Moving Target Defense Discrete Host Address Mutation and Analysis in SDN

Charan Gudla, Andrew H. Sung
School of Computing Sciences and Computer Engineering
The University of Southern Mississippi
Hattiesburg, MS, USA
{charan.gudla, andrew.sung}@usm.edu

***Abstract*—Moving Target Defense is a technique focused on disrupting certain phases of a cyber-attack. Reconnaissance is the preliminary phase of the attack in the cyber kill chain. The static nature of the existing networks gives an adequate amount of time to the adversaries in gathering enough data concerning the target and succeed in mounting an attack. Randomization of the host addresses is well known MTD technique that hides the actual network configuration from external scanners. Although random host mutation techniques are investigated extensively, the limitations such as less availability of unused public address space for mutation and host unavailability due to mutation time interval deteriorate the network's stability. Due to address space unavailability, each host address's mutation is not feasible according to the time interval, or the address space is repeated multiple times. When the host establishes a session of transmitting or receiving data, due to mutation interval, the session is interrupted, leading to the host's unavailability. In this paper, we propose a moving target defense technique to achieve the following objectives: (1) using mutation technique, randomization of IP addresses is achieved to create high uncertainty in adversary scanning; (2) the mutation time interval is separated from each host to preserve network performance and stability; (3) the mutation scheme is adapted by analyzing the data stats from the individual host (4) the analyzed data stats are used to manage the available unused address space.**

Keywords—Moving Target Defense (MTD), Software Defined Network, Cyber Kill Chain, Cyber-Attack, Reconnaissance, Mutation

I. Introduction

Reconnaissance is the preliminary phase of the cyber kill chain [1]. The static nature of the traditional network gives the adversary a significant amount of time in gathering information. Since the network is static, the adversary once gathers the information can spend an adequate amount of time to discover the vulnerabilities of the network and exploit. Moving target defense (MTD) technique is aimed to disrupt the phases of these scanning attacks.

The predictable behavior of the systems in the network allows the adversaries to determine the data flow parameters and scan the network accordingly. The MTD techniques are classified into mutation, diversity, and redundancy classes [2]. Shifting of attack surfaces [3][4][5] is referred as proactive defense measures against an adversary, which increases complexity and cost in system exploitation.

In this paper, we propose host address mutation deployed as a novel MTD technique in the SDN environment, which aims to create high uncertainty in adversary scanning by changing the IP addresses of the host in the network based on individual mutation time intervals. The main objectives of this paper are discussed in sequence. First, transparency is maintained in the mutation of the IP address of each host in the network. To provide transparency, the real IP (*rIP*) address of each host is unchanged, and a short-lived random virtual IP (*vIP*) address is assigned regularly to each real IP according to the mutation time interval.

Second, Once the session (S_{tcp}, S_{udp}, S_{icmp}) is established, the hosts are ready to transfer and receive the data from each other. The active session ($Act(h_i, h_k)$) mapping is created for each host, and it is monitored. Since the session time interval varies for each host, the mutation time interval also varies. Sometimes the session interval is long that adversaries can be successful in implementing a scanning attack. Even though the adversaries obtain the host's information, it is a virtual IP that will be changing rapidly. In this way, the mutation interval is different for every host, which does not need to change the address while in active session, preserving network performance and stability.

Third, we assume that every host in the network will not be given privileges to access sensitive data, modify network configuration, change firewall settings, and simply not even BIOS settings of the host itself in an enterprise network topology. At this situation, the hosts with administrative privileges are targeted more than the other. The scanning attacks on those specific host IP addresses will be high when compared to other host IP addresses. The host active in transferring and receiving data is targeted more than the host, which is less active; this attribute can also be added because the attackers are more likely to collect more useful data within a short period.

Fourth, to provide enough IP addresses to hosts in the network, the unused public address space range should be equivalent to the number of mutations in a host time interval. In a network, there will be hosts that can be reached publicly and need public address range, and some hosts are internal to the network, which can be provided with private address space range. The private address space range is always huge than the public address space. This mutation scheme using data stats is also effective when the available address space is less. Due to insufficient address space, some of the hosts cannot be moved or the host address is repeated multiple times in a short interval of time. To avoid this problem, data stats are used to allocate the address space range for a host involved actively in the network and targeted by the attackers frequently.

To implement these techniques, traditional network implementation is costly and poses more challenges. We use Software Defined Networking (SDN) infrastructure, which is quite flexible in developing and managing the network with minimal operational overhead. The network controller RYU is used to monitor and control the network using OpenFlow 1.3 protocol. The network topology is built using Mininet. The experimental results and analysis of the simulated network will show the significant rise in defending the network against reconnaissance attacks by increasing uncertainty in scanning, complexity in gathering the information about the network systems.

