INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

A NOVEL INTRUSION DETECTION USING DECENTRALIZED ATTACK ANALYZER AND NETWORK CONTROLLER IN VIRTUAL NETWORK SYSTEM

K. Senthil Raja[1], G. Sudhakar[2], Dr. S. Nithyanandam[3]

[1]M.E CSE, Ranganathan Engineering College,
Coimbatore - 641109, India.
rksenthilraja27@gmail.com

[2]Assistant professor and Head, Department Of CSE,
Ranganathan Engineering College,
Coimbatore - 641109, India.
sudhakar.g7018@gmail.com

[3]Principal, PGP Engineering College,
Namakkal - 637207, India.
ssnithiyanandam@gmail.com

Abstract: *Cloud security is one of most important issues that have attracted a lot of research and development effort in past few years. Particularly, attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service (DDoS). DDoS attacks usually involve early stage actions such as multi step exploitation, low frequency vulnerability scanning, and compromising identified vulnerable virtual machines as zombies, and finally DDoS attacks through the compromised zombies. Within the cloud system, especially the Infrastructure-as-a-Service (IaaS) clouds, the detection of zombie exploration attacks is extremely difficult. This is because cloud users may install vulnerable applications on their virtual machines. To prevent vulnerable virtual machines from being compromised in the cloud, we propose a multi-phase distributed vulnerability detection, measurement, and countermeasure selection mechanism called NICE, which is built on attack graph based analytical models and reconfigurable virtual network-based countermeasures. The proposed framework leverages Decentralized Open Flow network programming APIs to build a monitor and control plane over distributed programmable virtual switches in order to significantly improve attack detection and mitigate attack consequences. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution.*

Keywords*: Performance of Systems, Computer Systems Organization, Communication/Networking and Information Technology, General, Network-Level Security and Protection.*

1. INTRODUCTION

A recent Cloud Security Alliance (CSA) survey shows that among all security issues, abuse and nefarious use of cloud computing is considered as the top security threat [1], in which attackers can exploit vulnerabilities in clouds and utilize cloud system resources to deploy attacks. In traditional data centers, where system administrators have full control over the host machines, Vulnerabilities can be detected and patched by the system administrator in a centralized manner. However, patching known security holes in cloud data centers, where cloud users usually have the privilege to control software installed on their managed VMs, may not work effectively and can violate the Service Level Agreement (SLA).

Furthermore, cloud users can install vulnerable software on their VMs, which essentially contributes to loopholes in Cloud security. The challenge is to establish an effective multiple VMs.

Vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users.

In [2], M. Armbrust et al. addressed that protecting "Business continuity and services availability" from service outages is one of the top concerns in cloud computing systems. In a cloud system where the infrastructure is shared by potentially millions of users, abuse and nefarious use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and use its resource to deploy attacks in more efficient ways [3]. Such attacks are more effective in the cloud environment since cloud users usually share computing

resources, e.g., being connected through the same switch, sharing with the same data storage and file systems, even with potential attackers [4]. The similar setup for VMs in the cloud, e.g., virtualization techniques, VM OS, installed vulnerable software, networking, etc., attracts attackers to compromise network intrusion detection solution.

In this paper, we propose Secure Intrusion Detection and Attack measure exquisite in Virtual Systems to establish a defense-in-depth intrusion detection framework. For better attack detection, NICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of NICE does not intend to improve any of the existing intrusion detection algorithms; indeed, NICE employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs.

In general, NICE includes two main phases: (1) deploy lightweight mirroring-based network intrusion detection agent (NICE-A) on each cloud server to capture and analyze cloud traffic. A NICE-A periodically scans the virtual system vulnerabilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability towards the collaborative attack goals, NICE will decide whether or not to put a VM in network inspection state. (2) Once a VM enters inspection state, Deep Packet Inspection (DPI) is applied, and/or virtual network reconfigurations can be deployed to the inspecting VM to make the potential attack behaviors prominent.

The rest of paper is organized as follows. Section II presents the related work. Section III describes system approach and implementation. System models are described in Section IV describes the approach to hardening the network in NICE. The proposed NICE is presented in Section V and Section VI evaluates NICE in terms of network performance and security. Finally, Section VII describes future work and concludes this paper.

