International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

An Efficient Approach of Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems

Shaik Shanawaz[1], K. Thyagarajan[2]

[1]M.Tech, Sri Venkateswara College of Engineering and Technology,
RVS nagar, Chittoor, Andhra Pradesh, India

[2]Associate Professor, M.Tech, Sri Venkateswara College of Engineering and Technology,
RVS nagar, Chittoor, Andhra Pradesh, India

Abstract: ***Intrusion Detection and Prevention Systems (IDPS) are used: to identify possible attacks, collecting information about them and the trying to stop their occurrence and at last reporting them to the system administrator. These systems are used by some organizations to detect the weaknesses in their security policies, documenting existing attacks and threats and preventing an individual from violating security policies. Because of their advantages these systems became an important part of the security infrastructure in nearly every organization. In a Cloud computing environment, attackers can determine the vulnerabilities in the cloud systems and compromise the virtual machines to set out large scale Distributed Denial-of-Service (DDOS) attack. To avert these virtual machines from concession, we propose a multi-phase solution E-NICE (An Efficient Network Intrusion Detection and Countermeasure selection in Virtual Network Systems using defense in depth frame work).***

Keywords: Network Security, Cloud Computing, Intrusion Detection, Attack Graph, Zombie Detection

1. Introduction

Recent studies have shown that users trekking to the cloud consider security as the most important factor. A recent Cloud Security Alliance (CSA) survey shows that among all security issues, misuse and outrageous use of cloud computing is considered as the top security threat in which attackers can use susceptibilities in clouds and utilize cloud system schemes to utilize attacks. In traditional data centers, where system administrators have full commandl over the host machines, susceptibilities can be detected and covered by the system administrator in a rationalized manner. However, covering known security slits in cloud data centers, where cloud users usually have the advantage to control software placed on their managed VMs, may not work effectually and can violate the *Service Level Agreement* (SLA). Furthermore, cloud users can set up breakable software on their VMs, which essentially accords to loop slits in cloud security. The challenge is to establish productive susceptibility/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users. In a cloud system where the infrastructure is shared by potentially millions of users, misuse and outrageous use of the shared infrastructure benefits attackers to use susceptibilities of the cloud and use its scheme to utilize attacks in more systematic ways. Such attacks are more productive in the cloud territory since cloud users usually share computing schemes, e.g., being attached through the same control, sharing with the same data storage and file systems, even with potential attackers . The same setup for VMs in the cloud e.g., virtualization techniques, VM OS, placed breakable software, networking, etc., tempts attackers to compromise multiple VMs. This project propose E-NICE (An Efficient Network Intrusion detection and Countermeasures Selection in virtual network systems) to establish a defense-in-depth intrusion detection structure. For better attack detection, E-NICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of E-NICE does not intend to improve any of the existing intrusion detection algorithms; indeed, E-NICE utilizes a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus intercepting zombie VMs. In general, E-NICE incorporates two main phases: (1) utilize a lightweight mirroring-based network intrusion detection agent [E-NICE (A)] on each cloud server to capture and analyze cloud traffic. E-NICE (A) periodically scans the virtual system susceptibilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of associated susceptibility towards the collaborative attack goals; E-NICE will decide whether or not to put a VM in network investigation state. (2) Once a VM enters investigation state, Deep Packet Investigation (DPI) Digital Object Identifier.

2. Present Work

This project propose E-NICE (An Efficient Network Intrusion detection and Countermeasure Selection in virtual network systems) to establish a defense-in-depth intrusion detection structure. For better attack detection, E-NICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of E-NICE does not intend to improve any of the existing intrusion detection algorithms; indeed, E-NICE utilizes a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus intercepting zombie VMs.

2.2 Advantages of Present Work

The contributions of E-NICE are presented as follows:

- We devise E-NICE, a new multi-phase distributed network intrusion detection and interception structure in a virtual networking territory that captures and inspects suspicious cloud traffic without interrupting users' applications and cloud services.
- E-NICE incorporates a software controlling solution to quarantine and inspect suspicious VMs for further investigation and protection. Through guidable network approaches, E-NICE can improve the attack detection probability and improve the resiliency to VM use attack without interrupting existing normal cloud services.
- E-NICE utilizes a novel attack graph approach for attack detection and interception by correlating attack behavior and also suggests productive countermeasures.
- E-NICE optimizes the execution on cloud servers to minimize scheme consumption. Our study shows that E-NICE consumes less computational overhead collated to proxy-based network intrusion detection solutions.