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00017

II. Related work

A. Mutation of Real Endpoint Configuration

Dynamic Network Address Translation (DYNAT) is proposed by Kewley et al. [5] The adversaries spend most of the time in scanning the network, DYNAT replaces the information in TCP/IP header to prevent malicious scanning. Predefined key parameters are provided to the trusted users assuring the availability of the service. Network Address Space Randomization (NASR) proposed by Antonatos et al. [6] Aiming at worm attacks, infected endpoints are analyzed first. Using the DHCP protocol, the endpoint information is changed. Jia et al. [7][8] proposed disclosing the endpoint information before and after the hopping period to solve the problem of synchronization in IP address mutation. A keyed hashing based self-synchronization mechanism is proposed by Luo et al. [9] for encoding the port address to synchronize port transformation. The first-in-first-out mutation channel based on the rate of transmission is proposed by Badishi et al. [10] to reduce the cost of implementation. The problem of limited address space for IPv4 and constant hopping period is aimed by Dunlop et al. [11][12] proposed a moving target defense technique based on IPv6 (MT6D). IPv6 address space is adopted to scale the hopping address space. To set a hopping period, a pseudo-random number is used to enhance randomness.

B. Mutation of Virtual Endpoint Configuration

OpenFlow random host address mutation (OF-RHM) is proposed by Jafarian et al. [13] to defend against stealthy scanning. To prevent session interruption during address hopping, E Al-Shaer et al. [14] proposed Random Host Mutation (RHM) to achieve the transformation of the virtual address of a host and to obtain real-time management by deploying gateway hopping. To prevent the leakage of MAC address, MacFarland et al. [15] used the DNS hopping controller to hide IP address, port numbers, and link of the endpoint. Skowyra et al. [16] proposed PHARE, which is a network identity elimination mechanism to prevent MAC address leakage by transforming header randomly when the packet leaves the end host. Sun et al. [17] proposed Decoy-Enhanced Seamless IP Randomization (DESIR) to identify the unauthenticated node in the platform and analyze the behavior to confirm malicious or not. Once confirmed as malicious, it uses honeypots to change the information of the endpoint rapidly. Jafarian et al. [18] proposed Spatial and Temporal Random Host Mutation (ST-RHM) to minimize the disadvantages due to the fixed hopping period. Jafarian et al. [19] proposed a hypothesis to analyze the adversaries scanning behavior and implement the mutation technique. Self-Adaptive Endpoint Hopping Technique (SEHT) is proposed by Lei et al. [20] to address three categories of the scanning strategy. Based on the scanning strategy, endpoint information mutation and frequency are framed. Based on this, literature [21] proposed a self-adaptive attack surface. Shaolei Wang et al. [22] proposed to enlarge the changing scope of terminal hosts.

Based on the above literature review, the moving target defense technique is widely investigated, and broad methodologies have been proposed for mutation of host address. In this paper, we propose the methodology for mutation of host address, reducing the overhead of the controller by considering the data flow stats, and each host will be having discrete mutation frequency while preserving network performance and availability. The proposed methodology also handles address space effectively when its size is insignificant.

III. Proposed methodology and architecture

Moving Target Defense mitigates certain phases of cyber-attacks in the cyber kill chain [1]. The attacker model involves a series of steps from gathering the data to exploiting the vulnerabilities, being persistent in the network.

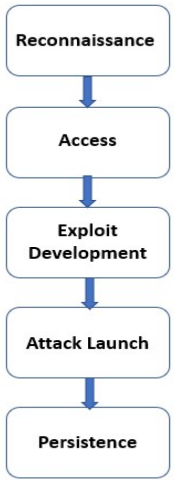


Fig.1. Cyber Kill Chain

In this moving target defense host address mutation scheme. Consider a network topology with N_s number of hosts. Here we created the topology in Mininet, and the RYU controller is used. The controller implements the data flow between the network components

$$N_s = \{h_1, h_2, h_3, h_4 \, . \, . \, . \, . \, . \, . \, . \, . \, . \, h_n\}$$

The real Ip address (rIP) of each host in the network is replaced by a short-lived virtual IP address (vIP). The virtual IP addresses are assigned randomly from the pool of available unused address space. The mapping function f_{map} is used to assign a virtual IP address to a real IP address.

