MTD Analysis and evaluation framework in Software Defined Network (MASON)

Ankur Chowdhary
Arizona State University
Tempe, AZ
achaud16@asu.edu

Adel Alshamrani
Arizona State University
Tempe, AZ
aalsham4@asu.edu

Dijiang Huang
Arizona State University
Tempe, AZ
dijiang@asu.edu

Hongbin Liang
Southwest Jiatong University
Chengdu, China
hbliang@swjtu.edu.cn

ABSTRACT

Security issues in a Software Defined Network (SDN) environment like system vulnerabilities and intrusion attempts can pose a security risk for multi-tenant network managed by SDN. In this research work, Moving target defense (MTD) technique based on shuffle strategy - port hopping has been employed to increase the difficulty for the attacker trying to exploit the cloud network. Our research work *MASON*, considers the problem of multi-stage attacks in a network managed using SDN. SDN controller can be used to dynamically reconfigure the network and render attacker's knowledge in multi-stage attacks redundant. We have used a threat score based on vulnerability information and intrusion attempts to identify Virtual Machines (VMs) in systems with high-security risk and implement MTD countermeasures port hopping to assess threat score reduction in a cloud network.

CCS CONCEPTS

• **Networks → Network experimentation**; • **Computer systems organization → Cloud computing**; • **Security and privacy →** *Security requirements*; *Vulnerability scanners*;

KEYWORDS

Software Defined Networking (SDN), Intrusion Detection System (IDS), Moving Target Defense (MTD)

ACM Reference Format:
Ankur Chowdhary, Adel Alshamrani, Dijiang Huang, and Hongbin Liang. 2018. MTD Analysis and evaluation framework in Software Defined Network (MASON). In *SDN-NFV Sec'18: 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, March 19–21, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3180465.3180473

1 INTRODUCTION

Software Defined Networking (SDN) provides centralized command and control (C&C) for a cloud network. Network-wide visibility and network device programmability are some key benefits offered by SDN. SDN is ideally suited for centralized network monitoring and security enforcement.

The SDN controller can exercise direct control over the state of data plane elements using application programming interface (API) such as OpenFlow [10]. Applications running on the controller can be used to correlate threats from different security assessment tools and take necessary action as noted in *et al* [18].

NICE [3] framework has utilized attack graph based security assessment and countermeasure selection using SDN in order to deal with active security threats and vulnerabilities in the cloud network. The assumption in such security model is that the cloud services are static in nature, i.e., once deployed, configuration remains same. This makes a reconnaissance of critical services easy for the attackers. The attackers can target vulnerabilities on cloud edge machines, and use them as a stepping stone for targeting core network resources such as de-militarized zone (DMZ). Such attacks are known as multi-stage attacks.

Moving Target Defense (MTD) has emerged as a good solution to deal with security issues associated with the static cloud environments. The goal of MTD is to reduce the attack surface (network services, OS information, vulnerabilities) available to the attackers [21]. The scope of pro-actively making changes to the cloud network makes SDN as an ideal candidate for deploying MTD in the cloud network. SDN can ensure network reconfiguration and traffic rerouting with limited service interruption.

An ad-hoc approach of switching services and connections in the network can prove to be more catastrophic than useful. Randomization of IP address [4] [7] may not scale well on a large cloud network. Other SDN based techniques for MTD that consider system resources, network bandwidth, cost-benefit analysis while taking MTD decision, are either purely analytic in nature or use evaluation metrics agnostic of the security state of the network (lack of situation awareness) [9].

While some solutions use vulnerability assessment [5] [21] for evaluating the effectiveness of MTD, only vulnerability scores may not be enough for taking MTD countermeasure. We need an MTD solution based on real-world security metrics. In this research work, we propose a security assessment framework *MASON* based on the system vulnerabilities and active intrusion events. We use MASON framework to calculate *threat score* and deploy MTD countermeasures for the services and virtual machines with high security risk using SDN controller. We utilize both active (intrusion detection) and passive (vulnerability) information for analyzing MTD effectiveness. We have not considered aspects such as change in security policies due to MTD, as discussed in [16], [6] as a part of this research work.

