SECURING THE NETWORK IN CLOUD ENVIRONMENT

P. Sakthi Priyanka, C. Radha
AP/MCA, Muthayammal Engineering College, Rasipuram
Tamilnadu, India

***Abstract*— A multiphase 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 Open Flow network programming APIs to build a monitor and control plane over distributed programmable virtual switches to significantly improve attack detection and mitigate attack consequence. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution. 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. 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.**

***Index Terms*- Counter measures, IDS, Network Security, Virtual Machine, Vulnerability**

I. INTRODUCTION

A network intrusion that is composed of multiple attacks preparing for each other can infiltrate a well-guarded network. Defending against such multi-step intrusions is important but challenging. It is usually impossible to respond to such intrusions based on isolated alerts corresponding to individual attack steps. The reason lies in the well-known impreciseness of Intrusion Detection Systems (IDSs). That is, alerts reported by IDSs are usually filled with false alerts that correspond to either normal traffic or failed attack attempts. To more effectively defend against multi-step intrusions, isolated alerts need to be correlated into attack scenarios. The propose Network Intrusion detection and Countermeasures Election in virtual network systems (NICE) 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 a 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 toward 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. NICE significantly advances the current network IDS/IPS solutions by employing programmable virtual networking approach that allows the system to construct a dynamic reconfigurable IDS system. By using software switching techniques, NICE constructs a mirroring-based traffic capturing framework to minimize the interference on users' traffic compared to traditional bump-in-the-wire (i.e.proxy-based) IDS/IPS. The programmable virtual networking architecture of NICE enables the cloud to establish inspection and quarantine modes for suspicious VMs according to their current vulnerability state in the current SAG. Based on the collective behavior of VMs in the SAG,NICE can decide appropriate actions, for example, DPI or

traffic filtering, on the suspicious VMs. Using this approach, NICE does not need to block traffic flows of a suspicious VM in its early attack stage. The contributions of NICE are presented as follows:

- We devise NICE, a new multiphase 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.
- 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.
- NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures. NICE optimizes the implementation on cloud servers to minimize resource consumption. NICE consumes less computational overhead compared to proxy-based network intrusion detection solution.

II. RELATEDWORK

Network vulnerability analysis enumerates potential attack sequences between fixed initial conditions and attack goals .To avoid potential combinatorial explosion in the number of attack sequences, we adopt a notation of attack graphs similar to that However, we do not assume fixed initial or goal conditions but let alerts to indicate the actual start and end of an intrusion. The Bayesian network-based approach. We adopt the vulnerability-centric approach to alert correlation because it can effectively filter out bogus alerts irrelevant to the network. Attack scenarios broken by missed attacks are reassembled by clustering alerts with similar attributes and those caused by incomplete knowledge are pieced together through statistical analyses. Instead of repairing a broken scenario afterwards, our method can tolerate and hypothesize missed attacks at the same time of correlation. Real-Time detection of isolated alerts some products claim to support real-time analyses of alerts, such as the Tivoli Risk Manager. Designed for a different purpose, this paper extends the preliminary results reported in as follows. First, we show that a result graph produced by the basic QG approach given in still contains redundant information, such as transitive edges. The transitive edges can be removed to simplify a result graph without losing any information. Moreover, some alerts in a result graph are indistinguishable with respect to their relationship with others. Those alerts can thus be aggregated to make the result graph more perceptible. Second, we modify the basic QG approach such that it produces result graphs with all transitive edges removed and indistinguishable alerts aggregated. Unlike existing approaches that take extra efforts to compress result graphs, the modified version of four QG approach not only produces more compact result graphs, but also is more efficient in most cases. Several solutions have been proposed to select optimal countermeasures based on the likelihood of the attack path and cost benefit analysis. Proposed an attack countermeasure tree (ACT) to consider attacks and countermeasures together in an attack tree structure. They devised several objective functions based on greedy and branch and bound techniques to minimize the number of countermeasure, reduce investment cost, and maximize the benefit from implementing a certain countermeasure set. In their design, each countermeasure optimization problem could be solved with and without probability assignments to the model. However, their solution focuses on a static attack scenario and predefined countermeasure for each attack. Problem and applied a genetic algorithm to solve countermeasure optimization problem. Our solution utilizes a new network control approach called SDN where networking functions can be programmed through software switch and Open Flow protocol, plays a major role in this research. Flow based switches, such as OVS and Open Flow Switch (OFS) support fine-grained and flow-level control for packet switching. With the help of the central controller, all Open Flow-based switches can be monitored and configured. We take advantage of flow-based switching and network controller to apply the selected network countermeasures in our solution.