2. RELATED WORKS

The contributions of NICE are presented as follows:

- 1. We devise NICE, a new multi-phase distributed network intrusion detection and prevention framework in a virtual networking environment that captures and inspects suspicious cloud traffic without interrupting users' applications and cloud services.
- 2. NICE incorporates a software switching solution to quarantine and inspect suspicious VMs for further investigation and protection. Through programmable network approaches, NICE can improve the attack detection probability and improve the resiliency to VM exploitation attack without interrupting existing normal cloud services.
- 3. NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures.
- 4. NICE optimizes the implementation on cloud servers to minimize resource consumption. Our study shows that NICE consumes less computational overhead compared to proxy-based
-

The area of detecting malicious behavior has been well explored. The work by Duan et al. [6] focuses on the detection of compromised machines that have been recruited to serve as spam zombies. Their approach, SPOT, is based on sequentially scanning outgoing messages while employing a statistical method Sequential Probability Ratio Test (SPRT), to quickly determine whether or not a host has been compromised. BotHunter [7] detects compromised machines based on the fact that a thorough malware infection process has a number of well defined stages that allow correlating the intrusion alarms triggered by inbound traffic with resulting outgoing communication patterns. BotSniffer [8] exploits uniform spatial-temporal behavior characteristics of compromised machines to detect zombies by grouping flows according to server connections and searching for similar behavior in the flow.

An attack graph is able to represent a series of exploits, called atomic attacks, that lead to an undesirable state, for example a state where an attacker has obtained administrative access to a machine. There are many automation tools to construct attack graph. O. Sheyner et al. [9] proposed a technique based on a modified symbolic model checking NuSMV [10] and Binary Decision Diagrams (BDDs) to construct attack graph. Their model can generate all possible attack paths, however, the scalability is a big issue for this solution. P. Ammann et al. [11] introduced the assumption of monotonicity, which states that the precondition of a given exploit is never invalidated by the successful application of another exploit. In other words, attackers never need to backtrack. With this assumption, they can obtain a concise, scalable graph representation for encoding attack tree. X. Ou et al. proposed an attack graph tool called MulVAL [12], which adopts a logic programming approach and uses Data log language to model and analyze network system. Intrusion Detection System (IDS) and firewall are widely used to monitor and detect suspicious events in the network. However, the false alarms and the large volume of raw alerts from IDS are two major problems for any IDS implementations. In order to identify the source or target of the intrusion in the network, especially to detect multi-step attack, the alert correction is a must-have tool. The primary goal of alert correlation is to provide system support for a global and condensed view of network attacks by analyzing raw alerts [13]. Many attack graph based alert correlation techniques have been proposed recently. L. Wang et al. [14] devised an in-memory structure, called queue graph (QG), to trace alerts matching each exploit in the attack graph. However, the implicit correlations in this design make it difficult to use the correlated alerts in the graph for analysis of similar attack scenarios. Roschke et al. [15] proposed a modified attack-graph-based correlation algorithm to create explicit correlations only by matching

alerts to specific exploitation nodes in the attack graph with multiple mapping functions, and devised an alert dependencies graph (DG) to group related alerts with multiple correlation criteria. Several solutions have been proposed to select optimal countermeasures based on the likelihood of the attack path and cost benefit analysis. A. Roy et al. [16] proposed an attack countermeasure tree (ACT) to consider attacks and countermeasures together in an attack tree structure. [17] Proposed a Bayesian attack graph (BAG) to address dynamic security risk management problem and applied a genetic algorithm solve countermeasure optimization problem.

3. NICE MODELS

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

3.1 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-based attack detection and reconfiguration solutions to improve the resiliency to zombie explorations.

Our work does not involve host-based IDS and does not address how to handle encrypted traffic for attack detections. Our proposed solution can be deployed in an Infrastructure- s-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.

3.2. 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 [18]. In an attack graph, each node represents either precondition or consequence of an exploit. Attack graph is helpful in identifying potential threats, possible attacks and known vulnerabilities in a cloud system.

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 2 Epre _ ND _ NC represents that ND must be satisfied to achieve NC. An edge e 2 Epost _ NC _ ND means that the consequence shown by ND can be obtained if NC is satisfied.

Node VC E NC is defined as a three tuple (Hosts; vul; alert) representing a set of IP addresses, vulnerability information such as CVE [19], 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

- A is a set of aggregated alerts. An alert a 2 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.
- Each alert a maps to a pair of vertices (vc; vd) in SAG using map (a) function, E is a set of directed edges representing correlation between two alerts
- P is set of paths in ACG.