3. Defense-In-Depth Intrusion Detection Framework in Virtual Network System

Network and security management has to assure uninterrupted access to the communication infrastructure. With growing networks and increasing amount of transported data, it gets more and more complicated to supervise the operation of the communication systems. Sometimes computer networks are not well protected against attacks from the outside, so additional surveillance may be necessary. But even well protected networks need surveillance. A lot of these networks are threatened from the inside. Intrusion Detection Systems (IDSs) help securing these networks. This paper focuses on a tool for visualizing and detecting anomalies of the traffic structure. Several distributed Denial of Service attacks have shown the necessity of better protecting computers and networks connected to the Internet. Due to widely available attack tools, attacks of this kind can be carried out by persons without in-depth knowledge of the attacked system. Insufficient protected Open University networks are an example for networks that need additional surveillance. These networks often include vulnerable computers and offer high bandwidth connections to the Internet. These features are the reason why attackers are interested in these networks. The machines in these networks are not the goal of the attacks. Normally, they do not contain interesting information for the attacker, but they are suitable for scanning other networks and starting (for example) Denial of Service attacks. The Internet is increasingly important as the vehicle for global electronic commerce. Many organize- tins also use Internet TCP/IP protocols to build intra-networks (intranets) to share and disseminate internal information. A large scale attack on these networks can cripple important world-wide Internet operations. The Internet Worm of 1988 caused the Internet to be unavailable for about _vet days Seven years later, there is no system to detect or an- laze such a problem on an Internet-wide scale. The development of a secure infrastructure to defend the Internet and other networks is a major challenge. In this paper, we present the design of the Graph-depends Intrusion Detection System (Girds). Girds' design goal is to analyze network activity on TCP/IP networks with up to several thousand hosts. Its primary function is to detect and anal- lyre large-scale attacks, although it also has the ca- ability of detecting intrusions on individual hosts. Girds aggregates network activity of interest into Activity graphs, which are evaluated and possibly re- ported to a system security ocher (SSO). The hiker- archival architecture of Grids allows it to scale to large networks. Grads is being designed and built by the authors using formal consensus decision-making and a well-documented software process. We have completed the Grids design and have almost _knishes building a prototype. This paper is organized as follows. Brier describes related work on intrusion detection systems and motivates the need for Grids'. Section 1.2 discusses classes of attacks that we expect to detect. In the simple Grids' detection algorithm is described, followed by a more detailed discussion in has a treat- meant of the hierarchical approach to scalability and discusses how the hierarchy is managed. outlines the policy language. Covers some limitations of Girds. Finally, presents conclusions and discusses future work. Network security is a complicated subject, historically only tackled by well-trained and experienced experts. However, as more and more people become ``wired", an increasing number of people need to understand the basics of security in a networked world.

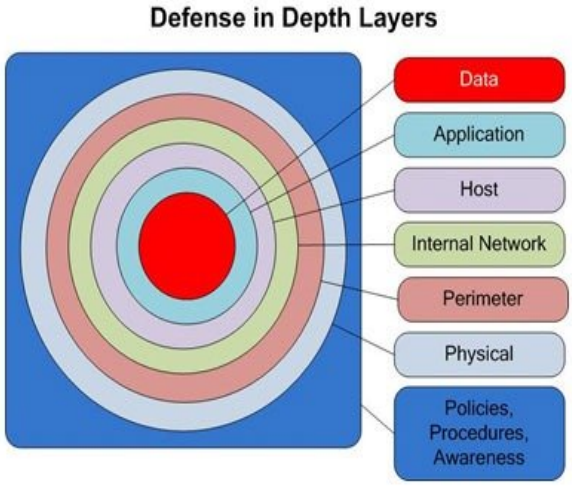


Figure 1: Layers of defense in depth framework

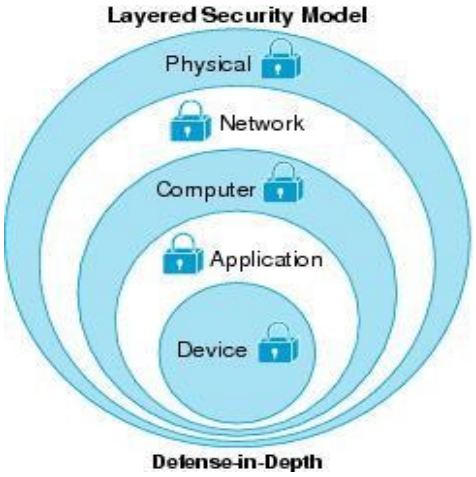


Figure 2: Defense in depth layered security model

4. E-Nice Models

In this section, we describe how to utilize attack graphs to model security threats and susceptibilities in a virtual networked system, and propose a VM protection model depends on virtual network reconfiguration approaches to prevent VMs from being exploited.

4.1 Threat Model

In our attack model, we assume that an attacker can be located either outside or inside of the virtual networking system. The attacker's primary goal is to exploit vulnerable VMs and compromise them as zombies. Our protection model focuses on virtual-network-depends attack detection and reconfiguration solutions to improve there siliency to zombie explorations. Our work does not involve host-depends IDS and does not address how to handle encrypted traffic for attack detections. Our proposed solution can be deployed in an Infrastructure-as-a-Service (IaaS) cloud networking system, and we assume that the Cloud Service Provider (CSP) is benign. We also assume that cloud service users are free to install whatever operating systems or applications they want, even if such action may introduce susceptibilities to their controlled VMs. Physical security of cloud server is out of scope of this project.

We assume that the hypervisor is secure and free of any susceptibilities. The issue of a malicious tenant breaking out of DomU and gaining access to physical server has been studied in recent work and is out of scope of this project.

