International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-4)

Research Article | **April 2015**

Dynamic Protection and Intelligent Intrusion Detection in Virtual Clouds

D. Parameswari [1], G. Micheal[2], Dr. K. P. Kaliyamurthie[3]
[2]Associate Professor, [3]Head of the Department
[1, 2, 3] Department of Computer Science and Engineering,
Bharath University, Chennai, India

Abstract-

The main purpose of the cloud is outsourcing. It is responsible for managing applications and data at runtime. Attackers explore vulnerabilities of cloud system and compromise virtual machine to deploy large scale DDOS. Vulnerability is the weakness which allows attacker to reduce the system access and has capacity to exploit. DDOS is Distributed Denial Of Service that the attackers send a number of requests to the system and compromise the systems called Zombies, they slow down the performance by preventing from getting service. Virtual Machine is a software implementation of a machine that executes programs like a physical machine in multiple OS environment. Cloud system has service models such as Iaas, Paas, Saas and our model deals with Iaas which is computing infrastructure of virtual machines like Amazon, Windows Azure, Rackspace etc. In Iaas detection of zombie attack is difficult. Cloud user install vulnerable application on virtual machine and compromise them. Hence here we use multiphase distributed vulnerability detection and countermeasure selection mechanism based on analytical graph based model DDDID - Dynamic Defense in Depth Intrusion Detection Framework in Virtual Clouds. Here we increase detection accuracy and scalability by decentralized network.

Keywords— DDDID, DDOS, cloud computing, zombie detection, intrusion detection, virtual machines.

I. INTRODUCTION

We use cloud system resources to deploy. 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 attacks. Cloud users share computing resources which are connected through same switch, same data storage and file systems. Attackers to compromise multiple VMs. In this paper we propose Dynamic Defense in Depth Intrusion Detection Framework (DDDID) and it is a reconfigurable virtual network approach to detect and counter measures to compromise VMs. In general, DDDID includes two main phases: 1) deploy a lightweight mirroring-based network intrusion detection agent (DDDID-A) on each cloud server to capture and analyze cloud traffic. A DDDID-A periodically scans the establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability toward the collaborative attack goals, DDDID 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 virtual system vulnerabilities within a cloud server to DDDID 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. The contributions of DDDID are presented as follows:

- A multiphase distributed network intrusion detection and prevention framework in virtual networking environment.
- It captures and inspects cloud traffic without interrupting user's application and cloud services.
- Software switching solution to inspect doubtful VMs for investigation and protection.
- It improves attack detection probability and improve resilense
- It has attack graph approach
- It minimizes resource computation
- It has less computational overhead compared to proxy based network IDs.

II. RELATED WORK

SPOT, is based on sequentially scanning outgoing messages while employing a statistical method Sequential Probability Ratio Test (SPRT), to quickly determine whether a host has been compromised. BotHunter detects compromised machines based on the fact that a thorough malware infection BotSniffer 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 Binary Decision Diagrams (BDDs) to construct attack graph. Their model can generate all possible attack paths, however, the scalability is a big issue 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 a modified attack-graph-based correlation algorithm to create explicit correlations only by matching alerts to specific exploitation an alert dependencies graph (DG) to group related alerts with multiple correlation criteria. Each path in DG represents a subset of alerts that might be part of an attack scenario. However, their algorithm involved all pairs shortest path searching and sorting in DG, which consumes considerable computing power. An attack countermeasure tree (ACT) to consider attacks and countermeasures together in an attack tree structure. a Bayesian attack graph (BAG) to address dynamic security risk management problem and applied a genetic algorithm to solve countermeasure optimization problem. 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 (OVS) and network controller to apply the selected network countermeasures in our solution.

III. DDDID 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. Attack graph is helpful in identifying potential threats, possible attacks, and known vulnerabilities in a cloud system.

A Scenario Attack Graph An SAG is a tuple SAG =(V,E). CVE is Common Vulnerabilities and Exposures CVSS is Common Vulnerability Scoring System and OVSDB is Open Source Vulnerability Database 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. An ACG is a three tuple ACG=(A,E,P) where A is a set of aggregated alerts consists of source IP address, destination IP address, type of alert and time stamp. E is a directed edges and P is a set of paths Raw alerts having same source and destination IP addresses, attack type, and time stamp within a specified window are aggregated as Meta Alerts. time stamp 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.