III. NICE MODELS

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.

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-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 vulnerabilities to their controlled VMs. Physical security of cloud server is out of scope of this paper. We assume that the hypervisor is secure and free of any vulnerabilities. The issue of a malicious tenant breaking out of DomU and gaining access to physical server have been studied in recent work and are out of scope of this paper.

Attack Graph Model

An attack graph is a modeling tool to illustrate all possible multistage, 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 because normal protocol interactions can also be used for attacks. Attack graph is helpful in identifying potential threats, possible attacks, and known vulnerabilities in a cloud system. Since the attack graph provides details of all known vulnerabilities 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 Multiple VAL and define as Scenario Attack Graph (SAG).

Definition 1 (SAG). An SAG is a tuple
SAG =(V,E),Where

1. $V = N_C \cup N_D \cup N_R$ 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. $E = E_{pre} \cup E_{post}$ denotes the set of directed edges
3. $e \in E_{pre} \subseteq N_D \times N_C$ represents that ND must be satisfied to achieve NC.

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 (ACG). An ACG is a three tuple ACG =(A,E,P),where A is a set of aggregated alerts. An alert a ϵ A is a data structure (src,dst,cls,ts) representing source IPaddress, destination IP address, type of the alert, and time stamp of the alert respectively.

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

1. Stable. There does not exist any known vulnerability on the VM.
2. Vulnerable. Presence of one or more vulnerabilities on a VM, which remains unexploited.
3. Exploited. At least one vulnerability has been exploited and the VM is compromised.
4. Zombie. VM is under control of attacker

IV. NICE SYSTEM DESIGN

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

System Design Overview

The proposed NICE framework is illustrated in Fig. 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 cloud server, a 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).NICE-A 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 Open Flow tunneling or VLAN approaches. The network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer. In the following description, our terminologies are based on the XEN virtualization technology. NICE-A is a network intrusion detection engine that can be installed in either Dom0 or DomU of a XEN cloud server to capture and filter malicious traffic. Intrusion detection alerts are sent to control center when suspicious or anomalous traffic is detected. After receiving an alert, attack analyzer evaluates the severity of the alert based on the attack graph, decides what countermeasure strategies to take, and then initiates it through the network controller. An attack graph is established according to the vulnerability information derived from both offline and real-time vulnerability scans. Offline scanning can be done by running penetration tests and online real-time vulnerability scanning can be triggered by the network controller (e.g., when new ports are opened and identified by OFSs) or when new alerts are generated by the NICE-A. Once new vulnerabilities are discovered or countermeasures are deployed, the attack graph will be reconstructed Countermeasures are initiated by the attack

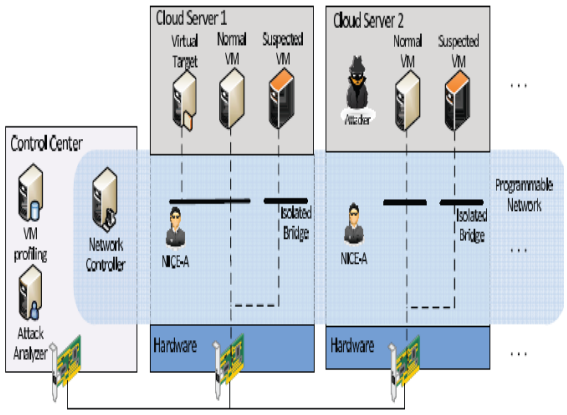

Fig. 1. NICE architecture within one cloud server cluster.