The key contribution for our research are as follows; 1) We formulate a threat scoring system based on vulnerabilities and IDS alerts to select MTD countermeasure. 2) We examine the impact of MTD countermeasure port hopping on threat score 3) We deploy MTD solution on an real world SDN cloud testbed - Science DMZ [1].

The rest of the paper is organized as follows. Section 2 discusses background details about MTD methods used in this work and multi-stage attacks in detail. Section 3 presents the system architecture and physical setup details of MTD testbed Science DMZ. In Section 4, we describe the implementation and experimental analysis details that we performed to study results obtained on SDN testbed. The related work has been discussed in Section 5. Finally, we discuss the limitations of our approach and conclude the research work in Section 6.

2 BACKGROUND

MTD techniques that can be deployed in a cloud network can be broadly classified into *Shuffle* [7], *Diversity* [11] and *Redundancy* [5] as described in the Table 1 below.

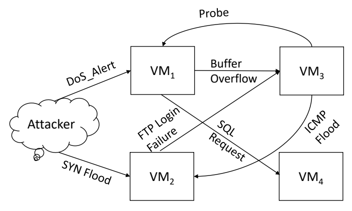
Table 1: MTD Techniques

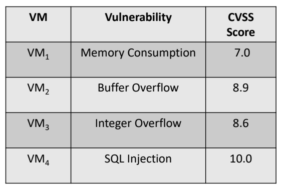
MTD Cateogory	Description
Shuffle	Continuous change of port number of a particular service (port hopping) or IP address of a VM (IP hopping), VM Migration.
Diversity	Deploying functionally equivalent variant of a web application, program or a VM.
Redundancy	Creating multiple replicas/decoys of a service or VM.

In this paper, we focus on shuffle based MTD techniques that can be implemented at a network level. Some research works such as Clark *et al* [4] have utilized MTD techniques such as IP randomization to deal with timing and fingerprint based attacks, but the impact on legitimate users due to randomization has not been evaluated. Zhu *et al* [20] use MTD in a multi-stage attack similar to our work, but they lack realistic reward values in the proposed game theoretic approach.

Hong *et al* [5] have analyzed cost-benefit of MTD based on attack graphs and attack trees. The MTD strategy proposed is based on vulnerabilities present on the system. Jia *et al* [8] proposed redundancy techniques to deal with the dynamic attack such as distributed denial of service (DDoS). There is no framework however that considers both vulnerabilities present in the system and intrusion events such as buffer overflow, DDoS attacks affecting various VMs in the cloud while taking MTD decision.

2.1 Threat Model





VM	Vulnerability	CVSS Score
VM$_1$	Memory Consumption	7.0
VM$_2$	Buffer Overflow	8.9
VM$_3$	Integer Overflow	8.6
VM$_4$	SQL Injection	10.0

a) Intrusion Events reported by IDS b) Vulnerabilities reported by scanner

Figure 1: Use Case Diagram

We consider a cloud system with virtual machines running various services such as *ftp, ssh, http*. There can be vulnerabilities on the services, e.g., Openssh 5.4 had vulnerabilities *CVE-2016-0777* and *CVE-2016-0778*, which lead to leakage of private information.

A cloud system is shown in Fig. 1 has been used as an example in our threat model. Following assumptions are used for the example:

- Vulnerabilities (static events) are present on VMs, e.g., memory consumption vulnerability on VM_1 Fig 1a. Each vulnerability has a CVSS [12] score, which indicates the severity of a vulnerability.
- Intrusion attempts are reported by IDS on the cloud system, e.g., DDoS attempt on VM_2. Each intrusion event has severity *LOW*, *MEDIUM* or *HIGH* based on Snort [17] IDS rules.
- VM_1 and VM_2 are edge nodes (VMs) in the cloud network that are directly accessible to attacker located on the Internet.
- The goal of the attacker is to launch multi-stage attack and compromise core nodes VM_3 and VM_4

2.2 Event Correlation and Threat Ranking

DEFINITION 1. *We define an attack graph* $G = \{S, A, \rightarrow, I\}$, *where S represent the states of the system, e.g., memory consumption vulnerability exploited, A denotes attacker's actions, e.g., SQL Inject attempt,* $\rightarrow \subseteq S \times A \times S$ *is transition relation, e.g., user to root on* VM_4, *AP depicts set of all atomic propositions, I is initial privilege of the attacker, i.e., user on* VM_1 *and* VM_2.

