Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

New-Multi-Phase Distribution Network Intrusion Detection

J.Sasidevi[1], Dr. R. Sugumar[2], P. Shanmuga Priya[3]
[1, 3] Research Scholar, St.Peter's University, Chennai, India
[2] Associate Professor, Veltech Multitech Engineering College, Chennai, India

Abstract - Cloud security is one of most important issues that have attracted a lot of research and development effort in past few years. Attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service. This paper proposes a multiphase distributed vulnerability detection, measurement, and countermeasure selection mechanism called NICE. NICE is built on attack graph-based analytical models and reconfigurable virtual network-based countermeasures. The proposed framework leverages Open Flow network programming APIs. The attacker is analyzed by attack analyzer and VM profiling.

Keywords: Intrusion Detection, Network Security, NICE, Countermeasure Selection, VM profiling.

I. INTRODUCTION

Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or are assigned an ID and password or other authenticating information that allows them access to information and programs within their authority.

Network security covers a variety of computer networks, both public and private, that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. It does as its title explains: It secures the network, as well as protecting and overseeing operations being done. The most common and simple way of protecting a network resource is by assigning it a unique name and a corresponding password.

Network security starts with authenticating, commonly with a username and a password. Since this requires just one detail authenticating the user name —i.e. the password— this is sometimes termed one-factor authentication. With two-factor authentication, something the user 'has' is also used (e.g. a security token or 'dongle', an ATM card, or a mobile phone); and with three-factor authentication, something the user 'is' is also used (e.g. a fingerprint or retinal scan).

Once authenticated, a firewall enforces access policies such as what services are allowed to be accessed by the network users. Though effective to prevent unauthorized access, this component may fail to check potentially harmful content such as computer worms or Trojans being transmitted over the network. Anti-virus software or an intrusion prevention system (IPS) helps detect and inhibit the action of such malware. An anomaly-based intrusion detection system may also monitor the network like wireshark traffic and may be logged for audit purposes and for later high-level analysis. Communication between two hosts using a network may be encrypted to maintain privacy.

Network Intrusion detection and Countermeasure sElection 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 zombieVMs.

II. RELATED WORKS

BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation present a new kind of network perimeter monitoring strategy, which focuses on recognizing the infection and coordination dialog that occurs during a successful malware infection. BotHunter is an application designed to track the two-way communication flows between internal assets and external entities, developing an evidence trail of data exchanges that match a state-based infection sequence model. BotHunter consists of a correlation engine that is driven by three malware-focused network packet sensors, each charged with detecting specific stages of the malware infection process, including inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog, and outbound attack propagation.

The BotHunter correlator then ties together the dialog trail of inbound intrusion alarms with those outbound communication patterns that are highly indicative of successful local host infection. When a sequence of evidence is found to match BotHunter's infection dialog model, a consolidated report is produced to capture all the relevant events and event sources that played a role during the infection process. This refer to this analytical strategy of matching the

dialog flows between internal assets and the broader Internet as dialog-based correlation, and contrast this strategy to other intrusion detection and alert correlation methods.

III. NICE MODELS

Network Intrusion detection and Countermeasure sElection in virtual network systems (NICE) is used to establish a defense-in-depth intrusion detection framework. It incorporates attack graph analytical procedures into the intrusion detection processes. 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 optimizes the implementation on cloud servers to minimize resource consumption. NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures.

Multi-factor authentication (also MFA, Two-factor authentication, TFA, T-FA or 2FA) is an approach to authentication which requires the presentation of two or more of the three authentication factors: a knowledge factor ("something only the user knows"), a possession factor ("something only the user has"), and an inherence factor ("something only the user is"). After presentation, each factor must be validated by the other party for authentication to occur. Two-factor authentication is commonly found in electronic computer authentication, where basic authentication is the process of a requesting entity presenting some evidence of its identity to a second entity.

3.1 User Request

The cloud user register and login themselves. The user registration and login is done using multifactor authentication. The multifactor authentication includes the identification of username and password, onetime password to users email address and graphical password authentication. They request for the resource they need in the cloud. This request is monitored by Virtual machine Monitoring.

3.2 Cloud server

The user request in the virtual machine monitoring is analyzed by the cloud server. The server response to the user by retrieving the data from the database.