Algorithm 1 Alert correlation

Require: alert ac, SAG, ACG
if (ac is a new alert) then create node ac in ACG
n1←vc 2 map (ac) for
all n2 parent(n1) do
create edge (n2,alert,ac) for all Si containing a do
if a is the last element in Si then append ac to Si
else
create path Si+1={subset (Si a)ac } end if
end for
add ac to n1alert
end for
end if
return S

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

- Stable: there does not exist any known vulnerability on the VM.
- Vulnerable: presence of one or more vulnerabilities on a VM, which remains unexploited.
- Exploited: at least one vulnerability has been exploited and the VM is compromised.
- Zombie: VM is under control of attacker.

4. NICE SYSTEM DESIGN

In this section, we first present the system design overview of NICE and then detailed descriptions of its components.

4.1 System design overview

The proposed NICE framework is illustrated in Figure 1.It shows the NICE framework within one cloud server cluster. Major components in this framework are distributed and light-weighted NICE-A on each physical

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

cloud server, a decentralized network controller, a VM profiling server, and an attack analyzer. The latter three components are located in a centralized control center connected to software switches on each cloud server (i.e., virtual switches built on one or multiple Linux software bridges).

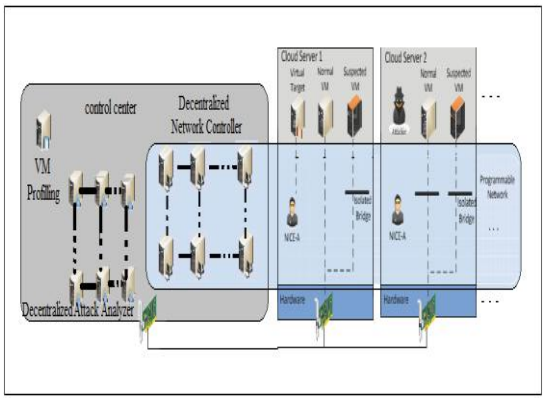

Figure1: NICE within one cloud server cluster.

4.2. System Components

In this section we explain each component of NICE.

4.2.1. NICE-A

The NICE-A is a Network-based Intrusion Detection System (NIDS) agent installed in either dom0 or domU in each cloud server. It scans the traffic going through Linux bridges that control all the traffic among VMs and in/out from the physical cloud servers. NICEA is a software agent implemented in each cloud server connected to the control center through a dedicated and isolated secure channel, which is separated from the normal data packets using OpenFlow tunneling or VLAN approaches. The decentralized network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer.

4.2.2. VM Profiling

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, etc. VM profiles are maintained in a database and contain comprehensive information about vulnerabilities, alert and traffic. The data comes from:

Attack graph generator: while generating the attack graph, every detected vulnerability is added to its corresponding VM entry in the database.

NICE-A: the alert involving the VM will be recorded in the VM profile database. Decentralized Network controller: the traffic patterns involving the VM are based on 5 tuples (source MAC address, destination MAC address, source IP address, destination IP address, protocol). We can have traffic pattern where packets emanate from a single IP and are delivered to multiple destination IP addresses, and vice-versa.

4.2.3. Attack Analyzer

The major functions of NICE system are performed by attack analyzer, which includes procedures such as attack graph construction and update, alert correlation and countermeasure selection.

The process of constructing and utilizing the Scenario Attack Graph (SAG) consists of three phases: information gathering, attack graph construction, and potential exploit path analysis. With this information, attack paths can be modelled using SAG.

In summary, NICE attack graph is constructed based on the following information: Cloud system information is collected from the node controller and VM's Virtual Interface (VIF) information. Virtual network topology and configuration information is collected from the decentralized network controller, every VM's IP address, MAC address, port information, and traffic flow information. Vulnerability information is generated by both on demand vulnerability scanning.

4.2.4. Network Controller

The Decentralized network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration feature based on Open-Flow protocol [20]. In NICE, within each cloud server there is a software switch, for example, Open vSwitch (OVS) [5], which is used as the edge switch for VMs to handle traffic in & out from VMs. The decentralized network controller is responsible for collecting network information of current OpenFlow network and provides input to the decentralized attack analyzer to construct attack graphs.

5. MITIGATION & COUNTERMEASURE

In this section, we present the methods for selecting the countermeasures for a given attack scenario. The countermeasure serves the purpose of 1) protecting the target VMs from being compromised; and 2) making attack behavior stand prominent so that the attackers' actions can be identified.