At each state the actions $\alpha \subseteq A$ are probabilistically distributed over the normalized threat severity of all possible actions in that state, such that $\forall s \in S$, $\exists \alpha \in A$, s.t. $\sum_{a \in \alpha} \alpha = 1$.

Our threat scoring scheme utilizes ranking scheme similar to Google's *Page Rank* algorithm [14]. The PageRank algorithm, which is based on user behavior, assumes there is a "random surfer" who visits web pages until he gets bored. Once the random surfer gets bored, he can click on any random page.

To capture the notion of randomness, a damping factor d is used, where $0 < d < 1$. The factor $1 - d$ denotes random transitions from a given state to all possible states in a web graph consisting of web pages as graph nodes and web links as edges.

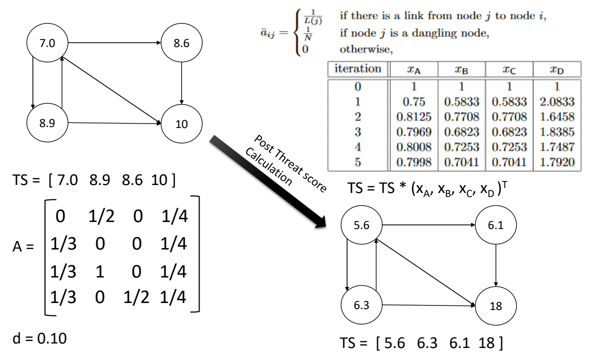
If the graph consists of *N* nodes (web pages), Let *In(j)* be set of pages linking to web page *j* and *Out(j)* be set of outlinks from page *j*.

The probability of random surfer being at page i is given by influence matrix

$$x_i = \frac{1-d}{N} + d \sum_{j \in In(i)} \frac{x_j}{|Out(j)|} \tag{1}$$

For instance, VM_1 in Fig. 1, has $x_1 = \frac{1-0.1}{4} + 0.1 \times \frac{x_3}{2}$, since $In(i) = x_3$ and |Out(j)| = 2 for VM_3. Similarly, $x_3 = \frac{1-0.1}{4} + 0.1 \times \{\frac{x_1}{2} + \frac{x_2}{1}\}$. Equation 1 is computed recursively until x_i converges. Let PageRank vector be $PR = (pr_1, pr_2, pr_3, ..., pr_N)^T$, where rank of page i is pr_i. The PageRank of page i is defined by the probability x_i.

In the threat scoring, the event probabilities are assigned to attacker's actions, and the probability for the system is computed until the values converge to a steady state. The parallels between PageRank random surfer model and using a similar approach in attack graph ranking is justified by the fact that brute force methods such as Distributed Denial of Service (DDoS) use sequential probing of various network services. We consider *Threat Score (TS)* as analogous to PageRank in our model.



$$\bar{a}_{ij} = \begin{cases} \frac{1}{L(j)} & \text{if there is a link from node } j \text{ to node } i, \\ \frac{1}{N} & \text{if node } j \text{ is a dangling node,} \\ 0 & \text{otherwise,} \end{cases}$$

iteration	x_A	x_B	x_C	x_D
0	1	1	1	1
1	0.75	0.5833	0.5833	2.0833
2	0.8125	0.7708	0.7708	1.6458
3	0.7969	0.6823	0.6823	1.8385
4	0.8008	0.7253	0.7253	1.7487
5	0.7998	0.7041	0.7041	1.7920

$$TS = TS * (x_A, x_B, x_C, x_D)^T$$

$$A = \begin{bmatrix} 0 & 1/2 & 0 & 1/4 \\ 1/3 & 0 & 0 & 1/4 \\ 1/3 & 1 & 0 & 1/4 \\ 1/3 & 0 & 1/2 & 1/4 \end{bmatrix}$$