$$vIP = f_{map}(rIP)$$

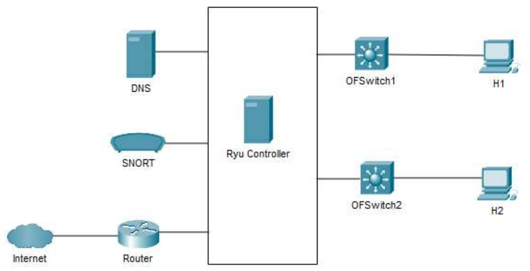


Fig. 2. Network Topology with two hosts

When the host h_i initiates communication with host h_j, a DNS request will be sent to the server to resolve the domain name into the vIP address of the host h_j and a session (S_{tcp} , S_{udp} , S_{icmp}) will be established between host h_i and h_j by installing the flows in the required switches and active session mapping $Act(h_i, h_j)$ is created for monitoring the data stats. The controller will monitor the data stats for each session. Data stats include parameters like session interval, source IP, destination IP, source MAC, destination MAC, data packets sent, and received in that time interval, etc. Once the session is ended, it is removed from the active session table. If the host wants to communicate and establish a session again, the DNS request should be sent to the server to retrieve the *vIP* address of the host.

Since each session time interval (S_t) depends on the length of time taken by the hosts to complete the transfer of data between each other, the randomization is applied when the session is ended to provide network performance and stability. The mutation time interval M_t is the session time interval. If the mutation time interval is random, the session between the hosts will be interrupted, and the packets are dropped.

$$S_t = S_{(tcp,\ udp,\ icmp)}(Act(h_i,\ h_j))$$

$$M_t = S_t$$

If the address space in network topology is N_s. The available active hosts in the network are N_a. The time taken for reconnaissance attack for each active host is $T_{r,h}$. The total average time $T_{a,\ r}$ that the adversary will spend on reconnaissance attack on all hosts will be:

$$T_{a,\ r} = N_a \text{ x } T_{r,\ h}$$

If $C_{r,\ h}$ is the cost to spend for reconnaissance attack on a single host. The total cost $C_{a,\ r}$ that the adversary needs to spend on reconnaissance on all active hosts will be:

$$C_{a,\ r} = N_a \text{ x } C_{r,\ h}$$

Fingerprinting is a technique used by the adversary to find the vulnerable host with high probability in the network by analyzing the data collected from the reconnaissance. Similar fingerprint operations will not be repeated on the host if it fails to find. The probability of identifying the vulnerable host in the network successfully with *i* steps will be:

$$P_i = 1 / N_a\ ,\ 1 \leq i\ \leq N_a$$

If $T_{f,\ a}$ is the fingerprint time spent on the single host, then the average time taken by the adversary for reconnaissance and fingerprinting analyses to find the vulnerable host is defined as:

$$T_f = N_s \text{ x } T_{a,\ r} + ((N_a + 1) \text{ x } T_{f,\ a}) / 2$$

From the above analysis if T_m is the mutation time interval, then:

$$T_m \leq (N_s \text{ x } T_{r,\ h} + T_{f,\ a})$$

From the above analysis, for each T_m interval, the adversary cannot complete a reconnaissance attack. The adversaries will target the host, which is very active in transferring and receiving data. The adversaries tend to collect a large amount of traffic within a small time. The amount of available unused address space should be equivalent to the number of hosts in the network topology. If the address space in network topology is N_s and the available unused address space is $N_{u,\ s}$ then:

$$N_{u,\ s} \equiv N_s$$

If the available unused address space is less, then some of the hosts in the network cannot be moved or cannot be moved frequently according to the mutation time interval. To solve this problem, the hosts which are highly active in the network are identified. The adversaries try to gather the maximum amount of data within less time, which is possible when the host is highly active and can be targeted. The scanning attack stats and data packets stats can be monitored using SNORT, and it will alert the controller according to the rules written into it. Using SNORT rules, we can analyze the data packets and discover the hosts which are highly active in the network and targeted by the adversaries. In this way, the unused IP addresses can be used effectively. Ryu and SNORT can be configured on a single machine or different machines.

A. Architecture

The topology is implemented in the Mininet network using the Ryu controller. The topology can be seen in Fig. 2 in detail. Ryu controller acts as intermediate central software to manage network activities like IP mutations, DNS responses, Session establishment, IP address space management, Data stats analysis using SNORT.

The unmatched packets in the OpenFlow switches will be encapsulated and sent to the Ryu controller. The controller discovers the type of packets and required actions are taken. If the packet is DNS request to resolve the host, then the controller will follow series of steps to authenticate the host and install the necessary flows in necessary OpenFlow switches to establish the session between the hosts. The active session will be mapped into the table $Act(h_i,\ h_j)$ to track the number of active sessions in the network. The mutation interval is discrete for each host, and the session will be terminated when the hosts stop communicating with each other. Every time the hosts need to communicate, the name should be resolved and establish a session between each other. SNORT will monitor all the data packets and alert the controller in case of anomalies detected according to the rules. SNORT will collect data stats about each host, which can be used to discover highly active hosts in the network. This data can also be used when the unused address space is limited.