4.2 Attack Graph Model

An attack graph is a modeling tool to illustrate all possible multi-stage, multi-host attack paths that are crucial to understand threats and then to decide appropriate countermeasures. In an attack graph, each node represents either precondition or consequence of an exploit. The actions are not necessarily an active attack since normal protocol interactions can also be used for attacks. Attack graph is helpful in identifying potential threats, possible attacks and known susceptibilities in a cloud system. Since the attack graph provides details of all known susceptibilities in the system and the connectivity information, we get a whole picture of current security situation of the system where we can predict the possible threats and attacks by correlating detected events or activities. If an event is recognized as a potential attack, we can apply specific countermeasures to mitigate its impact or take actions to prevent it from contaminating the cloud system. To represent the attack and the result of such actions, we extend the notation of MulVAL logic attack graph as presented by X. Ou et al. and define as Scenario Attack Graph (SAG).

Definition 1 (Scenario Attack Graph)**.** An Scenario Attack Graph is a tuple SAG=(V, E), where

1) V = NC ∪ND ∪NR denotes a set of vertices that include three types namely conjunction node NC to represent exploit, disjunction node ND to denote result of exploit, and root node NR for showing initial step of an attack scenario.
2) E = Epre ∪ Epost denotes the set of directed edges. An edge e ∈ Epre ⊆ ND × NC represents that ND must be satisfied to achieve NC. An edge e ∈Epost ⊆ NC × ND means that the consequence shown by ND can be obtained if NC is satisfied. Node vc ∈NC is defined as a three tuple (Hosts, vul, alert) representing a set of IP addresses, susceptibility information such as CVE, and alerts related to vc, respectively. ND behaves like a logical OR operation and contains details of the results of actions. NR represents the root node of the scenario attack graph. For correlating the alerts, we refer to the approach described in [15] and define a new Alert Correlation Graph (ACG) to map alerts in ACG to their respective nodes in SAG. To keep track of attack progress, we track the source and destination IP addresses for attack activities.

Definition 2 (Alert Correlation Graph)**.** An ACG is a three tuple ACG = (A,E, P), where

1) A is a set of aggregated alerts. An alert a ∈ A is a data structure (src, dst, cls, ts) representing source IP address, destination IP address, type of the alert, and timestamp of the alert respectively.
2) Each alert a maps to a pair of vertices (vc, vd) in SAG using map(a) function, i.e., map(a) : a _→{(vc, vd)|(a.src ∈vc.Hosts) ∧(a.dst ∈vd.Hosts) ∧(a.cls = vc.vul)}.
3) E is a set of directed edges representing correlation between two alerts (a, a_) if criteria below satisfied:
   i. (a.ts < a_.ts) ∧(a_.ts − a.ts < threshold)
   ii. ∃(vd, vc) ∈ Epre : (a.dst ∈ vd.Hosts ∧ a_.src ∈ vc.Hosts)
4) P is set of paths in ACG. A path Si ⊂P is a set of related alerts in chronological order.

We assume that A contains aggregated alerts rather than raw alerts. Raw alerts having same source and destination IP addresses, attack type and timestamp within a specified window are aggregated as Meta Alerts. Each ordered pair (a, a_) in ACG maps to two neighbor vertices in SAG with timestamp difference of two alerts within a predefined threshold. ACG shows dependency of alerts in chronological order and we can find related alerts in the same attack scenario by searching the alert path in ACG. A set P is used to store all paths from root alert to the target alert in the SAG, and each path Si ⊂P represents alerts that belong to the same attack scenario.

Algorithm 1 Alert_Correlation

Require: alert a_c, SAG, ACG
1: **if** (a_c is a new alert) **then**
2: create node a_c in ACG
3: $n_1 \leftarrow v_c \in map(a_c)$
4: **for all** $n_2 \in parent(n_1)$ **do**
5: create edge $(n_2.alert, a_c)$
6: **for all** S_i containing a **do**
7: **if** a is the last element in S_i **then**
8: append a_c to S_i
9: **else**
10: create path $S_{i+1} = \{subset(S_i, a), a_c\}$
11: **end if**
12: **end for**
13: add a_c to $n_1.alert$
14: **end for**
15: **end if**
16: **return** S

4.3 VM Protection Model

The VM protection model of E-NICE consists of a VM profiler, a security indexer and a state monitor. We specify security index for all the VMs in the network depending upon various factors like connectivity, the number of susceptibilities present and their impact scores. The impact score of susceptibility, as defined by the CVSS guide, helps judge the confidentiality, integrity, and availability impact of the susceptibility being exploited. Connectivity metric of a VM is decided by evaluating incoming and outgoing connections.

Definition 3 (VM State)**.** Depends on the information gathered from the network controller, VM states can be defined as following:

1) Stable: there does not exist any known susceptibility on the VM.
2) Vulnerable: presence of one or more susceptibilities on a VM, which remains unexploited.
3) Exploited: at least one susceptibility has been exploited and the VM is compromised.
4) Zombie: VM is under control of attacker.

5. Performance Evaluation

In this section we present the performance evaluation of E-NICE. Our evaluation is conducted in two directions: the security performance, and the system computing and network reconfiguration overhead due to introduced security mechanism.