Figure 2: Use Case Diagram

The initial TS of nodes in our threat model is the vulnerability score defined in Fig. 1, i.e., $TS = (7.0, 8.9, 8.6, 10)$. The influence matrix x on convergence as shown in Fig. 2 of threat scoring algorithm is multiplied with the initial threat score $TS \times x$ to calculate the threat score of a given node. The influence matrix due to various in-links on nodes in the graph converge to $x = (0.79, 0.70, 0.70, 1.79)^T$, where symbol T represents matrix transpose. This shows that there are more chances of VM_4, in Fig. 1, being exploited due to more intrusion events. We multiply this with TS, i.e, $TS = (7.0, 8.9, 8.6, 10) \times (0.79, 0.70, 0.70, 1.79)^T$ to achieve $TS \approx (5.6, 6.3, 6.1, 18)$.

We conducted an empirical study of vulnerability distribution for about 100 services and effect of threat events on those services, as shown in Fig. 3. In the first study, virtual machines were allocated services with low vulnerabilities and custom attack scripts were used to perform network and service level attack such as denial of service attack, SYN Flood attack, etc., as shown in left part of Fig. 3. In another study, we simulated virtual machines with high CVSS score on the virtual machines but used intrusion events with low severity as shown in the right part of the figure. The goal was to observe the effect of high severity intrusion events vs high severity vulnerabilities.

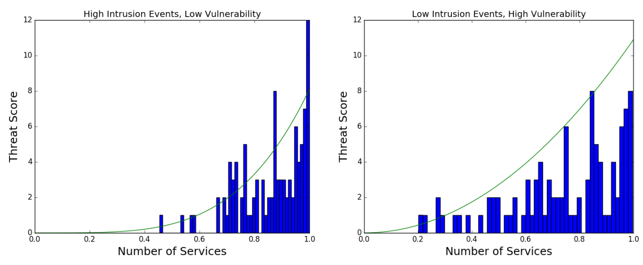


Figure 3: Threat Events vs Vulnerability Evaluation

The services with low vulnerability but high threat events have higher threat score, compared to services with low intrusion events but high vulnerability score. This leads to the conclusion that security models such as [5] and MTD measures that are purely based on vulnerability assessment for the network are not enough for taking MTD decision. Thus we use both security vulnerabilities and intrusion events into account using threat scoring mechanism before taking MTD countermeasure.

3 SYSTEM DESIGN AND ARCHITECTURE

3.1 Science DMZ: SDN enabled Secured Cloud Testbed

We established a secured MTD based testbed to evaluate Internet 2 backbone network traffic for the ASU campus network. The Science DMZ testbed consists of SDN based security command and control center.

We use Opendaylight based SDN controller and PHP lavarel web framework as front-end. The Openstack is used for compute and network resource provisioning on Physical servers. The VMs of physical servers are managed and controlled by SDN controller via Open vSwitch.

Science DMZ SDN testbed that we use for experimental analysis consists of four Dell R620 servers and two Dell R710 servers all hosted in the ASU data center. Each Dell R620 has 128 GB of RAM, 1.5 TB hard disk storage and 16 core CPU. Each Dell R710 has 144 GB RAM, 1 TB disk storage and core CPU.

We use OpenFlow enabled hardware switch (HP OpenFlow 1.3) to connect physical servers together. One physical server acts as gateway and provides proxy services to important components of Science DMZ like Openstack Cloud, SDN controller, etc. The SDN controller Opendaylight has been used for network management and orchestration.

The Science DMZ GUI uses the REST API to manage SDN controller, Openstack backend, and other network segments. The Snort agent has been used as Network IDS (NIDS) and NESSUS has been used for vulnerability scanning.

