in the target network (some network applications may bear this, but some may not). For example, in the Network Elements Migration strategy, virtual machines are transferred from one physical host to another physical host. Although the VM migration will migrate the main memory, there is a temporary network break right after the closure of the original VM. The new generated VM needs to take some time to take over the ongoing network traffic. The Network Topology Mutation strategy may also cause the temporary unavailability for the virtual network devices are running in the virtual machine. Except for the cost brought to the protected nodes and services, MTD strategies may also affect the normal network management. A target network needs to keep the order of network management protocols, like DHCP and ICMP protocols, to maintain the normal network management and operation. When deploying the MTD strategies, the security engineer needs to take care of the network management protocols and keep them running smoothly.

Consider the performance cost brought by the network level MTD network-level strategies, there are different considerations when applying the strategies to different application servers. Different types of servers may take the different loss on the different performance costs of MTD strategies. For example, the stateless services (do not need to hold state information between requests) do not rely on the previous traffic information. They do not rely on reliable connections and are allowed to bear the above performance cost introduced by the MTD strategies. If there is temporary service unavailability, it is easy for the clients to resend the visit requests. The representative stateless service is NFS service, which is default running with UDP protocol. Compared with TCP protocol, UDP is faster and needs less overhead. Based on the features of the stateless services, the network level MTD strategies are natural choices for stateless service protection. Unlike stateless services, stateful services rely on reliable network connections and need to record the previous traffic information. The stateful services can be divided into two categories. One is short-session based service, and the other is long-session service. The short-session services are typically transaction oriented and request-and-response services. If a transaction is broken before the finish, the client can send another request and it will not take too much price. The short-session services include the DNS service, certificate services, and some database services. These services are not high-availability services and usually rely on simple query-and-reply communications. For these short-session services, it is also relatively reasonable for them to take the network level MTD strategies. Compared with the short-session services, the long-session services need to maintain the long stateful network connections, which requires stable network service. A high reliable network service is required by the long-session services. If the network connection is interrupted during the service, the clients need to reconnect the server and re-establish the connection. The representative long-session services are Telnet and proxy service. For the long-session services, some MTD strategies (may cause temporary network unavailability) may not be good choices.

8. Conclusion

The static nature of current computer systems makes them easy targets of cyber attack. Attackers are able to spend as much time as they can to find an effective way to compromise a target system. MTD was proposed as a promising defense paradigm to break the static nature of current computer systems. Consider most online attacks use computer network as an attacking vector, we try to adopt the MTD defense philosophy and bring changes and uncertainties to computer networks, to raise the attacking bar for attackers. In this paper, we propose a network level Moving Target Defense Engine and a batch of network-level MTD strategies to protect virtualized networks. With Shoal, we can dynamically change the protected network environment to disturb attackers. We established a Shoal prototype to protect a virtual network and demonstrated a couple of defense scenarios of our method. Our experimental results show the effectiveness of our engine and the MTD strategies.

References

[1] Plauger, P.J. (1992) *The standard C library* (Prentice-Hall, Inc.).

[2] Panjwani, S., Tan, S., Jarrin, K.M. and Cukier, M. (2005) An experimental evaluation to determine if port scans are precursors to an attack. In *Proc. International Conference on Dependable Systems and Networks, DSN 2005.* (IEEE): 602–611.

[3] Vivo, M., Carrasco, E., Isern, G. and de Vivo, G.O. (1999) A review of port scanning techniques. *Computer Communication Review* **29**(2): 41–48.

[4] Lyon, G.F. (2009) *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning* (Insecure).

[5] DHS, USA. URL https://www.dhs.gov/science-and-technology/csd-mtd.

[6] Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C. and Wang, X.S. (2011) *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats* (Springer).

[7] Zheng, J. and Namin, A.S. (2019) A survey on the moving target defense strategies: An architectural perspective. *Journal of Computer Science and Technology* **34**(1): 207–233.

[8] Peng, W., Li, F., Huang, C.T. and Zou, X. (2014) A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In *IEEE International Conference on Communications (ICC)*: 804–809.

[9] Zhuang, R., DeLoach, S.A. and Ou, X. (2014) Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense, MTD, Scottsdale, Arizona, USA*: 31–40.

[10] Jafarian, J.H., Al-Shaer, E. and Duan, Q. (2012) Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN.*

[11] Jia, Q., Wang, H., Fleck, D., Li, F., Stavrou, A. and Powell, W. (2014) Catch me if you can: A cloud-enabled ddos defense. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (IEEE): 264–275.

[12] Skowyra, R., Bauer, K., Dedhia, V. and Okhravi, H. (2016) Have no phear: Networks without identifiers. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*: 3–14.

[13] Kang, M.S., Lee, S.B. and Gligor, V.D. (2013) The crossfire attack. In *2013 IEEE symposium on security and privacy* (IEEE): 127–141.

[14] Huang, Y. and Ghosh, A.K. (2011) Introducing diversity and uncertainty to create moving attack surfaces for web services. In *Moving target defense* (Springer), 131–151.

[15] Wang, L., Wang, Z., Sun, K. and Jajodia, S. (2013) Reducing attack surface with vm-based phantom server. In *MILCOM 2013-2013 IEEE Military Communications Conference* (IEEE): 1429–1435.