System Components

In this section, we explain each component of NICE.

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. In our experiment, Snort is used to implement NICE-A in Dom0. It will sniff a mirroring port on each virtual bridge in the Open v Switch (OVS). Each bridge forms an isolated subnet in the virtual network and connects to all related VMs. The traffic generated from the VMs on the mirrored software bridge will be mirrored to a specific port on a specific bridge using SPAN, RSPAN, or ERSPAN methods. The NICE-A sniffing rules have been custom defined to suite our needs. Dom0 in the Xen environment is a privilege domain, that includes a virtual switch for traffic switching among VMs and network drivers for physical network interface of the cloud server. It is more efficient to scan the traffic in Dom0 because all traffic in the cloud server needs go through it; however, our design is independent to the installed VM. In the performance evaluation section, we will demonstrate the tradeoffs of installing NICE-A in Dom0 and DomU.

VM Profiling

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, and so on. One major factor that counts toward a VM profile is its connectivity with other VMs. Any VM that is connected to more number of machines is more crucial than the one connected to fewer VMs because the effect of

compromise of a highly connected VM can cause more damage. Also required is the knowledge of services running on a VM so as to verify the authenticity of alerts pertaining to that VM. These factors combined will form the VM profile.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.

Network Controller: The traffic patterns involving the VM are based on five 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.

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 SAG consists of three phases: Information gathering, attack graph construction, and potential exploit path analysis. Each path from an initial node to a goal node represents a successful attack:

Cloud system information: it is collected from the node controller (i.e., Dom0 in Xen Server). The information includes the number of VMs in the cloud server, running services on each VM, and VM's Virtual Interface (VIF) information.

Virtual network topology and configuration information: it is collected from the network controller, which includes virtual network topology, host connectivity, VM connectivity, every VM's IP address, MAC address, port information, and traffic flow information. Vulnerability Information is generated by both access and vulnerability scanning (i.e., initiated by the network controller and NICE-A) and regular penetration testing using the well-known vulnerability databases, such as Open Source Vulnerability Database (OSVDB), Common Vulnerabilities and Exposures List (CVE), and NIST National Vulnerability Database (NVD).The attack analyzer also handles alert correlation and analysis operations. This component has two major functions: 1) constructs ACG, and 2) provides threat information and appropriate countermeasures to network controller for virtual network reconfiguration. Fig. 2 shows the workflow in the attack analyzer component. After receiving an alert from NICE-A, alert analyzer matches the alert in the ACG. If the alert already exists in the graph and it is a known attack (i.e., matching the attack signature), the attack analyzer performs countermeasure selection procedure according to the algorithm and then notifies network controller immediately to deploy countermeasure or mitigation actions. If the alert is new, attack analyzer will perform alert correlation and analysis according to Algorithm 1, and updates ACG and SAG. This algorithm correlates each new alert to a matching alert correlation set (i.e., in the same attack scenario). A selected countermeasure is applied by the network controller based on the severity of evaluation results.

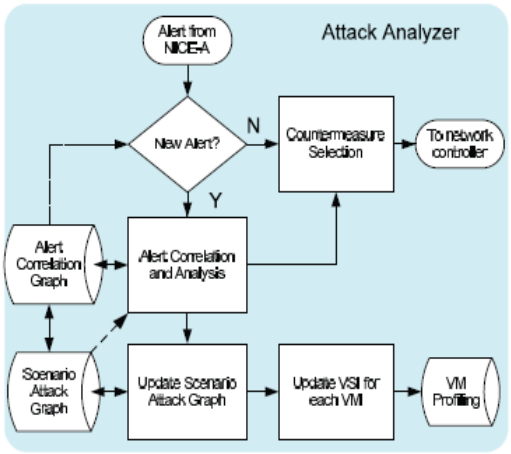

Fig. 2. Workflow of attack analyzer.