TABLE I. DESCRIPTION OF NOTATIONS

N_s	Network address space
N_a	Active hosts in the network
rIP	Real IP address
vIP	Short-lived virtual IP address
f_{map}	Real IP to virtual IP mapping function
S_{tcp}	TCP session between hosts
S_{udp}	UDP session between hosts
S_{icmp}	ICMP session between hosts
$Act(h_i,\ h_j)$	Active session mapping between host h_i and host h_j
S_t	Session time interval
M_t	Mutation time interval
$T_{r,h}$	Time taken for reconnaissance attack on each active host in the network
$T_{a,\ r}$	Total average time spent on reconnaissance attack on all active hosts in the network
$C_{r,\ h}$	Cost to implement reconnaissance attack on a single host

$C_{a,r}$	The total cost of implementing reconnaissance attack on all active hosts
P_i	The probability of identifying vulnerable host in the network
$T_{f,a}$	Time spent for fingerprinting on a single host
T_f	Average fingerprinting time spent on all hosts in the network
$N_{u,s}$	Available unused address space
SNORT	Data packet sniffer, IDS, IPS

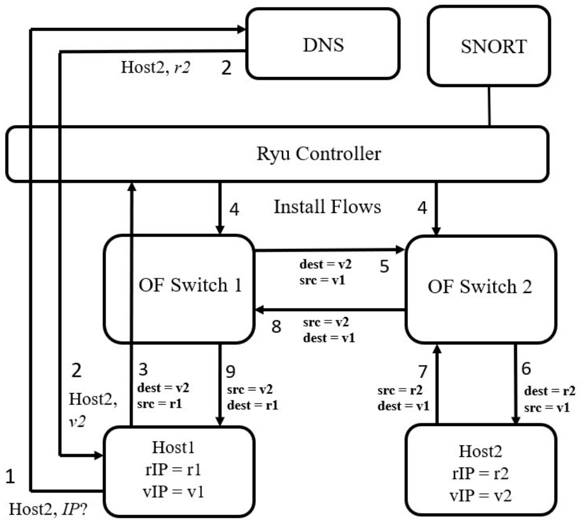


Fig. 3 Ryu controller and communication between two hosts

The communication [24] steps are as follows:

Step 1: Host1 sends DNS query to DNS server for Host2 IP address

Step 2: DNS server sends a response to the controller. The controller randomly chooses a virtual IP address ***v2*** from the pool of available IP addresses and sends it to Host1, creating a mapping between real IP address and virtual IP address. After changing the IP address, the controller sends the DNS response to Host1 with the virtual IP address.

Step 3: Host1 sends the data packets to the received virtual IP address ***v2*** with source IP as its real IP address ***r1***. Since the switch1 cannot match the destination IP address in its table, it forwards the packets to the controller.

Step 4: The controller checks the source IP and destination IP address. If the source IP address is real, It randomly chooses a virtual IP address ***v1*** from the pool and creates a mapping. The destination IP address will be verified with the mapping table, and when the entry is found, it checks the real IP address host and installs the necessary flows in the switches which can reach the destination host.

Step 5 & 6: The data packets are delivered to the Host2 according to the flows. At switch2, since the Host2 is directly connected to it, destination IP will be changed to ***r2*** by the controller to route the packets to Host2 because the switch2 is not aware of the virtual IP address of Host2

Step 7: Host2 responds to the data packets received, and now the source IP address will be real IP address ***r2,*** and the destination IP address will be ***v1***. The controller changes the ***r2*** real address to virtual ***v2***.

Step 8 & 9: The data packets are delivered to the Host1 according to the flows. At switch1, since the Host1 is directly connected to it, destination IP will be changed to ***r1*** by the controller to route the packets to Host1 because the switch1 is not aware of the virtual IP address of Host1.

The IP mutation is implemented by the Ryu controller using the following algorithm [24].

B. Algorithm Ryu Controller

```
if pkt is DNS request
    set DNS response change real rIP addr to vIP addr
    Map{rIP, vIP}
end if
if pkt is TCP, UDP
    if pkt.dest is rIP connected to the switch directly
        then pkt.out to rIP address
    end if
    if pkt.dest is vIP
        if pkt.src is rIP
            pkt.srcIP = vIP addr
            Map{rIP, vIP}
        end if
        refer pkt.dest to IP addr Mapping
        install necessary flows in OF-switches
            towards destination
    end if
end if
```

C. Traffic Generation and Reconnaissance

TCP and UDP traffic [24] are generated between the hosts using iPerf command line tool. To show the data packet analysis, data packets are sniffed by using TCPDump and saved in a PCAP file. The PCAP file is analyzed using Snort IDS. The scanning attack on the network is done using Nmap from Kali Linux OS. SNORT gives the details of number of TCP and UDP sessions between the hosts, events that needed to be reviewed etc.

