ISSN: 2321-7782 (Online)
Volume 3, Issue 2, February 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Reduce the Vulnerability Attack by using Virtual Network

Gorripati Chengamma[1]
M.Tech Student
Department of Computer Science and Engineering
Shree Institute of Technical education, JNTUA
Anantapur, Gopalapuram, Chittor district
Andhra Pradesh, India

Nasana Yedukondalu[2]
Assistant Professor
Department of Computer Science and Engineering
Shree Institute of Technical education, JNTUA
Anantapur, Gopalapuram, Chittor district
Andhra Pradesh, India

Abstract: ***Security is becoming an important issue in cloud environment. Attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large scale Distributed Denial of Service (DDoS). To prevent vulnerable virtual machine from being compromised in cloud a multiphase distributed vulnerability detection, measurement and countermeasure selection mechanism called NICE have been proposed. Now days, these systems are used by number of organizations to detect the weaknesses threats and preventing them. For this purpose, these systems became an important part of the security in nearly every organization. In a Cloud computing environment attackers.***

I. INTRODUCTION

Users migrating to the cloud consider security as the most important factor. 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, 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 vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users 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 infra structure 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 multiple VMs.

II. OBJECTIVE

The main aim of is to prevent the vulnerable virtual machines from being compromised in the cloud server using multi-phase distributed Vulnerability detection, measurement, and countermeasure selection mechanism called NICE.

III. NICE System Design

System Design

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 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. 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 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 Open Flow switches) 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 analyzer based on the evaluation results from the cost-benefit analysis of the effectiveness of countermeasures. Then, the network controller initiates Countermeasure actions by reconfiguring virtual or physical overflow switches.

System Components

1. Nice-A:

N I C E -A is a network based Intrusion D e t e c t i o n System (NIDS) agent installed in each cloud server. It scans the traffic going through the bridges that control all the traffic among VMs and in/out from the physical cloud servers. It will sniff a mirroring port on each virtual bridge in the Open switch. 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. It's more efficient to scan the traffic in cloud server since all traffic in the cloud server needs go through it; however our design is independent to the installed VM. The false alarm rate could be reduced through our architecture design.

2. VM Profiling:

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, etc. One major factor that counts towards a VM profile isits connectivity with other VMs. Also required is the knowledge of services running on a VM so as to verify the authenticity of alerts pertaining to that VM. An attacker can use port scanning program to perform an intense examination of the network to look for open ports on any VM. So information about any open ports on a VM and the history of opened ports plays a significant role in determining how vulnerable the VM is. All these factors combined will form the VM profile. VM profiles are maintained in a database and contain comprehensive information about vulnerabilities, alert and traffic.

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 modeled using SAG. The Attack Analyzer also handles alert correlation and analysis operations. This component has two major functions:

(1) Constructs Alert Correlation Graph (*ACG*),

(2) Provides threat information and appropriate countermeasures to network controller for virtual network reconfiguration.

NICE attack graph is constructed based on the following information: Cloud system information, Virtual network topology and configuration information, Vulnerability information.

4. Network Controller:

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration. 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 also consults with the attack analyzer for the flow access analyzer to construct attack graphs. In NICE, the network control control by setting up the filtering rules on the corresponding OVS and OFS. Network controller is also responsible for applying the countermeasure from attack analyzer. Based on *VM* Security Index and severity of an alert, countermeasures are selected by NICE and executed by the network controller.

IV. System Architecture

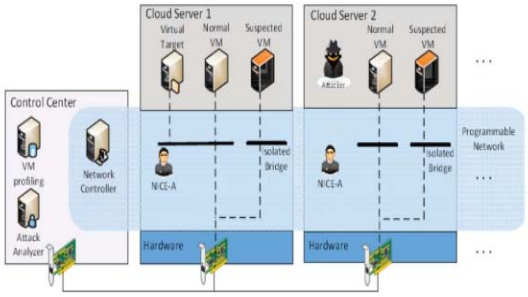

Fig. 1. NICE architecture within one cloud server cluster.

V. Contribution of NICE

The contributions of NICE are presented as follows:

- 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.
- 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. Our study shows that NICE consumes less computational overhead compared to proxy-based network intrusion detection solutions.

VI. NICE MODELS

We developed 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.

1. Threat Model

In 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.

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 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 MulVAL logic attack graph as presented and define as Scenario Attack Graph (*SAG*).

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.

Alert Correlation Graph

Alert Correlation Graph (*ACG*) to map alerts in ACG to their respective nodes in SAG.

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.*

A method for utilizing *SAG* and *ACG* together so as to predict an attacker's behavior. *Alert Correlation* algorithm is followed for every alert detected and returns one or more paths *Si*. For every alert *ac* that is received from the IDS, it is added to *ACG* if it does not exist. For this new alert *ac*, the corresponding vertex in the *SAG* is found by using function *map*(*ac*) (line 3). For this vertex in *SAG*, alert related to its parent vertex of type *NC* is then correlated with the current alert *ac* (line 5). This creates a new set of alerts that belong to a path *Si* in *ACG* (line 8) or splits out a new path *Si*+1 from *Si* with subset of *Si* before the alert *a* and appends *ac* to *Si*+1 (line 10). In the end of this algorithm, the ID of *ac* will be added to *alert* attribute of the vertex in *SAG*. Algorithm 1 returns a set of attack paths *S* in *ACG*.

Algorithm 1 Alert Correlation

Require: alert *ac*, *SAG*, *ACG*

1: **if** (*ac* is a new alert) **then**

2: create node *ac* in *ACG*

3: *n*1 ← *vc* ∈ *map*(*ac*)

4: **for all** *n*2 ∈ *parent*(*n*1) **do**

5: create edge (*n*2.*alert, ac*)

6: **for all** *Si* containing *a* **do**

7: **if** *a* is the last element in *Si* **then**

8: append *ac* to *Si*

9: **else**

10: create path *Si*+1 = *{subset*(*Si, a*)*, ac}*

11: **end if**

12: **end for**

13: add *ac* to *n*1.*alert*

14: **end for**

15: **end if**

16: **return** *S*

3. VM Protection Model

The VM protection model of 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 vulnerabilities present and their impact scores. The impact score of a vulnerability, as defined by the CVSS guide helps judge the confidentiality, integrity,

and availability impact of the vulnerability being exploited. Connectivity metric of a VM is decided by evaluating incoming and outgoing connections.

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.

VII. System Configuration

1. Hardware Configuration

Processor	-	Pentium –IV
Speed	-	1.1 GHz
RAM	-	256 MB(min)
Hard Disk	-	20 GB

Software Configuration:-

2. Operating System: Windows XP
Programming Lang.: JAVA/J2EE

Java Version: JDK 1.6 & above.

VIII. Conclusion

We developed NICE system models and designs to improve security for the system. To reduce the vulnerability attacks by using virtual machines. NICE systems are used by number of organizations to detect the threats.

References

1. Coud Sercurity Alliance, “Top threats to cloud computing v1.0,”
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 environment against 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 environments,” IEEE Security & 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. 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 alerts 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.