[16] Lantz, B., Heller, B. and McKeown, N. (2010) A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM Workshop on Hot Topics in Networks. HotNets*.

[17] Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M. and Gu, G. (2012) A security enforcement kernel for OpenFlow networks. In *Proceedings of the first workshop on Hot topics in software defined networks* (ACM).

[18] Scott-Hayward, S., O'Callaghan, G. and Sezer, S. (2013) Sdn security: A survey. In *IEEE SDN for Future Networks and Services, SDN4FNS*.

[19] Shin, S., Yegneswaran, V., Porras, P.A. and Gu, G. (2013) AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In *ACM SIGSAC Conference on Computer and Communications Security, CCS*.

[20] Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G. and Tyson, M. (2013) FRESCO: modular composable security services for software-defined networks. In *20th Annual Network and Distributed System Security Symposium, NDSS*.

[21] Sonchack, J., Aviv, A.J., Keller, E. and Smith, J.M. (2013) Enabling practical software-defined networking security applications with ofx. In *23th Annual Network and Distributed System Security Symposium, NDSS*.

[22] Dunlop, M., Groat, S., Urbanski, W., Marchany, R. and Tront, J. (2011) Mt6d: A moving target ipv6 defense. In *Military Communications Conference, 2011* (IEEE).

[23] Al-Shaer, E. (2011) Toward network configuration randomization for moving target defense. In *Moving Target Defense - Creating Asymmetric Uncertainty for Cyber Threats*, 153–159.

[24] Kewley, D., Fink, R., Lowry, J. and Dean, M. (2001) Dynamic approaches to thwart adversary intelligence gathering. In *Proc. of DARPA Information Survivability Conference &amp; Exposition II,*: 176–185.

[25] Groat, S., Dunlop, M., Urbanksi, W., Marchany, R. and Tront, J. (2012) Using an ipv6 moving target defense to protect the smart grid. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*: 1–7.

[26] Li, J., Reiher, P. and Popek, G.J. (2004) Resilient self-organizing overlay networks for security update delivery. *Selected Areas in Communications, IEEE Journal on* **22**(1): 189–202.

[27] Gupta, A., Zangrilli, M., Sundararaj, A.I., Huang, A.I., Dinda, P.A. and Lowekamp, B.B. (2006) Free network measurement for adaptive virtualized distributed computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium* (IEEE): 10–pp.

[28] Sundararaj, A.I., Gupta, A. and Dinda, P.A. (2004) Dynamic topology adaptation of virtual networks of virtual machines. In *Proceedings of the 7th workshop on Workshop on languages, compilers, and run-time support for scalable systems* (ACM): 1–8.

[29] Cho, J.H., Sharma, D.P., Alavizadeh, H., Yoon, S., Ben-Asher, N., Moore, T.J., Kim, D.S. *et al.* (2020) Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials* **22**(1).

[30] Wang, L. and Wu, D. (2016) Moving target defense against network reconnaissance with software defined networking. In *Proceedings of Information Security - 19th International Conference, ISC 2016, Honolulu, HI, USA*.

[31] Wang, H., Jia, Q., Fleck, D., Powell, W., Li, F. and Stavrou, A. (2014) A moving target ddos defense mechanism. *Computer Communications* **46**: 10–21.

[32] Woesner, H. and Verbeiren, D. (2015) SDN and NFV in telecommunication network migration. In *Fourth European Workshop on Software Defined Networks, EWSDN, Bilbao, Spain*: 125–126.

[33] Asadollahi, S., Goswami, B. and Sameer, M. (2018) Ryu controller's scalability experiment on software defined networks. In *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)* (IEEE).

[34] Wang, Y., van der Merwe, J.E. and Rexford, J. (2007) Vroom: Virtual routers on the move. In *HotNets*.

[35] Wang, Y., Keller, E., Biskeborn, B., Van Der Merwe, J. and Rexford, J. (2008) Virtual routers on the move: live router migration as a network-management primitive. *ACM SIGCOMM Computer Communication Review* **38**(4).

[36] Keller, E., Ghorbani, S., Caesar, M. and Rexford, J. (2012) Live migration of an entire network (and its hosts). In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*: 109–114.

[37] Lo, S., Ammar, M. and Zegura, E. (2013) Design and analysis of schedules for virtual network migration. In *2013 IFIP Networking Conference* (IEEE): 1–9.

[38] Deshpande, U., Wang, X. and Gopalan, K. (2011) Live gang migration of virtual machines. In *Proceedings of the 20th international symposium on High performance distributed computing*: 135–146.

[39] Manadhata, P.K. and Wing, J.M. (2010) An attack surface metric. *IEEE Transactions on Software Engineering* **37**(3): 371–386.

[40] Szefer, J., Keller, E., Lee, R.B. and Rexford, J. (2011) Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM conference on Computer and communications security*: 401–412.

[41] Rasool, R.U., Ashraf, U., Ahmed, K., Wang, H., Rafique, W. and Anwar, Z. (2019) Cyberpulse: a machine learning based link flooding attack mitigation system for software defined networks. *IEEE Access* **7**: 34885–34899.

[42] Wang, J., Wen, R., Li, J., Yan, F., Zhao, B. and Yu, F. (2018) Detecting and mitigating target link-flooding attacks using sdn. *IEEE Transactions on Dependable and Secure Computing* **16**(6): 944–956.