3.3 Virtual Machine Profiling

The VM profiling analyses the attacker in the Virtual machine monitoring System. 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. 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.

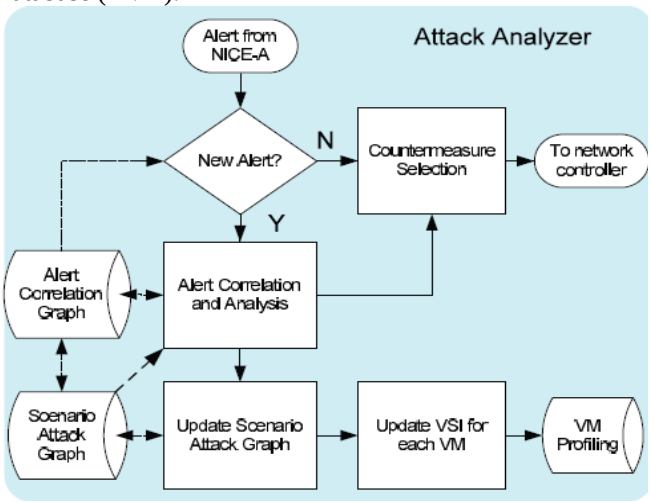
Algorithm 1. Alert_Correlation
Require: alert a_c, SAG, ACG

```
if (a_c is a new alert) then
    create node a_c in ACG
    n_1 ← v_c ∈ map(a_c)
    for all n_2 ∈ parent(n_1) do
        create edge (n_2.alert, a_c)
        for all S_i containing a do
            if a is the last element in S_i then
                append a_c to S_i
            else
                create path S_{i+1} = {subset(S_i, a), a_c}
            end if
        end for
        add a_c to n_1.alert
    end for
end if
return S
```

3.4 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. 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, NICE attack graph is constructed based on the following information:

- Cloud system information is collected from the node controller 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 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 (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).



3.5 Network controller

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration feature based on OpenFlow protocol. In NICE, within each cloud server there is a software switch, for example, OVS, which is used as the edge switch for VMs to handle traffic in and out from VMs. The communication between cloud servers (i.e., physical servers) is handled by physical Open Flow-capable Switch (OFS).

IV. COUNTER MEASURE SELECTIONS

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 counter measuresCM. The algorithm starts by selecting the node vAlert that corresponds to the alert generated by a ICE-A. Before selecting the countermeasure, we count the distance of vAlert to the target node.

Algorithm 2. Countermeasure_Selection
Require: $Alert, G(E, V), CM$
Let v_{Alert} = Source node of the $Alert$
if Distance_to_Target(v_{Alert}) $> threshold$ **then**
  Update_ACG
  return
end if
Let $T = Descendant(v_{Alert}) \cup v_{Alert}$
Set $Pr(v_{Alert}) = 1$
Calculate_Risk_Prob(T)
Let $benefit[|T|, |CM|] = \emptyset$
for each $t \in T$ **do**
  for each $cm \in CM$ **do**
    if $cm.condition(t)$ **then**
      $Pr(t) = Pr(t) * (1 - cm.effectiveness)$
      Calculate_Risk_Prob($Descendant(t)$)
      $benefit[t, cm] = \Delta Pr(target\_node)$.
    end if
  end for
end for
Let $ROI[|T|, |CM|] = \emptyset$
for each $t \in T$ **do**
  for each $cm \in CM$ **do**
    $ROI[t, cm] = \frac{benefit[t,cm]}{cost.cm + intrusiveness.cm}$.
  end for
end for
Update_SAG and Update_ACG
return Select_Optimal_CM(ROI)

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 (line 3). For the source node vAlert, all the reachable nodes (including the source node) are collected into a set T (line 6). Because the alert is generated only after the attacker has performed the action, we set the probability of vAlert to 1 and calculate the new probabilities for all of its child (downstream) nodes in the set T (lines 7 and 8). Now, for all t 2 T the applicable countermeasures in CM are selected and new probabilities are calculated according to the effectiveness of the selected countermeasures (lines 13 and 14). The change in probability of target node gives the benefit for the applied countermeasure using (7). In the next double for-loop, we compute the Return of Investment (ROI) for each benefit of the applied countermeasure based on (8). 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.

V. 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.

REFERENCES

[1] Coud Sercurity 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] 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.