5.1 Security Performance Analysis

To demonstrate the security performance of E-NICE, we created a virtual network testing environment consisting of all the presented components of E-NICE.

5.1.1 Environment and Configuration

To evaluate the security performance, a demonstrative virtual cloud system consisting of public (public virtual servers) and private (VMs) virtual domains is established as shown in Figure 3. Cloud Servers 1 and 2 are connected to Internet through the external firewall. In the Demilitarized Zone (DMZ) on Server 1, there is one Mail server, one DNS server and one Web server. Public network on Server 2 houses SQL server and NAT Gateway Server. Remote access to VMs in the private network is controlled through SSHD (i.e., SSH Daemon) from the NAT Gateway Server. 2 shows the susceptibilities present in this network and 3 shows the corresponding network connectivity that can be explored depends on the identified susceptibilities.

Table 2: Vulnerabilities in the virtual networked system

Host	Vulnerability	Node	CVE	Base Score
VM group	LICQ buffer overflow	10	CVE 2001-0439	0.75
	MS Video ActiveX Stack buffer overflow	5	CVE 2008-0015	0.93
	GNU C Library loader flaw	22	CVE-2010-3847	0.69
Admin Server	MS SMV service Stack buffer overflow	2	CVE 2008-4050	0.93
Gateway server	OpenSSL uses predictable random variable	15	CVE 2008-0166	0.78
	Heap corruption in OpenSSH	4	CVE 2003-0693	1
	Improper cookies handler in OpenSSH	9	CVE 2007-4752	0.75
Mail server	Remote code execution in SMTP	21	CVE 2004-0840	1
	Squid port scan	19	CVE 2001-1030	0.75
Web server	WebDAV vulnerability in IIS	13	CVE 2009-1535	0.76

Table 3: Virtual network Connectivity

From	To	Protocol
Internet	NAT Gateway server	SSHD
	Mail server	IMAP, SMTP
	Web server	HTTP
Web server	SQL server	SQL
NAT Gateway server	VM group	Basic network protocols
	Admin server	Basic network protocols
VM Group	NAT Gateway server	Basin network protocols
	Mail server	IMAP, SMTP
	SQL server	SQL
	Web server	HTTP
	DNS server	DNS

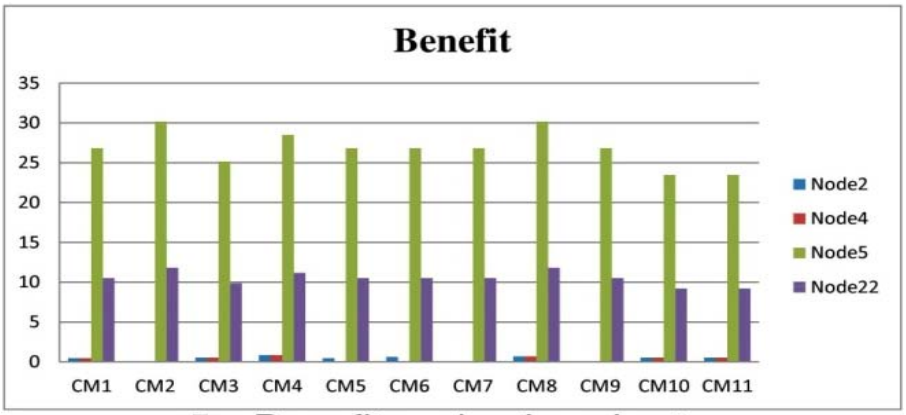
5.1.2 Attack Graph and Alert Correlation

The attack graph can be generated by utilizing network topology and the susceptibility information. As the attack progresses, the system generates various alerts that can be related to the nodes in the attack graph. Creating an attack graph requires knowledge of network connectivity, running services and their susceptibility information. This information is provided to the attack graph generator as the input. Whenever a new susceptibility is discovered or there are changes in the network connectivity and services running through them, the updated information is provided to attack graph generator and old attack graph is updated to a new one. SAG provides information about the possible paths that an attacker can follow. ACG serves the purpose of confirming attackers' behavior, and helps in determining false positive and false negative. ACG can also be helpful in predicting attackers' next steps.