3.2 System Architecture

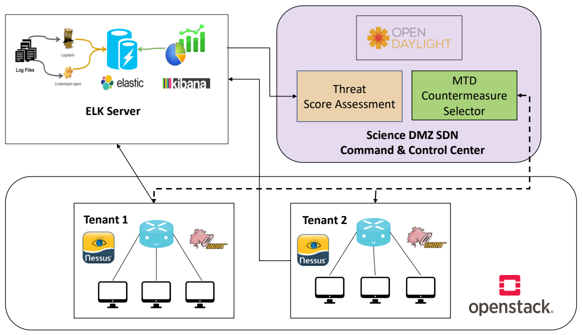


Figure 4: System Architecture

Fig. 4 shows system architecture. Key components have been described below:

(1) **Snort IDS and Nessus agents** are used inside each tenant for signature-based intrusion detection and vulnerability assessment in cloud network. The alert information on various services and network devices is sent to the ELK Server ash shown in Fig. 4. If the vulnerability is already known *Vulnerability Scanner (NESSUS)* listens for any network topology or flow rule updates from SDN command and control center. Thus, we check only new vulnerabilities in case of network services or system updates.

(2) **ELK Server** collects system and application logs from VMs. We plan to do a correlation of IDS alerts, ELK (Elastic Search, Logstash, Kibana) logs, and system vulnerabilities to perform zero-day attack detection in the near future. Currently, this component collects and stores logs from various services for security analysis.

(3) **SDN Command and Control Center** is used for monitoring the control plane traffic for SDN environment. SDN controller ODL interacts with log analyzer to make informed security decision regarding port hopping. SDN Controller uses Threat score to implement SDN countermeasures.

The real-time analysis in the system ensures that vulnerability scanner and IDS are aware of any network topology reconfigurations and continuously monitor network services. If the risk score is above a certain threshold, we check the effectiveness of the particular countermeasure on the system given the level of risk.

4 IMPLEMENTATION AND EVALUATION

In this section, we discuss the implementation algorithm and experiments we conducted on Science DMZ testbed to analyze network threat score and the effectiveness of MTD countermeasures in decreasing the threat score discussed in previous sections. We use the terms threat score and risk value interchangeably in following sections.

Algorithm 1: NETWORK-THREAT-SCORING

```
   Input: $G = \{N, E\}$, d=0.1
   Output: Emit- TS(N) : Threat Score on Algo. Termination
1  A: Link Matrix
2  $a_{ij} \in A \leftarrow \frac{1}{L(j)}$ : For link from node j to i
3  $a_{ij} \leftarrow \frac{1}{N}$ : For dangling node j
4  $a_{ij} \leftarrow 0$ : Otherwise
5  $x \leftarrow (1, 1, 1, ..)$ : Probability Matrix
6  $k \leftarrow 0$
7  $\epsilon \leftarrow 1 \times 10^{-3}$ : Algo. stopping criteria
8  TS(N) : Initial threat score of nodes
9  while $|x_{k+1} - x_k| \leq \epsilon$ do
10     $x_{k+1} \leftarrow \frac{1-d}{N} + dA^T x_k$
11     $k \leftarrow k + 1$
12 return $x$
13 $TS = TS \times x^T$
```

We use NETWORK-THREAT-SCORING algorithm 1 to assess the threat score of software vulnerabilities and intrusion events on each service running on a physical server. The algorithm takes Graph $G = \{N, E\}$ of vulnerabilities and threat events as input. Lines 2-4 describe the initial value of link matrix A. The algorithm updates influence matrix x values - line 10,11 till the difference compared to previous influence matrix value $x_{k+1} - x_k$ is very small, i.e, $\leq \epsilon$. The influence matrix value is multiplied with initial threat score TS to calculate final threat score - line 13.

We conducted following experiments on Science DMZ MTD testbed.