5.1 Mitigation Strategies

Based on the security metrics defined in the previous subsection, NICE is able to construct the mitigation strategies in response to detected alerts. First, we define the term countermeasure pool as follows:

Definition 4 (Countermeasure Pool).

A countermeasure pool CM = (cm1; cm2;: : ; cmn) is a set of countermeasures. Where

1. Cost is the unit that describes the expenses required to apply the countermeasure in terms of resources and operational complexity, and it is defined in a range from 1 to 5, and higher metric means higher cost;

2. intrusiveness is the negative effect that a countermeasure brings to the Service Level Agreement (SLA) and its value ranges from the least intrusive (1) to the most intrusive (5), and the value of intrusiveness is 0 if the countermeasure has no impacts on the SLA;

3. Condition is the requirement for the corresponding

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

countermeasure;

4. Effectiveness is the percentage of probability changes of the node, for which this countermeasure is applied.

In general, there are many countermeasures that can be applied to the cloud virtual networking system depending on available countermeasure techniques that can be applied. Without losing the generality, several common virtual-networking-based countermeasures are listed in Table 1.

Table 1: Possible Countermeasure Types

No	Countermeasure	Intrusiveness	Cost
1	Traffic redirection	3	3
2	Traffic isolation	4	2
3	Deep packet inspection	3	3
4	Creating filtering rules	1	2
5	MAC address changes	2	1
6	IP address changes	2	1
7	Block port	4	1
8	Software patch	5	4
9	Quarantine	5	2
10	Network reconfiguration	0	5
11	Network topology changes	0	5

5.2. Countermeasure selection

Algorithm 2 presents how to select the optimal countermeasure for a given attack scenario. Input to the algorithm is an alert, attack graph G, and a pool of countermeasures CM. The algorithm starts by selecting the node vAlert that corresponds to the alert generated by a NICE-A. The countermeasure which when applied on a node gives the least value of ROI, is regarded as the optimal countermeasure. Finally, SAG and ACG are also updated before terminating the algorithm.

Algorithm 2 countermeasure selection

Require: Alert; G (E, V); CM
Let vAlert=Source node of the alert
if Distance to target (vAlert) > threshold then update ACG
Return end if
Let T= Descendant (vAlert) U vAlert set pr (vAlert) =1
Calculate Risk Prob (T) Let benefit [jTj; jCMj] = Ø
for each t E T
do
for each cm E CM do if cm: condition (t) then
Pr (t) = Pr (t) (1-cm: effectiveness)
Calculate Risk Prob (Descendant (t)) benefit [t; cm]
Pr(target node): (7)
end if
end for end for
Let ROI [jTj; jCMj] = Ø for each t E T do
for each cm E CM do ROI [t; cm]
end for end for
Update SAG and Update ACG
return Select Optimal CM (ROI)

6. PERFORMANCE EVALUATION

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

6.1 Security Performance Analysis

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

6.1.1 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 2. 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. Table 2 shows the vulnerabilities present in this network.

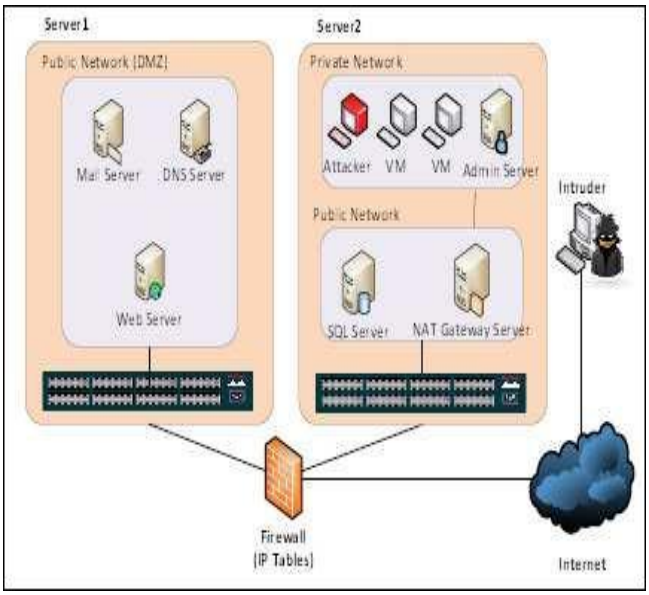

Figure 2: Virtual Network Technology for security Evaluation

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

Table 2: Vulnerabilities in the virtual networked system

Host	Vulnerability	Node	CVE	Base Scope
VM Group	LICQ buffer overflow	10	CVE2001-0439	0.75
	MS Video ActiveX stack buffer overflow	5	CVE	0.93
	GNU C Library loader	22	CVE	0.69
Admin Server	MS SMV service stack buffer overflow	2	CVE	0.93
Gateway server	Open SSL uses predictable random variable	15	CVE	0.78
	Heap corruption in open SSH	4	CVE	1
	Improper cookies handler in OpenSSH	9	CVE	0.75
Mail Server	Remote code execution in SMTP	21	CVE	1
	Squid port in scan	19	CVE	0.75
Web Server	WebDAV Vulnerability in IIS	133	CVE	0.76