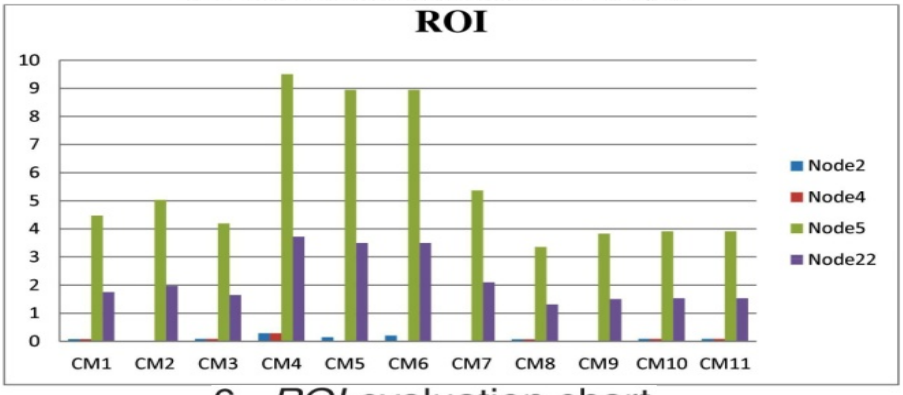
5.1.3 Countermeasure Selection

To illustrate how E-NICE works, let us consider for example, an alert is generated for node 16 (vAlert = 16) when the system detects LICQ Buffer overflow. After the alert is generated, the cumulative probability of node 16 becomes 1 because that attacker has already compromised that node. This triggers a change in cumulative probabilities of child nodes of node 16. Now the next step is to select the countermeasures from the pool of countermeasures CM. If the countermeasure CM4: create filtering rules is applied to node 5 and we assume that this countermeasure has effectiveness of 85%, the probability of node 5 will change to 0.1164, which causes change in probability values of all child nodes of node 5 thereby accumulating to a decrease of 28.5% for the target node 1. Following the same approach for all possible countermeasures that can be applied, the percentage change in the cumulative probability of node 1, i.e., benefit computed using (7) are shown in Figure 5.

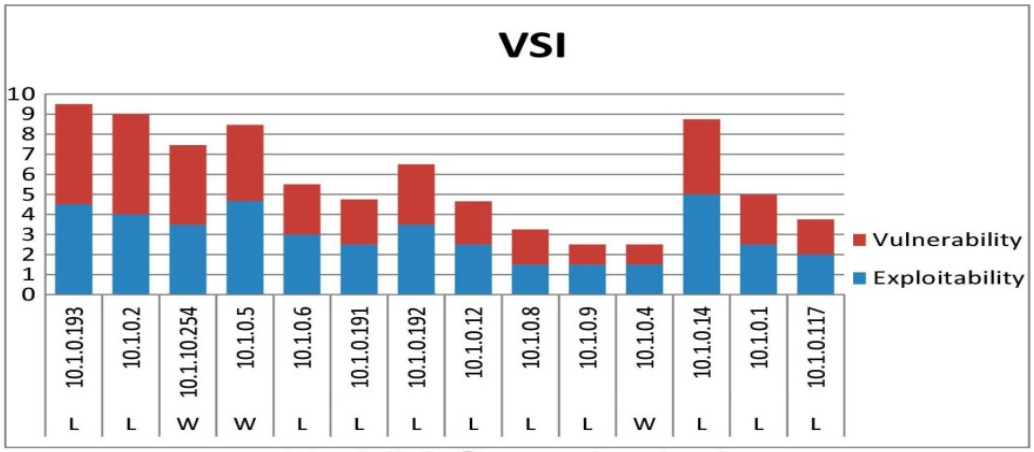
Apart from calculating the benefit measurements, we also present the evaluation depends on Return of Investment (ROI) using (8) and represent a comprehensive evaluation considering benefit, cost and intrusiveness of countermeasure. Figure 6 shows the ROI evaluations for presented countermeasures. Results show that countermeasures CM2 and CM8 on node 5 have the maximum benefit evaluation; however their cost and intrusiveness scores indicate that they might not be good candidates for the optimal countermeasure and ROI evaluation results confirm this. The ROI evaluations demonstrate that CM4 on node 5 is the optimal solution.



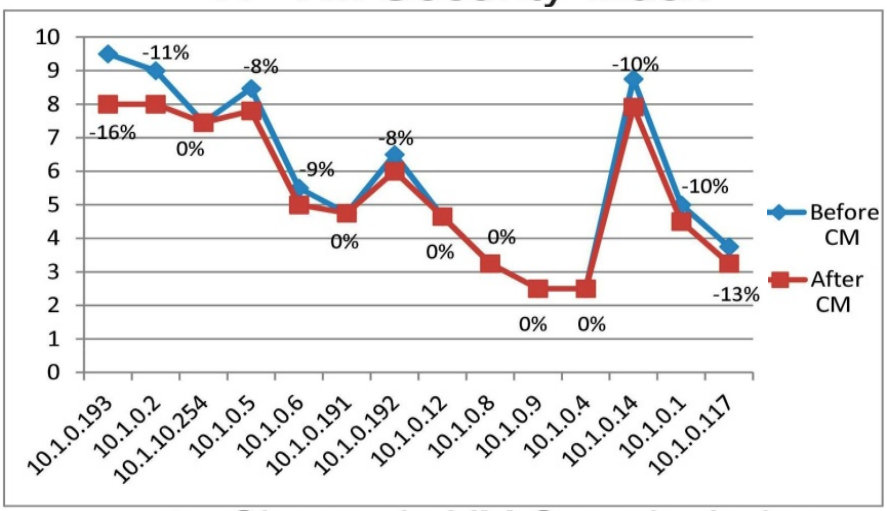
5. *Benefit* evaluation chart.



6. *ROI* evaluation chart.