Network Controller

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration feature based on Open Flow protocol. In NICE, within each cloud server there is a software switch, In NICE, we integrated the control functions for both OVS and OFS into the network controller that allows the cloud system to set security/filtering rules in an integrated and comprehensive manner. The network controller is responsible for collecting network information of current Open Flow network and provides input to the

attack analyzer to construct attack graphs. Through the cloud internal discovery modules that use DNS, DHCP, LLDP, connectivity information from OVS and OFS. This information includes current data paths on each switch and detailed flow information associated with these paths, such as TCP/IP and MAC header. The network flow and topology change information will be automatically sent to the controller and then delivered to attack analyzer to reconstruct attack graphs.

V. NICE SECURITY MEASUREMENT, AND COUNTERMEASURES

1 Security Measurement Metrics
2 Mitigation Strategies
3 Countermeasure Selection

VI. PERFORMANCE EVALUATION

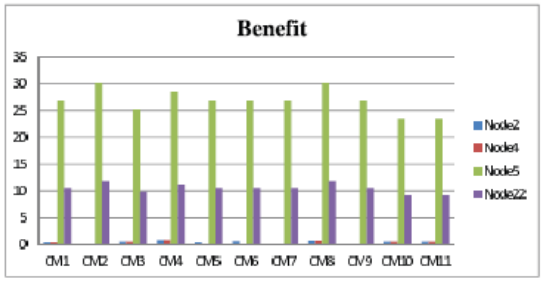

Fig. 3 Benefit evaluation chart

NICE System Performance

To demonstrate the feasibility of our solution, comparative studies were conducted on several virtualization approaches. We evaluated NICE based on Dom0 and DomU implementations with mirroring-based and proxy based attack detection agents (i.e., NICE-A). In mirror based IDS scenario, we established two virtual networks in each cloud server: Normal network and monitoring network .NICE-A is connected to the monitoring network. Traffic on the normal network is mirrored to the monitoring network using Switched Port Analyzer (SPAN)

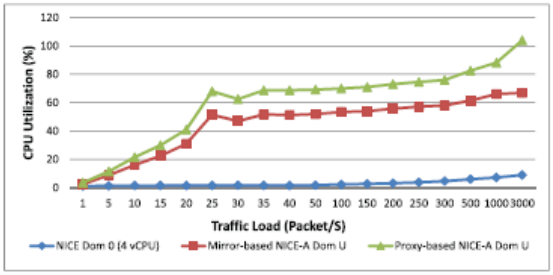

Fig. 4 CPU utilization of NICE-A

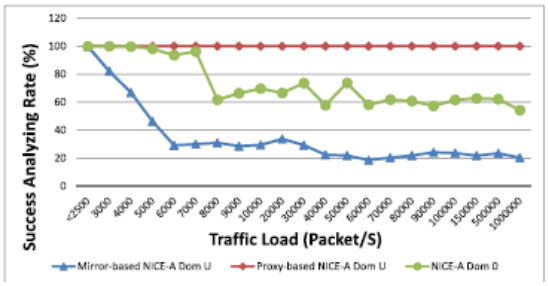

Fig. 5 NICE-A success analysis rate

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 IDS, a decentralized approach must be devised

VII. CONCLUSION AND FUTURE WORK

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. 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. This should be investigated in the future work. Additionally, as indicated in the paper, we will investigate the scalability of the proposed NICE solution by investigating the decentralized network

control and attack analysis model based on current study.

REFERENCES

[1] Cloud Security Alliance, "Top Threats to CloudComputingv1.0,"ttps://cloudsecurityalliance.org/topthreats/csathreats. v1.0.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 v Switch 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,"Bot Hunter: 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, "Bot Sniffer: Detecting Botnet Command and Control Channels in Network Traffic," Proc. 15thAnn. Network and Distributed System 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.