6.1.2 Attack Graph and Alert Correlation

The attack graph can be generated by utilizing network topology and the vulnerability information, and it is shown in Figure 3. 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 vulnerability information. This information is provided to the attack graph generator as the input.

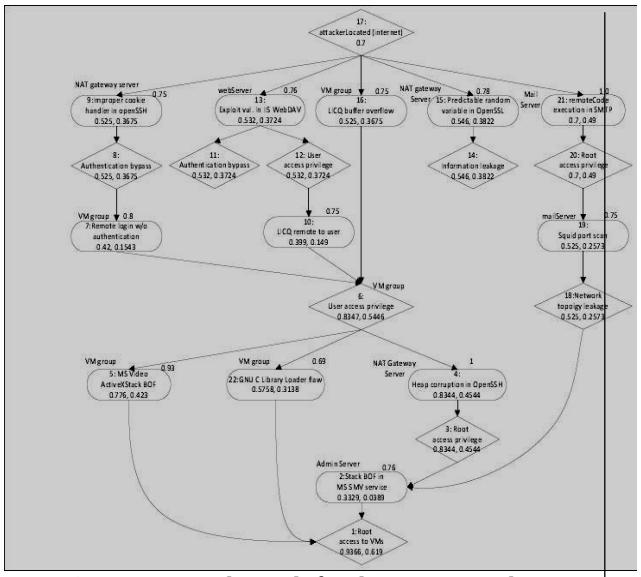

Figure 3: Attack graph for the test network

Definition 5 (VM Security Index). VSI for a virtual machine k is defined as V SIk = (Vk + Ek)=2, where

1. Vk is vulnerability score for VM k. The score is the exponential average of base score from each vulnerability in the VM or a maximum 10, i.e., Vk = minf10; lnPeBaseScore (v) g.

2. Ek is exploitability score for VM k. It is the exponential average of exploitability score for all vulnerabilities or a maximum 10 multiplied by the ratio of network services on the VM, i.e., Basically, vulnerability score considers the base scores of all the vulnerabilities on a VM. The base score depicts how easy it is for an attacker to exploit the vulnerability and how much damage it may incur. Figure 4 shows benefit evaluation for presented countermeasure. The exponential addition of base scores allows the vulnerability score to incline towards higher base score values and increases in logarithm-scale based on the number of vulnerabilities.

Apart from calculating the benefit measurements, we also present the evaluation based on Return of Investment (ROI) using (8) and represent a comprehensive evaluation considering benefit, cost and intrusiveness of countermeasure. Figure 5 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.

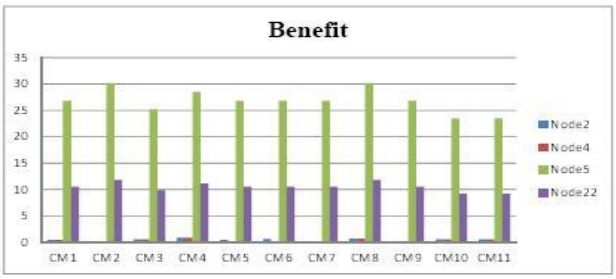

Figure 4: Benefit evaluation chart

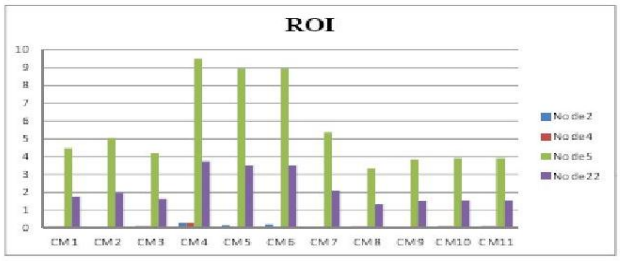

Figure 5: ROI evaluation chart

6.2. NICE System Performance

We evaluate system performance to provide guidance on how much traffic 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. Figure 6 shows CPU utilization of NICE-A for presented countermeasure.
Traffic planning is needed to run NICE, which is beyond the scope of this paper. Due to the space limitation, we will investigate the research involving multiple cloud clusters in the future.