By using the SNORT statistics, the scanned IP addresses can be known and the number of times each IP is scanned. Using Nmap scanning, the attacker can gather whether the host is up or down, open ports available, services running on the open ports and their versions etc. Even though the attacker gathers all the details, the IP address will be changed frequently creating complex situation to understand about the network

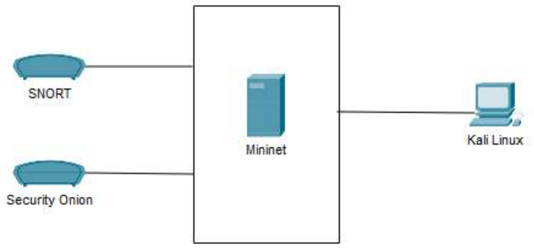


Fig. 4 Reconnaissance using Kali

Using security onion, further analysis can be done on the PCAP file. If the adversary succeeds reconnaissance stage and try to infect the PC, as a security analyst, detailed packet analysis should be done. The following Fig.5. shows the series of steps involved in the analysis.

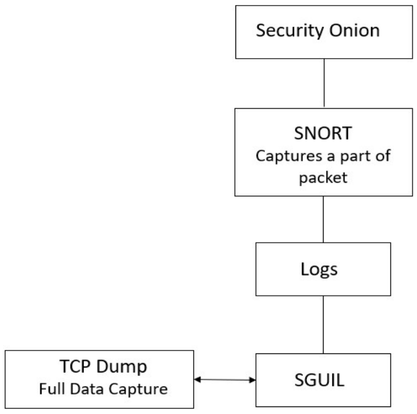
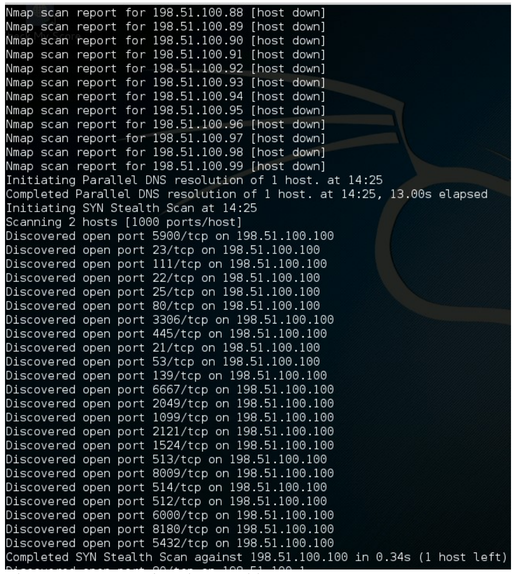


Fig. 5 SGUIL for packet analysis

IV. Results and Analysis

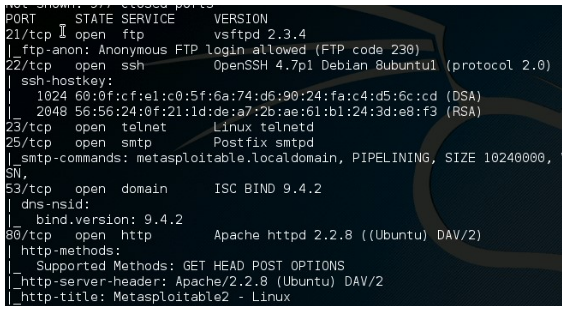
The following picture shows the Nmap scanning the network IP addresses.



```
Nmap scan report for 198.51.100.88 [host down]
Nmap scan report for 198.51.100.89 [host down]
Nmap scan report for 198.51.100.90 [host down]
Nmap scan report for 198.51.100.91 [host down]
Nmap scan report for 198.51.100.92 [host down]
Nmap scan report for 198.51.100.93 [host down]
Nmap scan report for 198.51.100.94 [host down]
Nmap scan report for 198.51.100.95 [host down]
Nmap scan report for 198.51.100.96 [host down]
Nmap scan report for 198.51.100.97 [host down]
Nmap scan report for 198.51.100.98 [host down]
Nmap scan report for 198.51.100.99 [host down]
Initiating Parallel DNS resolution of 1 host. at 14:25
Completed Parallel DNS resolution of 1 host. at 14:25, 13.00s elapsed
Initiating SYN Stealth Scan at 14:25
Scanning 2 hosts [1000 ports/host]
Discovered open port 5900/tcp on 198.51.100.100
Discovered open port 23/tcp on 198.51.100.100
Discovered open port 111/tcp on 198.51.100.100
Discovered open port 22/tcp on 198.51.100.100
Discovered open port 25/tcp on 198.51.100.100
Discovered open port 80/tcp on 198.51.100.100
Discovered open port 3306/tcp on 198.51.100.100
Discovered open port 445/tcp on 198.51.100.100
Discovered open port 21/tcp on 198.51.100.100
Discovered open port 53/tcp on 198.51.100.100
Discovered open port 139/tcp on 198.51.100.100
Discovered open port 6667/tcp on 198.51.100.100
Discovered open port 2049/tcp on 198.51.100.100
Discovered open port 1099/tcp on 198.51.100.100
Discovered open port 2121/tcp on 198.51.100.100
Discovered open port 1524/tcp on 198.51.100.100
Discovered open port 513/tcp on 198.51.100.100
Discovered open port 8009/tcp on 198.51.100.100
Discovered open port 514/tcp on 198.51.100.100
Discovered open port 512/tcp on 198.51.100.100
Discovered open port 6000/tcp on 198.51.100.100
Discovered open port 8180/tcp on 198.51.100.100
Discovered open port 5432/tcp on 198.51.100.100
Completed SYN Stealth Scan against 198.51.100.100 in 0.34s (1 host left)
```