- **Experiment 1** Network threat score analysis based on software vulnerabilities and IDS alerts to prioritize most vulnerable services. This experiment allowed us to prioritize candidates for MTD countermeasure.
- **Experiment 2** Study of MTD countermeasures port hopping. We used effective network *threat score* to check attacker's reward with varying number of services and VMs selected for MTD countermeasures.

Experimental setup consisted of virtual machines - Windows 7, metasploitable, CentOS, Ubuntu Server 14.04 with varying number of services on two tenants that we created on the Openstack cloud network. Each tenant was connected to one OpenFlow switch. The Open vSwitch rules can be updated by Opendaylight controller present in command and control center.

4.1 Experiment 1: Network Threat Score Analysis

We used vulnerability scanning to check software and system vulnerabilities on the services running on the network. We combined this information with the active attack information from NIDS alerts to calculate the service threat score $TS(s_i)$ for all the services running on the network.

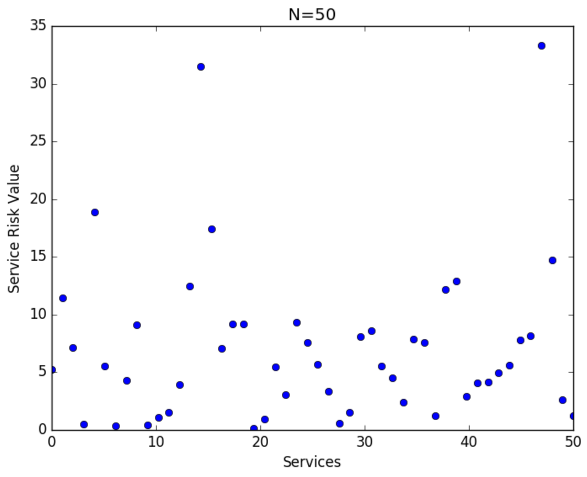


Figure 5: Threat Score vs N=50 Number of Services

Fig. 5 shows the service threat score for a network with 50 services on two tenants. The threat score accounts for the events and vulnerabilities on a particular service, e.g., FTP server and influence of services connected directly or indirectly (via a reachable path) to FTP server. The figure depicts the distribution of threat scores for various services.

We can use this information to set a threshold for triggering MTD countermeasure. It can be seen that in a network with 50 services, about 10-20 percent services have a threat score above 10, so we can deploy the MTD port hopping (discussed in next experiment) for 10-20 percent services. We conducted another experiment to assess threat score of a network with about 200 services running Fig. 6. We observed that very few services 5 telnet, rlogin, etc. had a very high threat score (above 80). This is because of the high centrality of these services in the network so they have reachability from most services along multi-stage attack paths. The average threat score of this network is higher than a network with 50 services since attacker gets more options for exploration and exploitation of network resources. We can prioritize deployment of MTD countermeasure for 10-15 percent of services for this network to have a greater influence on the overall network security threat.

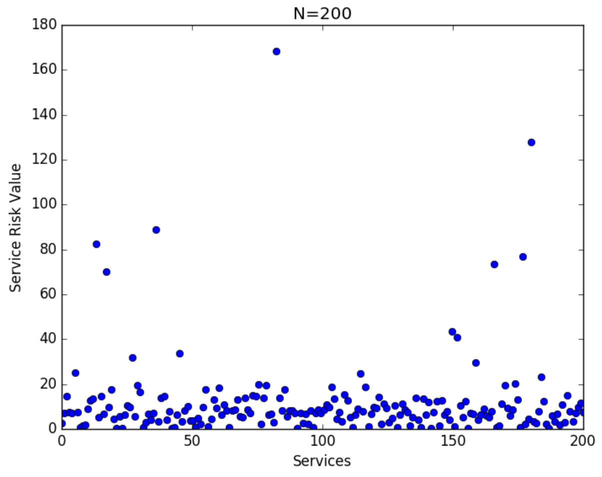


Figure 6: Threat Score vs N=200 Number of Services

The information from this experiment can be utilized for assessing the threat level in a network and forecasting the network security budget to maximize the security coverage of the overall network.