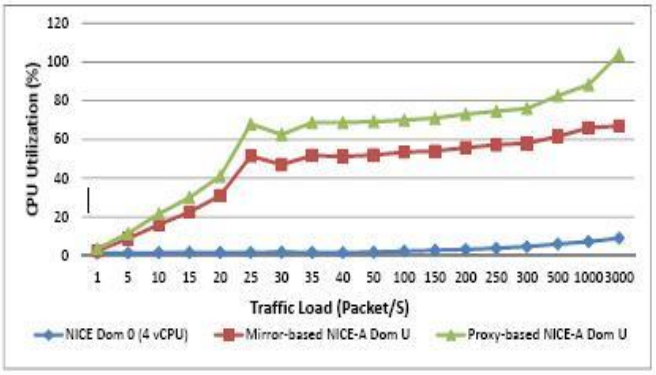

Figure 6: CPU utilization of NICE-A

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS.....

7. CONCLUSION

In this paper, we presented NICE, which is proposed to detect and mitigate collaborative attacks in the cloud virtual networking environment. NICE utilizes the attack graph model to conduct attack detection and prediction. The proposed solution investigates how to use the programmability of software switches based solutions to improve the detection accuracy and defeat victim exploitation phases of collaborative attacks. The system performance evaluation demonstrates the feasibility of NICE and shows that the proposed solution can significantly reduce the risk of the cloud system from being exploited and abused by internal and external attackers. NICE only investigates the network IDS approach to counter zombie explorative attacks. In order to improve the detection accuracy, host-based IDS solutions are needed to be incorporated and to cover the whole spectrum of IDS in the cloud system.

REFERENCES

[1] Cloud Computing Alliance," Top threats to cloud computing v1.0," https://cloudsecurityalliance.org /top threats/csathreats. v 1.0.pdf, Mar.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 Comm., vol. 53, no. 4, pp. 50-58, Apr. 2010.

[3] B. Joshi, A. Vijayan, and B. Joshi, "Securing Cloud Computing Environment Against DDoS Attacks," Proc. IEEE Int'l Conf. Computer Comm. and Informatics (ICCCI '12), Jan. 2012.

[4] H. Takabi, J.B. Joshi, and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31, Dec. 2010.

[5] "Open vSwitch Project," http://openvswitch.org, May 2012.

[6] Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J. Barker, "Detecting Spam Zombies by Monitoring Outgoing Messages," IEEE Trans. Dependable and Secure Computing, vol. 9, no. 2, pp. 198-210, Apr. 2012.

[7] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection through IDS-driven Dialog Correlation," Proc. 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. 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. Security and Privacy, pp. 273-284, 2002,

[10] "NuSMV: A New Symbolic Model Checker," http://afrodite.itc. it:1024/nusmv. Aug. 2012.

[11] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph based network vulnerability analysis," Proc. 9th ACM Conf. Computer and Comm. Security (CCS '02), pp. 217-224, 2002.

[12] X. Ou, S. Govindavajhala, and A.W. Appel, "MulVAL: A Logic-Based Network Security Analyzer," Proc. 14th USENIX Security Symp., pp. 113-128, 2005.

[13] R. Sadoddin and A. Ghorbani, "Alert Correlation Survey: Framework and Techniques," Proc. ACM Int'l Conf. 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 Alerts," Computer Comm., vol. 29, no. 15, pp. 2917-2933, Sept. 2006.

[15] S. Roschke, F. Cheng, and C. Meinel, "A New Alert Correlation Algorithm Based on Attack Graph," Proc. Fourth Int'l Conf. Computational Intelligence in Security for Information Systems, pp. 58-67, 2011.

[16] A. Roy, D.S. Kim, and K. Trivedi, "Scalable Optimal Countermeasure Selection Using Implicit Enumeration on Attack Countermeasure Trees," Proc. IEEE Int'l Conf. Dependable Systems Networks (DSN '12), June 2012.

[17] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," IEEE Trans. Dependable and Secure Computing, vol. 9, no. 1, pp. 61-74, Feb. 2012.

[18] Open Networking Fundation, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, Apr. 2012.

[19] "Openflow," http://www.openflow.org/wp/learn More/,2012.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," SIGCOMM Computer Comm. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.