7. VM Security Index



8. Change in VM Security Index

5.1.4 False Alarms

A cloud system with hundreds of nodes will have huge amount of alerts raised by Snort. Not all of these alerts can be relied upon, and an effective mechanism is needed to verify if such alerts need to be addressed. Since Snort can be programmed to generate alerts with CVE id, one approach that our work provides is to match if the alert is actually related to some susceptibility being exploited. If so, the existence of that susceptibility in SAG means that the alert is more likely to be a real attack. Thus, the false positive rate will be the joint probability of the correlated alerts, which will not increase the false positive rate compared to each individual false positive rate. Moreover, we cannot keep aside the case of zeroday attack where the susceptibility is discovered by the attacker but is not detected by susceptibility scanner. In such case, the alert being real will be regarded as false, given that there does not exist corresponding node in SAG. Thus, current research does not address how to reduce the false negative rate. It is important to note that susceptibility scanner should be able to detect most recent susceptibilities and sync with the latest susceptibility database to reduce the chance of Zero-day attacks.

5.2 E-NICE System Performances

We evaluate system performance to provide guidance on how much traffic E-NICE can handle for one cloud server and use the evaluation metric to scale up to a large cloud system. In a real cloud system, traffic planning is needed to run E-NICE, which is beyond the scope of this project. Due to the space limitation, we will investigate the research involving multiple cloud clusters in the future. To demonstrate the feasibility of our solution, comparative studies were conducted on several virtualization approaches. We evaluated E-NICE depends on Dom0 and DomU implementations with mirroring-depends and proxy-depends attack detection agents (i.e., E-NICE-AGENT). In mirror-depends IDS scenario, we established two virtual networks in each cloud server: normal network and monitoring network. E-NICE-AGENT is connected to the monitoring network. Traffic on the normal network is mirrored to the monitoring network using Switched Port Analyzer (SPAN) approach. In the proxy-depends IDS solution, E-NICE-AGENT interfaces two VMs and the traffic goes through E-NICE-AGENT. Additionally, we have deployed the E-NICE-AGENT in Dom0 and it removes the traffic duplication function in mirroring and proxy-depends solutions. E-NICE-AGENT running in Dom0 is more efficient since it can sniff the traffic directly on the virtual bridge. However, in DomU, the traffic need to be duplicated on the VM's virtual interface (vif), causing overhead. When the IDS is running in Intrusion Prevention System (IPS) mode, it needs to intercept all the traffic and perform packet checking, which consumes more system resources as compared to IDS mode. To demonstrate performance

evaluations we used four metrics namely CPU utilization, network capacity, agent processing capacity, and communication delay. We performed the evaluation on cloud servers with Intel quad-core Xeon 2.4Ghz CPU and 32G memory. We used packet generator to mimic real traffic in the Cloud system. As shown in Figure 9, the traffic load, in form of packet sending speed, increases from 1 to 3000 packets per second. The performance at Dom0 consumes less CPU and the IPS mode consumes the maximum CPU resources. It can be observed that when the packet rate reaches to 3000 packets per second; the CPU utilization of IPS at DomU reaches its limitation, while the IDS mode at DomU only occupies about 68%.