4.2 Experiment 2 MTD Countermeasure Port Hopping

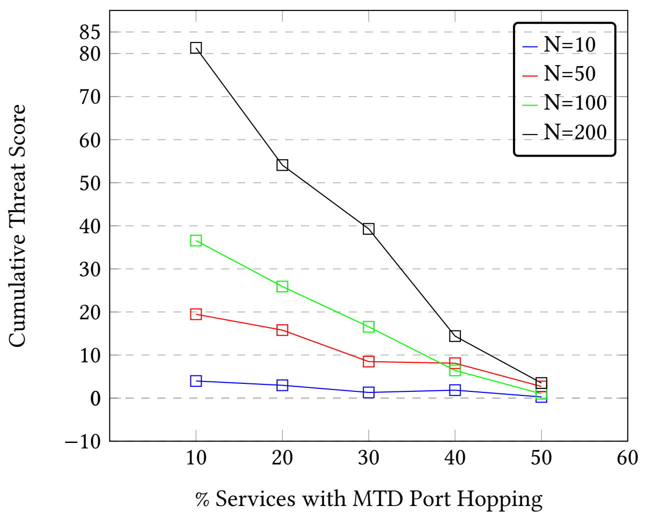


Figure 7: Threat Score vs MTD port hopping

We used effective *threat score* a for multi-stage attack on an SDN network with a number of services ranging from N=10 in the first experiment to N=200 in the last experiment. The number of services selected for the deployment of MTD countermeasure port hopping was varied from 10 percent to 50 percent as shown in Fig. 7. We took the *threat score* of an average of three runs for each experiment. It is notable that MTD countermeasure port hopping is not very much effective on a small sized network with reward value 3.98 for an attacker on the deployment of countermeasure for 10 percent nodes to 0.28 for 50 percent nodes.

As the number of nodes/services on the network increase, the threat score increases as well. For a network with 50 nodes higher risk obtained for the network is 19.47, whereas for a network of N=200 nodes highest threat obtained is 81.32 as can be seen in Fig. 7. This can be interpreted by the fact that the exploitation surface has many possible options and attack paths.

We can observe the effectiveness of MTD based port hopping on a large network with N=200 nodes, where the threat score decreases rapidly from 81.32 to 10 percent nodes with MTD based port hopping to 3.51 for 50 percent nodes - Fig. 7 with MTD port hopping implemented. This is because the port hopping increases the attack surface by a large amount, hence the strategy of the attacker to first discover important nodes in the network and then target using malicious attack traffic is rendered useless.

5 RELATED WORK

We have analyzed some of the related works that consider risk/ threats (situation awareness) in the network while deploying MTD countermeasures. Various MTD techniques discussed by Xu *et al* [19] and Okhravi *et al*[13] for the effectiveness of security attacks to select a defense strategy which was studied as a starting point for this work. A risk-aware MTD strategy has been discussed by Peng *et al*[15]. The authors model attack surface as a non-decreasing probability density function and estimate the risk of migrating a VM to a replacement node using probabilistic inference. The risk assessment can be improved using our threat scoring framework. Kampanakis *et al*[9] have discussed port hopping as a possible MTD strategy to deal with attacks in SDN environment. Authors have discussed OS fingerprinting and network reconnaissance as specific types of attacks in SDN. Random mutations of this nature may, however, disrupt any active services and some cost-benefit analysis of MTD strategy is necessary.

A system design to support intelligent MTD adaptations and its effectiveness compared to a static system have been discussed by Zhuang *et al*[22]. SDN based scalable MTD has been discussed in [2]. The paper utilizes vulnerability scores for creation of scalable attack graph in a cloud network. The countermeasure selection part has not been discussed in detail in this work. We use dynamic and static attack information to select port hopping as a MTD countermeasure. HARM[5] provides an attack graph based security modeling technique for the assessment of Moving Target Defense. The authors use network connections between VMs to create attack graph and relationship between various service vulnerabilities to construct an attack tree for network security analysis, however, attack model is based on vulnerability scores alone. In this work we consider active IDS alerts and vulnerabilities to take effective countermeasures.