Algorithm 1. Alert Correlation

Require: alert ac, SAG, ACG

1. if (ac is a new alert) then
2. create node ac in ACG
3. n1 -vc € map(ac)
4. for all n2 2 parent(n1) do
5. create edge (n2: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 n1:alert
14. end for
15. end if
16. return S

VM Protection Model

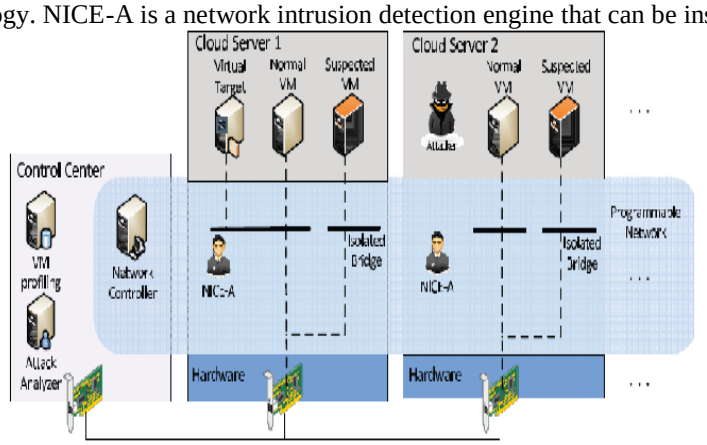
The VM protection model of NICE consists of a VM profiler, a security indexer, and a state monitor. Security index for all the VMs in the network depending upon various factors like connectivity, the number of vulnerabilities present and their impact scores. The impact score of a vulnerability, as defined by the CVSS guide , helps to judge the confidentiality, integrity, and availability impact of the vulnerability being exploited.

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. DDDID SYSTEM DESIGN

The proposed DDDID framework is illustrated in Fig. 1. It shows the DDDID framework within one cloud server cluster. Major components in this framework are distributed and light-weighted DDDID-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 The network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer. Our terminologies are based on the XEN virtualization technology. NICE-A is a network intrusion detection engine that can be installed in either



DDDID architecture within one cloud server cluster

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. An attack graph is established according to the vulnerability information derived Countermeasures are initiated by the attack analyzer based on the evaluation results from the cost benefit analysis of the effectiveness of countermeasures.

System Components

DDDID-A

The DDDID-A is a Network-based Intrusion Detection System (NIDS) agent installed in either Dom0 or DomU in each cloud server. It scans the traffic among VMs and in/out from the physical cloud servers. In our experiment, Snort is used to implement DDDID-A in Dom0. It will sniff a mirroring port on each virtual bridge in the Open vSwitch (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.

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. the effect of compromise of a highly connected VM can cause more damage. An attacker can use port-scanning program to perform an intense examination of the network to look for open ports on any VM. 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.
- DDDID-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).

Attack Analyzer

The major functions of DDDID 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. With this information, attack paths can be modeled using SAG. Each node in the attack graph represents an exploit by the attacker. Each path from an initial node to a goal node represents a successful attack. In summary, DDDID attack graph is constructed based on the following information:

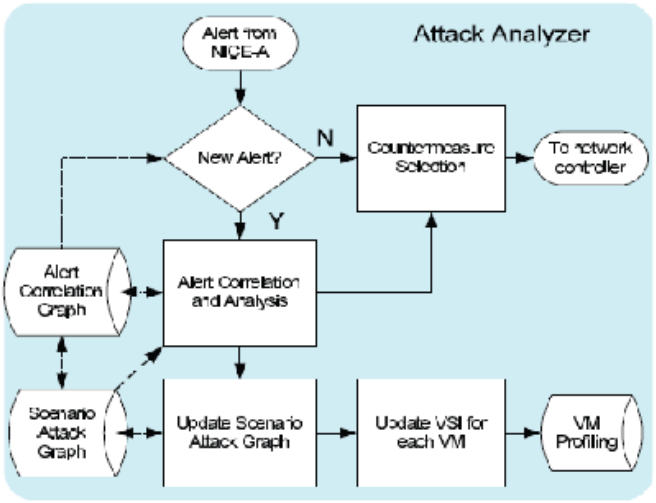
Cloud system information. It is from XEN It consist of number of VMs, running services on VM, VM Virtual Interface Information (VIF).

Virtual network topology and configuration information 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 on demand vulnerability scanning 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 DDDID-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. 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. If the alert is a new vulnerability and is not present in the DDDID attack graph, the attack analyzer adds it to attack graph and then reconstructs it.



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 The communication between cloud servers is handled by physical Open Flow-capable Switch (OFS). The network controller is responsible for collecting network information of current OpenFlow network and provides input to the attack analyzer to construct attack graphs. Through the cloud internal discovery modules that use DNS, DHCP, LLDP, and flow initiations, network controller is able to discover the network connectivity information from OVS and OFS. Another important function of the network controller is to assist the attack analyzer module. Network controller is also responsible for applying the countermeasure from attack analyzer. Based on VM Security Index (VSI) and severity of an alert, countermeasures are selected by DDDID and executed by the network controller.