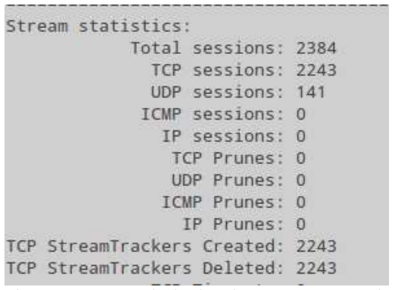
Fig. 6 Nmap scanning the IP addresses

From Fig. 6 you can see that the adversary can find the active host and identify the open ports available by using stealth scan. Fig. 7 shows the services running on the open ports and its versions. In Fig. 8, SNORT collects the individual host's total sessions for a certain period for further analysis.



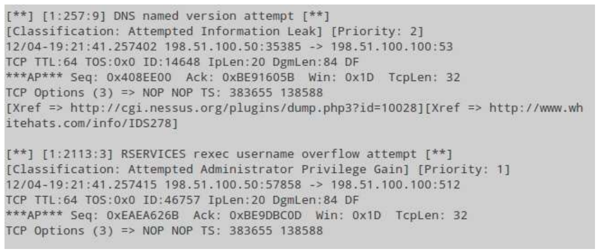
```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
SN,
53/tcp   open  domain      ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
```

Fig. 7 Nmap showing services on open ports and its versions



```
-------------------------------------
Stream statistics:
            Total sessions: 2384
              TCP sessions: 2243
              UDP sessions: 141
             ICMP sessions: 0
               IP sessions: 0
                TCP Prunes: 0
                UDP Prunes: 0
               ICMP Prunes: 0
                 IP Prunes: 0
TCP StreamTrackers Created: 2243
TCP StreamTrackers Deleted: 2243
```

Fig. 8 SNORT analysis showing TCP UDP sessions



```
[**] [1:257:9] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/04-19:21:41.257402 198.51.100.50:35385 -> 198.51.100.100:53
TCP TTL:64 TOS:0x0 ID:14648 IpLen:20 DgmLen:84 DF
***AP*** Seq: 0x408EE00  Ack: 0xBE91605B  Win: 0x1D  TcpLen: 32
TCP Options (3) => NOP NOP TS: 383655 138588
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028][Xref => http://www.wh
itehats.com/info/IDS278]

[**] [1:2113:3] RSERVICES rexec username overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/04-19:21:41.257415 198.51.100.50:57858 -> 198.51.100.100:512
TCP TTL:64 TOS:0x0 ID:46757 IpLen:20 DgmLen:84 DF
***AP*** Seq: 0xEAEA626B  Ack: 0xBE9DBC0D  Win: 0x1D  TcpLen: 32
TCP Options (3) => NOP NOP TS: 383655 138588
```

Fig. 9. SNORT Alerts generated

Fig. 9 shows the alerts generated by SNORT IDS. The attacker scanned the network for possible information leak, which includes a username overflow attempt classified under attempted administrator privilege gain. The alerts generated by SNORT shows the number of hosts scanned and the information gathered. The mutation technique thwarts the scanning by changing the IP address of the host. The attacker cannot know the details of which host the information is gathered. Some other types of events filtered by SNORT are shown below in Table II and only few are listed.