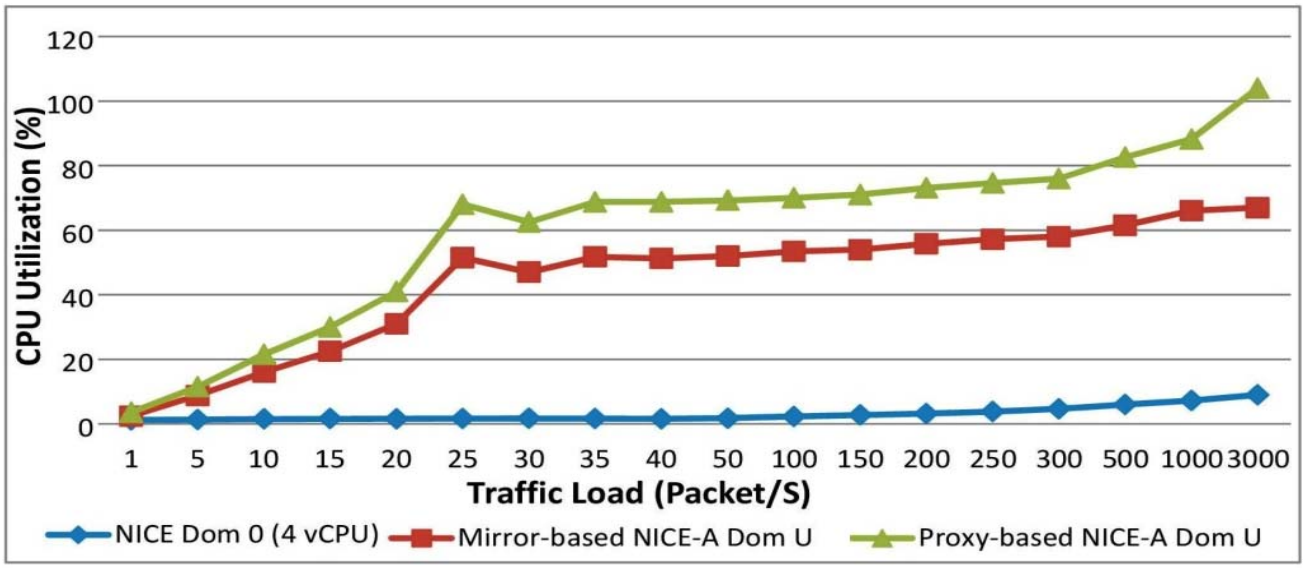


Figure 9: CPU utilization of E-NICE-AGENT

Figure 10, represents the performance of E-NICE-AGENT in terms of percentage of successfully analyzed packets, i.e., the number of the analyzed packets divided by the total number of packets received. The higher this value is, more packets this agent can handle. It can be observed from the result that IPS agent demonstrates 100% performance because every packet captured by the IPS is cached in the detection agent buffer. However, 100% success analyzing rate of IPS is at the cost of the analyzing delay. For other two types of agents, the detection agent does not store the captured packets and thus no delay is introduced. However, they all experience packet drop when traffic load is huge.

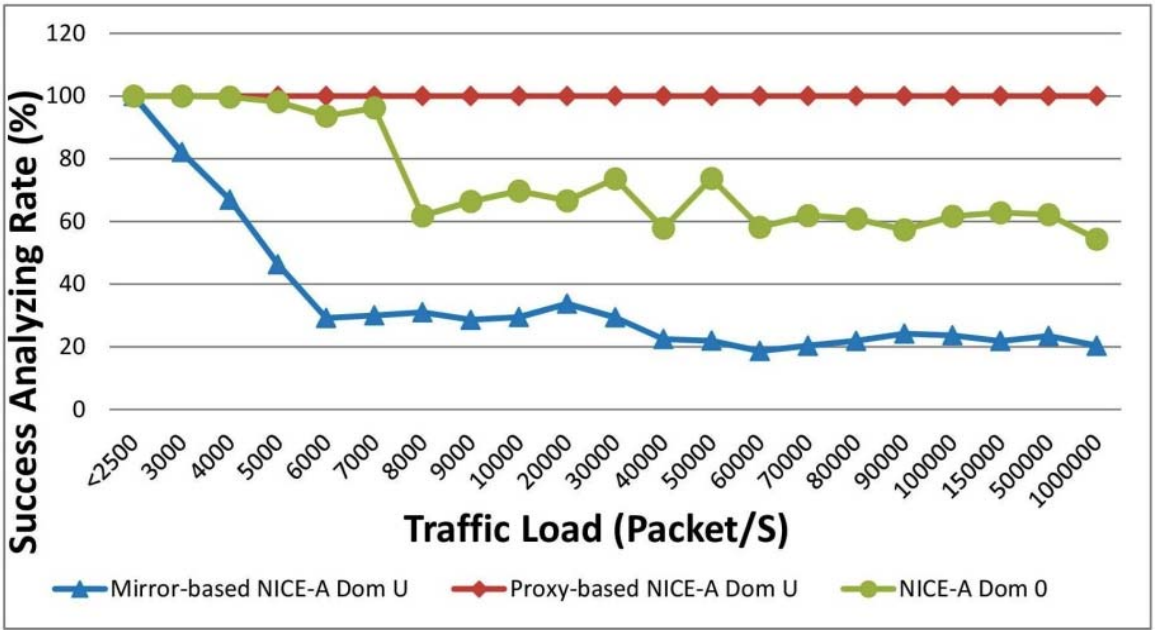


Figure 10: E-NICE-AGENT Success Analyzing Rate.

In Figure 11, the communication delay with the system under different E-NICE-AGENT is presented. We generated 100 consecutive normal packets with the speed of 1 packet per second to test the end-to-end delay of two VMs

compared by using E-NICE-AGENT running in mirroring and proxy modes in DomU and E-NICE running in Dom0. We record the minimal, average, and maximum communication delay in the comparative study. Results show that the delay of proxy-depends E-NICE-AGENT is the highest because every packet has to pass through it. Mirror-depends E-NICEA at DomU and E-NICE-AGENT at Dom0 do not have noticeable differences in the delay..

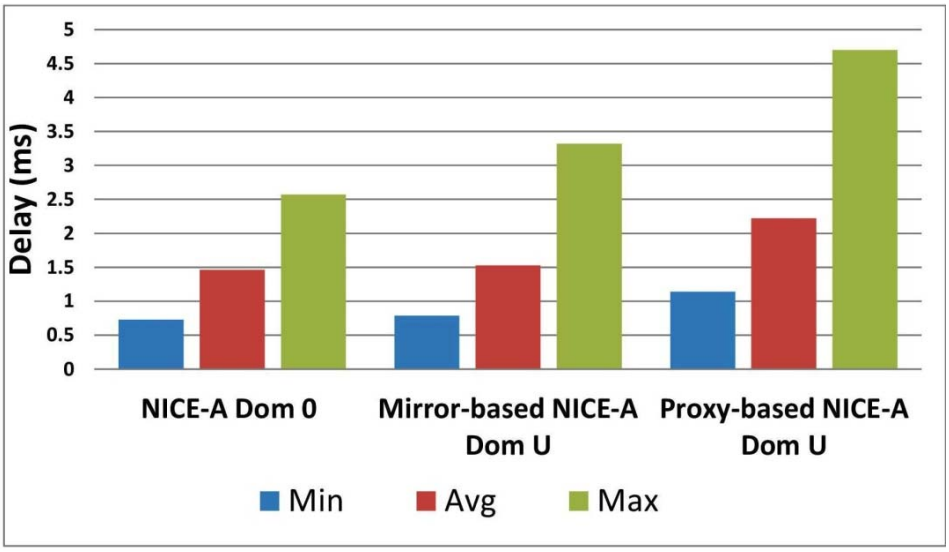


Figure 11: Network Communication Delay of E-NICE-AGENT.