V. DDDID SECURITY MEASUREMENT, ATTACK

Mitigation, And Countermeasures

When vulnerabilities are discovered or some VMs are identified as suspicious, several countermeasures can be taken to restrict attackers' capabilities and it is important to differentiate between compromised and suspicious VMs. 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. Security measure metrics are Base Score(BS), Common Vulnerability Scoring System(CVSS), National Vulnerability Database(NDB), Impact Value(IV), Confidentiality, Integrity(I), Availability(A), Exploitability, Access vector(AV), Access Complexity(AC), Authentication Instances(AU). DDDID is able to construct the mitigation strategies in response to detected alerts. A countermeasure pool CM = (cm1; cm2; . . . ; cm) is a set of countermeasures cm = (cost, intrusiveness, condition, effectiveness), where

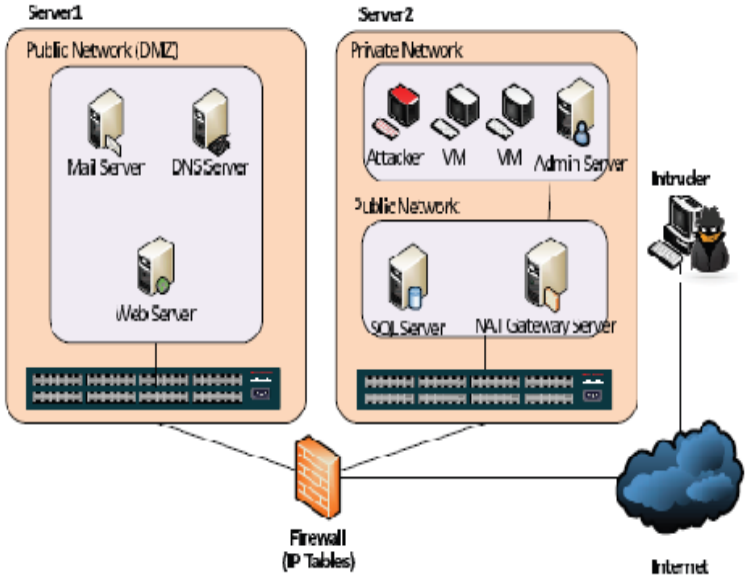
1. Cost is the unit that describes the expenses required to apply the countermeasure in terms of resources and operational complexity, and it is defined in a range from 1to 5, and higher metric means higher cost;
2.Intrusiveness is the negative effect that a countermeasure brings to the SLA and its value 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 countermeasure;
4. Effectiveness is the percentage of probability changes of the node, for which this countermeasure is applied.

In NICE, the network reconfiguration strategies mainly involve two levels of action: Layer-2 and layer-3.

Countermeasure Selection

Countermeasure Selection Algorithm 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 counter measures If the distance is greater than a threshold. value, we do not perform countermeasure selection but update the ACG to keep track of alerts in the system. Because the alert is generated only after the attacker has performed the action. The change in probability of target node

gives the benefit for the applied countermeasure. In the next double for-loop, we compute the Return of Investment (ROI) for each benefit of the applied countermeasure.



Virtual network topology for security evaluation

VI. PERFORMANCE EVALUATION

Our evaluation is conducted in two directions: The
Security performance and the system computing and network reconfiguration.

Security Performance Analysis

Security performance analysis involves the following Environment and configuration that is in private and public VMs which involves SSHD-Secure Shell Protocol Daemon and Alert graph and alert correlation which involves network topology, vulnerability information and generate alerts. The attack graph provides network connectivity, running services, vulnerability ACG can be updated and attacker's behavior of SAG. The countermeasure selection VSI measures the security level of each VM that gives the vulnerability score for VM and exploitability score for VM. VM with higher values of VSI are easily attacked. A false alarm which has zero day attack that is the vulnerability discovered by attacker but is not discovered by vulnerability scanner.

DDDID System Performance

Mirroring based attack detection consist of two Virtual machine in a cloud and normal network traffic is analyzed by SPAN and monitoring network by DDDID-A. Proxy based attack detection consist of two virtual machine deployed by DDDID-A and removes traffic duplication. When running in IPS mode, intercept traffic and packet checking which consumes system resources. The performance evaluation metrics are CPU utilization, Network capacity, Agent processing capacity and communication delay.. The percentage of analyzed packets is defined a s the number of analyzed packets per total number of packet received. The increase in analyzed packets is equal to the increase in the packet the agent handles. Packet drops when more traffic load. Communication delay is equal to the speed of one packet per second. DDDID-A at Dom0 and mirror based DDDID-A at Dom U has better performance in delay.

VII. CONCLUSION

DDDID detects and mitigates collaborative attacks in the cloud virtual networking environment. It is a dynamic defensive mechanism based software defined networking approach that involved multiphase intrusion detections. It achieves in large scale cloud system, design security, less intrusive, cost effective, and CPU and throughput are a better performance. Traffic planning is done. It has detection accuracy and scalability network

REFERENCES

[1] Coud Security Alliance, "Top Threats to Cloud Computing v1.0," https://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 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] Mitre Corporation, "Common Vulnerabilities and Exposures,CVE," http://cve.mitre.org/, 2012.

