Indian Journal of Communication Engineering and Systems (IJCES)
Vol.2.No.1 2014 pp 12.15.
available at: www.goniv.com
Paper Received :08-03-2014
Paper Accepted:22-03-2013
Paper Reviewed by: 1. R. Venkatakrishnan 2. R. Marimuthu
Editor : Prof. P.Muthukumar

DETECTING INTRUSION ZOMBIES IN SCALABLE NETWORK INFRASTRUCTURE

G.Saranya, K. Vijayakumar,
S.K.P Engineering College,
Tiruvannamalai ,india
Saranyagme@gmail.com, Vijayktvm@skpec.in

ABSTRACT

In Cloud networks are vulnerable to spoofing attacks, which allows for many other forms of attacks on the networks. Although the identity of a node can be verified through cryptographic authentication, authentication is not always possible because it requires key management and additional infrastructural overhead. In this paper we propose a method for both detecting spoofing attacks, as well as locating the positions of adversaries performing the attacks. We first propose an attack detector for cloud spoofing that utilizes MAC(Media access Control) and RSS(Received Signal strength) analysis. Next, we describe how we integrated our attack detector into a real-time indoor localization system, which is also capable of localizing the positions of the attackers. We then show that the positions of the attackers can be localized using either area-based or point-based localization algorithms with the same relative errors as in the normal case. Our results show that it is possible to detect cloud spoofing with both a high detection rate and a low false positive rate.

Index Terms—Network Security, Cloud Computing, Intrusion Detection, Attack Graph , Zombie Detection.

1. INTRODUCTION

RECENT studies have shown that 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 [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 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 multiple VMs.

In this article ,We proposed an defend against flood attacks on the Internet and in wireless sensor networks, they assume persistent connectivity and cannot be directly applied to DTNs that have intermittent connectivity.

A scheme to detect resource misuse in DTN . In their scheme, the gateway of a DTN monitors the activities of nodes and detects an attack if there is deviation from expected behavior. Different from their work that requires a special gateway for counting; our scheme works in a totally distributed manner and requires no special nodes. A scheme which exploits claim- carry- and -check to probabilistically detect the violation of rate limit in DTN environments . Our scheme uses efficient constructions to keep the computation, communication and storage cost low. Also, we analyzed the lower bound and upper bound of detection probability.

Extensive trace-driven simulations showed that our scheme is effective to detect flood attacks and it achieves such effectiveness in an efficient way. A distributed manner, not relying on any online central authority or infrastructure, which well fits the environment of the nodes involved are admin and clients which stands as UI for the system. The deployment is performed as per the requirements of Hardware and software specified in the requirements phase.

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. (2) Once a VM enters inspection state, Deep Packet Inspection (DPI).

2. ARCHITECTURE DESIGN

System Design Overview

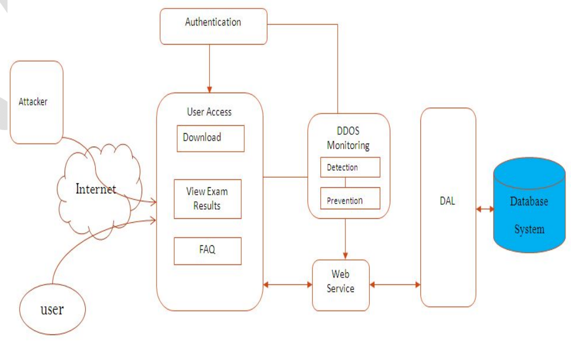
The proposed MAC is illustrated in figure in 1. It shows the NICE framework within one cloud server cluster [1]. Major components in this framework are distributed and light-weighted NICE-A on each physical cloud server, a network controller, a VM profiling server, and an attack analyzer [2]. The latter three components are located in a centralized control center connected to software switches on each cloud built on one or multiple Linux software bridges). 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 [5]. 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 Dom U 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 [6].

After receiving an alert, attack analyzer evaluates the severity of the alert on the attack graph, decides what countermeasure strategies to take, and then initiates it through the network controller.

3. VULNERABILITY SCANS

An attack graph is established according to the vulnerability information derived from both offline and real time vulnerability scans[7]. 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 Open Flow switches.

Architecture of MAC:



In Author [1] 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 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. EVALUATION

In Author [1] 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.

5. SECURITY ANALYSIS

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

6. RELATED WORKS

In this section, we present literatures of several highly related research areas to NICE, including: zombie detection and prevention, attack graph construction and security analysis, and software defined networks for attack countermeasures. 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.

O. Sheyner *et al.* [9] proposed a technique based on a modified symbolic model checking NuSMV [6] 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.* [8] introduced the assumption of monotonicity, which states that the precondition of a given exploit is never invalidated by the successful application of another exploit.

The primary goal of alert correlation is to provide system support for a global and condensed view of network attacks by analyzing raw alerts [1]. Many attack graph based alert correlation techniques have been proposed recently. L. Wang *et al.* [7] devised an in-memory structure, called *queue graph* (QG), to trace alerts matching each exploit in the attack graph.

7. CONCLUSIONS

In this paper, we employed rate limiting to mitigate flood attacks in DTNs, and proposed a scheme which exploits claim-carry-and-check to probabilistically detect the violation of rate limit in DTN environments. Our scheme uses efficient constructions to keep the computation, communication and storage cost low. Also, we analyzed the lower bound and upper bound of detection probability. Extensive trace-driven simulations showed that our scheme is effective to detect flood attacks and it achieves such effectiveness in an efficient way. Our scheme works in a distributed manner, not relying on any online central authority or infrastructure, which well fits the environment of DTNs. Besides, it can tolerate a small number of attackers to collude.

REFERENCES

[1]. 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

[2].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.

[3] 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.

[4] 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

[5]. “Openflow.” http://www.openflow.org/wp/learnmore/, 2012.

[6]. Open Networking Fundation, “Software-defined networking: The new norm for networks,” *ONF White Paper*, Apr. 2012.

[7].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.

[8]X. Ou, S. Govindavajhala, and A. W. Appel, “MulVAL: a logic based network security analyzer,” *Proc. of 14th USENIX Security Symp.*, pp. 113–128. 2005.

{9].L. Wang, A. Liu, and S. Jajodia, “Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts,” *Computer Communications*, vol. 29, no. 15, pp. 2917–2933, Sep. 2006.

[10] S. Roschke, F. Cheng, and C. Meinel, “A new alert correlation algorithm based on attack graph,” *Computational Intelligence in Security for Information Systems*, LNCS, vol. 6694, pp. 58–67. Springer, 2011.

[11] A. Roy, D. S. Kim, and K. Trivedi, “Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees,” *Proc. IEEE Int’l Conf. on Dependable Systems Networks (DSN ’12)*, Jun. 2012