6 CONCLUSION

This paper presents MASON, a framework for the assessment of cloud network vulnerabilities and intrusion events. MASON utilizes a threat scoring method based on PageRank algorithm to identify network services with high-security risk and takes corresponding MTD countermeasure port hopping. Our evaluation results show that situational awareness based on static vulnerability information and dynamic threat events should be used for taking MTD decisions. There is a marked reduction in the threat score of the network when we employ port hopping even on a small number of services. The difference is more prominent in a large size cloud network. The limitation of this approach is that Quality of Service (QoS) has not been accounted for while taking MTD decision. We plan to utilize parameters such as network throughput and service latency while evaluating the effectiveness of MTD as an extension of this work. The approach is based on signature-based detection methods and doesn't account for security issues such as Advanced Persistent Threats (APT's), which don't have an associated signature match. We plan to utilize machine learning methods that use OpenFlow statistics to identify APT scenarios and take proactive countermeasures using SDN controller.

ACKNOWLEDGMENTS

This research is based upon work supported by the National Science Foundation under Grant 1642031, NSF Secure and Resilient Networking Project under Grant 1528099 and NSFC under Grants 61628201 and 61571375. Special thanks to the UTO team at ASU for their continuous support.

REFERENCES

[1] A. Chowdhary, V. H. Dixit, N. Tiwari, S. Kyung, D. Huang, and G.-J. Ahn. Science dmz: Sdn based secured cloud testbed.

[2] A. Chowdhary, S. Pisharody, and D. Huang. Sdn based scalable mtd solution in cloud network. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, pages 27–36. ACM, 2016.

[3] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. Nice: Network intrusion detection and countermeasure selection in virtual network systems. *IEEE transactions on dependable and secure computing*, 10(4):198–211, 2013.

[4] A. Clark, K. Sun, and R. Poovendran. Effectiveness of ip address randomization in decoy-based moving target defense. In *52nd IEEE Conference on Decision and Control*, pages 678–685. IEEE, 2013.

[5] J. B. Hong and D. S. Kim. Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing*, 13(2):163–177, 2016.

[6] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao. Flowguard: building robust firewalls for software-defined networks. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 97–102. ACM, 2014.

[7] J. H. Jafarian, E. Al-Shaer, and Q. Duan. 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*, pages 127–132. ACM, 2012.

[8] Q. Jia, K. Sun, and A. Stavrou. Motag: Moving target defense against internet denial of service attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2013.

[9] P. Kampanakis, H. Perros, and T. Beyene. Sdn-based solutions for moving target defense network protection. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–6. IEEE, 2014.

[10] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[11] A. Newell, D. Obenshain, T. Tantillo, C. Nita-Rotaru, and Y. Amir. Increasing network resiliency by optimally assigning diverse variants to routing nodes. *IEEE Transactions on Dependable and Secure Computing*, 12(6):602–614, 2015.

[12] NIST. CVSS. https://www.first.org/cvss, 2016. [Online; accessed 19-Nov-2016].

[13] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein. Survey of cyber moving target techniques. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2013.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

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

[16] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang. Brew: A security policy analysis framework for distributed sdn-based cloud environments. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2017.

[17] M. Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[18] S. Scott-Hayward, G. O'Callaghan, and S. Sezer. Sdn security: A survey. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*, pages 1–7. IEEE, 2013.

[19] J. Xu, P. Guo, M. Zhao, R. F. Erbacher, M. Zhu, and P. Liu. Comparing different moving target defense techniques. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 97–107. ACM, 2014.

[20] Q. Zhu and T. Basar. Feedback-driven multi-stage moving target defense. In *Proc. Conf. Decision Game Theory Security.*

[21] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40. ACM, 2014.

[22] R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou, and A. Singhal. Investigating the application of moving target defenses to network security. In *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*, pages 162–169. IEEE, 2013.