TABLE II. SNORT filtered events

Events filtered by SNORT
DNS named version attempt
RSERVICES rexec username overflow attempt
SCAN nmap XMAS
POLICY FTP anonymous login attempt
FTP PORT bounce attempt
CHAT IRC nick change

SNMP request tcp	
SNMP AgentX/tcp request	

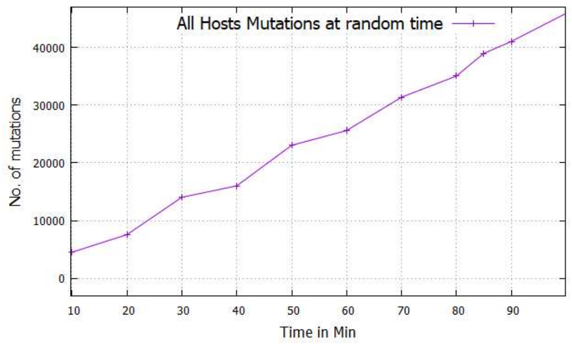


Fig. 10. Mutations of all hosts in a time interval

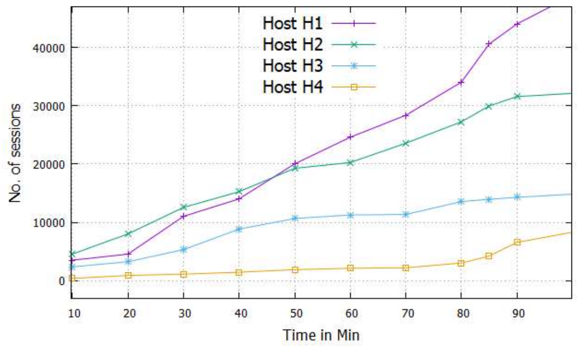


Fig. 11. No. of sessions of all hosts in a time interval

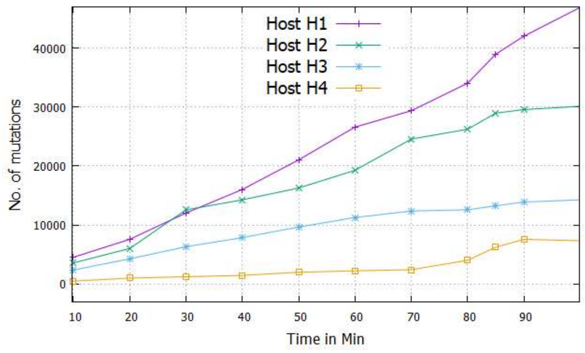


Fig. 12. No. of mutations of all hosts in a time interval

The number of mutations of all hosts for a certain period is shown in Fig. 10. Since the mutation time interval is random for all hosts, the host addresses are changed at the same time for all hosts. If a host is in a TCP/UDP session with other host or server, it will be interrupted due to random mutation. This will create unstability in the network and the network performance will be degraded. Fig. 11 shows the number of sessions established by each host. The session establishment number of each host vary with one another. Fig. 12 shows the number of mutations of each host after application of discrete mutation interval. The mutation technique is applied when the host terminate the session without interrupting. The stability of the network can be preserved with this technique. In Fig. 11, the number of sessions are different for every host. The host which is active in the network established more sessions than the other hosts. By analyzing this data, the address space range for each host can be assigned. The host which is active is assigned more address space range than other hosts which are less active. In this way, when the availability of host addresses is less , it can be managed with this technique. When the address space range is less, the same IP address is assigned to the host multiple times in a short interval. This gives adversary to gather more information with same IP address.

V. CONCLUSION

In this paper, we described the application of moving target defense and analysis using software-defined networking. The goal is to mitigate scanning attacks, which is the first phase of the cyber kill chain. We implemented the topology in Mininet using the Ryu controller. We described the architecture and roles in the controller. The implementation shows the rapid mutations with discrete time intervals of each host. The mutation time intervals depend upon the session time interval. The controller can handle address space when the available address space is minimal. The data stats are used to handle low address space and discover highly active hosts in the network.

Due to rapid mutations, the controller performance will degrade, eventually resulting in bandwidth consumption, memory overflow, latency, jitter, and other overheads. For future work, we plan to study the methodologies to minimize the overheads and effectively implement mutation techniques.

REFERENCES

[1] Ward, Bryan C., Steven R. Gomez, Richard Skowyra, David Bigelow, Jason Martin, James Landry, and Hamed Okhravi. “Survey of Cyber Moving Targets Second Edition.” (2018).

[2] Hong, J. B., & Kim, D. S. (2016). Assessing the Effectiveness of Moving Target Defenses Using Security Models. IEEE Transactions on Dependable and Secure Computing, 13(2), 163–177. DOI:10.1109/tdsc.2015.2443790

[3] P.K. Manadhata and J.M. Wing, “A formal model for a systems attack surface,” Advances in Information Security, vol.54, pp.1–28, 2011

[4] P.K. Manadhata and J.M. Wing, “An attack surface metric,” IEEE Trans. Softw. Eng., vol.37, no.3, pp.371–386, 2011.