From this test we expected to prove the proposed solution, thus achieving our goal"establish a dynamic defensive mechanism depends software defined networking approach that involves multiphase intrusion detections". The experiments prove that for a small-scale cloud system, our approach works well. The performance evaluation includes two parts. First, security performance evaluation. It shows that the our approach achieves the design security goals: to prevent vulnerable VMs from being compromised and to do so in less intrusive and cost effective manner. Second, CPU and throughput performance evaluation. It shows the limits of using the proposed solution in terms of networking throughputs depends on software switches and CPU usage when running detection engines on Dom 0 and Dom U. The performance results provide us a benchmark for the given hardware setup and shows how much traffic can be handled by using a single detection domain. To scale up to a data center level intrusion detection system, a decentralized approach must be devised, which is scheduled in our future research.

6. Conclusion and Future Scope

In this project we propose E-NICE, which is proposed to detect and mitigate collaborative attacks in the cloud virtual networking territory. E-NICE utilizes the attack graph model to conduct attack detection and prediction. The proposed solution investigates how to use the programmability of software controls based solutions to improve the detection correctness and defeat victim use phases of collaborative attacks. The system evaluation exhibits the attainability of E-NICE and shows that the proposed solution can significantly reduce the risk of the cloud system from being used and misused by intramural and external attackers. E-NICE only investigates the network IDS approach to counter zombie explorative attacks. In order to improve the detection correctness, host-based IDS solutions are needed to be incorporated and to cover the slit spectrum of IDS in the cloud system. This should be investigated in the future work. Additionally, as indicated in the project, we will investigate the scalability of the proposed E-NICE solution by investigating the denationalized network control and attack analysis model based on current study.

References

Good Teachers are worth more than thousand books, we have them in Our Department

References Made From:

[1] Coud Sercurity Alliance, "Top threats to cloud computing v1.0," https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, March 2010.
[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *ACM Commun.*, vol. 53, no. 4, pp. 50–58, Apr. 2010.
[3] B. Joshi, A. Vijayan, and B. Joshi, "Securing cloud computing territory hostile to DDoS attacks," *IEEE Int'l Conf. Computer Communication and Informatics (ICCCI '12)*, Jan. 2012.
[4] H. Takabi, J. B. Joshi, and G. Ahn, "Security and privacy challenges in cloud computing territorys," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, Dec. 2010.
[5] "Open vControl project," http://openvcontrol.org, May 2012.
[6] Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J. Barker, "Detecting spam zombies by observing outgoing messages," *IEEE Trans. Dependable and Secure Computing*, vol. 9, no. 2, pp. 198–210, Apr. 2012. IEEE TRANSACTIONS ON DEPEDABLE AND SECURE COMPUTING This article has been accepted

for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. 14

[7] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: detecting malware contamination through IDS-driven dialog correlation," *Proc. of 16th USENIX Security Symp. (SS '07)*, pp. 12:1–12:16, Aug. 2007.

[8] G. Gu, J. Zhang, and W. Lee, "BotSniffer: detecting botnet command and control channels in network traffic," *Proc. of 15th Ann. Network and Distributed Sytem Security Symp. (NDSS '08)*, Feb. 2008.

[9] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," *Proc. IEEE Symp. on Security and Privacy*, 2002, pp. 273–284.

[10] "NuSMV: A new symbolic model checker," http://afrodite.itc.it: 1024/nusmv. Aug. 2012.

[11] S. H. Ahmadinejad, S. Jalili, and M. Abadi, "A hybrid model for correlating aware of of known and unknown attack scenarios and updating attack graphs," *Computer Networks*, vol. 55, no. 9, pp. 2221–2240, Jun. 2011.

[12] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: a logicbased network security analyzer," *Proc. of 14th USENIX Security Symp.*, pp. 113–128. 2005.

[13] R. Sadoddin and A. Ghorbani, "Aware correlation survey: structure and techniques," *Proc. ACM Int'l Conf. on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (PST '06)*, pp. 37:1–37:10. 2006.

[14] L. Wang, A. Liu, and S. Jajodia, "Using attack graphs for correlating, hypothesizing, and predicting intrusion aware of," *Computer Communications*, vol. 29, no. 15, pp. 2917–2933, Sep. 2006.

[15] S. Roschke, F. Cheng, and C. Meinel, "A new aware correlation algorithm based on attack graph," *Computational Intelligence in Security for Information Systems*, LNCS, vol. 6694, pp. 58–67. Springer, 2011.