[5] Gudla, Charan, Md Shohel Rana, and Andrew H. Sung. "Defense techniques against cyber attacks on unmanned aerial vehicles." Proceedings of the International Conference on Embedded Systems, Cyber-physical Systems, and Applications (ESCS). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018.

[6] D. Kewley, R. Fink, J. Lowry, and M. Dean, “Dynamic approaches to thwart adversary intelligence gathering,” in Proceedings of the DARPA Information Survivability Conference and Exposition II, DISCEX 2001, pp. 176–185, Anaheim, Calif, USA, June 2001.

[7] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, “Defending against hitlist worms using network address space randomization,” Computer Networks, vol. 51, no. 12, pp. 3471–3490, 2007.

[8] L.-Y. Shi, C.-F. Jia and S.-W. Lu, “Research on end hopping for active network confrontation,” Tongxin Xuebao/Journal on Communication, vol. 29, no. 2, pp. 106–110, 2008.

[9] K. Lin, C.-F. Jia and L.-Y. Shi, “Improvement of distributed timestamp synchronization,” Tongxin Xuebao/Journal on Communication, vol. 33, no. 10, pp. 110–116, 2012.

[10] Y.-B. Luo, B.-S. Wang, X.-F. Wang and B.-F. Zhang, “A keyed hashing based self-synchronization mechanism for port address hopping communication,” Frontiers of Information Technology and Electronic Engineering, vol. 18, no. 5, pp. 719–728, 2017.

[11] G. Badishi, A. Herzberg, and I. Keidar, “Keeping denial-of-service attackers in the dark,” IEEE Transactions on Dependable and Secure Computing, vol. 4, no. 3, pp. 191–204, 2007.

[12] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, “MT6D: a moving target IPv6 defense,” in Proceedings of the Military Communications Conference (MILCOM ’11), pp. 1321– 1326, IEEE, Baltimore, Md, USA, November 2011.

[13] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, “Te blind Man’s bluff approach to security using IPv6,” IEEE Security & Privacy, vol. 10, no. 4, pp. 35–43, 2012.

[14] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “Openfow random host mutation: transparent moving target defense using software-defined networking,” in Proceedings of the 1st Workshop on Hot Topics in Sofware Defined Networks (HotSDN ’12), pp. 127–132, ACM, Helsinki, Finland, August 2012.

[15] E Al-Shaer, Q. Duan, and J. H. Jafarian, “Random host mutation for moving target defense,” in Proceeding of the SecureComm, pp. 310–327, 2012.

[16] D. C. MacFarland and C. A. Shue, “Te SDN shufe: creating a moving-target defense using host-based software-defined networking,” in Proceedings of the 2nd ACM Workshop on Moving Target Defense, MTD 2015, pp. 37–41, USA, 2015.

[17] R. Skowyra, K. Bauer, V. Dedhia, and H. Okhravi, “Have No PHEAR: networks without identifiers,” in Proceedings of the 2016 ACM Workshop on Moving Target Defense, MTD 2016, pp. 3–14, Austria, 2016.

[18] J. Sun and K. Sun, “DESIR: decoy-enhanced seamless IP randomization,” in Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016, pp. 1–9, April 2016.

[19] J. H. H. Jafarian, E. Al-Shaer, and Q. Duan, “Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers,” in Proceedings of the 1st ACM Workshop on Moving Target Defense (MTD ’14), pp. 69–78, Scottsdale, AZ, USA, November 2014.

[20] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “Adversary-aware IP address randomization for proactive agility against sophisticated attackers,” in Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015, pp. 738–746, Hong Kong, May 2015.

[21] C. Lei, H.-Q. Zhang, D.-H. Ma and Y.-J. Yang, “Network moving target defense technique based on self-adaptive end-point hopping,” Arabian Journal for Science and Engineering, vol. 42, no. 8, pp. 3249–3262, 2017.

[22] L. Cheng, M. Duo-He, Z. HongQi, Y. YingJie, and W. Li-Ming, “Moving target defense technique based on network attack surface self-adaptive mutation,” Chinese Journal of Computers, vol. 40, no. 130, 2017.

[23] Wang, S., Zhang, L., & Tang, C. A new dynamic address solution for moving target defense. 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. DOI:10.1109/itnec.2016.7560545.

[24] Gudla, C., & Sung, A. H. (2020, November). Moving Target Defense Application and Analysis in Software-Defined Networking. In 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 0641-0646). IEEE.