NETWORK INTRUSION DETECTION AND COUNTERMEASURE SELECTION IN VIRTUAL NETWORK SYSTEMS

Gomathy.M
Department of Computer Science, Shrimati Indira Gandhi College, Trichirapalli, India.
Email: m.gomathy.phd@gmail.com

Abstract
Cloud security is one of most important issues that has attracted a lot of research and development effort in past few years. Particularly, attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service (DDoS). DDoS attacks usually involve early stage actions such as multi-step exploitation, low frequency vulnerability scanning, and compromising identified vulnerable virtual machines as zombies, and finally DDoS attacks through the compromised zombies. Within the cloud system, especially the Infrastructure-as-a-Service (IaaS) clouds, the detection of zombie exploration attacks is extremely difficult. This is because cloud users may install vulnerable applications on their virtual machines. To prevent vulnerable virtual machines from being compromised in the cloud, we propose a multi-phase 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 network programming APIs to build a monitor and control plane over distributed programmable virtual switches in order to significantly improve attack detection and mitigate attack consequences. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution.

Keywords: Network Intrusion, Detection, Virtual Networks and Countermeasure.

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, in which attackers can exploit vulnerabilities in clouds and utilize cloud system resources to deploy attacks.

Problem Statement
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.

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. 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.

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.

LITERATURE SURVEY
BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation
Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee, College of Computing Computer Science Laboratory Georgia Institute of Technology SRI International 266 Ferst Drive 333 Ravenswood Avenue Atlanta
We 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. We 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. We present our experimental results using BotHunter in both virtual and live testing environments, and discuss our Internet release of the BotHunter prototype. BotHunter is made available both for operational use and to help stimulate research in understanding the life cycle of malware infections.

BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic
Guofei Gu, Junjie Zhang, and Wenke Lee, School of
Computer Science, College of Computing; Georgia Institute of Technology

Botnets are now recognized as one of the most serious security threats. In contrast to previous malware, botnets have the characteristic of a command and control (C&C) channel. Botnets also often use existing common protocols, e.g., IRC, HTTP, and in protocol-conforming manners. This makes the detection of botnet C&C a challenging problem. In this paper, we propose an approach that uses network-based anomaly detection to identify botnet C&C channels in a local area network without any prior knowledge of signatures or C&C server addresses. This detection approach can identify both the C&C servers and infected hosts in the network. Our approach is based on the observation that, because of the pre-programmed activities related to C&C, bots within the same botnet will likely demonstrate spatial-temporal correlation and similarity. For example, they engage in coordinated communication, propagation, and attack and fraudulent activities. Our prototype system, BotSniffer, can capture this spatial-temporal correlation in network traffic and utilize statistical algorithms to detect botnets with theoretical bounds on the false positive and false negative rates. We evaluated BotSniffer using many real-world network traces. The results show that BotSniffer can detect real-world botnets with high accuracy and has a very low false positive rate.

Detecting Spam Zombies by Monitoring Outgoing Messages
Zhenhai Duan, Peng Chen, Fernando Sanchez, Florida State University; Yingfei Dong, University of Hawaii; Mary Stephenson, James Barker, Florida State University

Compromised machines are one of the key security threats on the Internet; they are often used to launch various security attacks such as spamming and spreading malware, DDoS, and identity theft. Given that spamming provides a key economic incentive for attackers to recruit the large number of compromised machines, we focus on the detection of the compromised machines in a network that are involved in the spamming activities, commonly known as spam zombies. We develop an effective spam zombie detection system named SPOT by monitoring outgoing messages of a network. SPOT is designed based on a powerful statistical tool called Sequential Probability Ratio Test, which has bounded false positive and false negative error rates. Our evaluation studies based on a two-month email trace collected in a large U.S. campus network show that SPOT is an effective and efficient system in automatically detecting compromised machines in a network. For example, among the 440 internal IP addresses observed in the email trace, SPOT identifies 132 of them as being associated with compromised machines. Out of the 132 IP addresses identified by SPOT, 126 can be either independently confirmed (110) or highly likely (16) to be compromised. Moreover, only 7 internal IP addresses associated with compromised machines in the trace are missed by SPOT.

In addition, we also compare the performance of SPOT with two other spam zombie detection algorithms based on the number and percentage of spam messages originated or forwarded by internal machines, respectively, and show that SPOT outperforms these two detection algorithms.

MulVAL: A Logic-based Network Security Analyzer
Xinming Ou Sudhakar Govindavajhala Andrew W. Appel, Princeton University

To determine the security impact software vulnerabilities have on a particular network, one must consider interactions among multiple network elements. For a vulnerability analysis tool to be useful in practice, two features are crucial. First, the model used in the analysis must be able to automatically integrate formal vulnerability specifications from the bug-reporting community. Second, the analysis must be able to scale to networks with thousands of machines. We show how to achieve these two goals by presenting MulVAL, an end-to-end framework and reasoning system that conducts multihost, multistage vulnerability analysis on a network. MulVAL adopts Datalog as the modeling language for the elements in the analysis (bug specification, configuration description, reasoning rules, operating-system permission and privilege model, etc.). We easily leverage existing vulnerability-database and scanning tools by expressing their output in Datalog and feeding it to our MulVAL reasoning engine. Once the information is collected, the analysis can be performed in seconds for networks with thousands of machines.

**METHODOLOGY**

Existing system

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.

**Drawbacks**

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).

Detecting malicious behavior

1)Duan et al. focused 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.

2)BotHunter detected 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.

3)BotSniffer exploited 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, for example a state where an attacker has obtained administrative access to a machine. There are many automation tools to construct attack graph.

**Binary Decision Diagrams (BDDs)**

O. Sheyner et al. proposed a technique based on a modified symbolic model checking NuSMV and Binary Decision Diagrams (BDDs) to construct attack graph.

**Drawbacks**

Their model can generate all possible attack paths, however, the scalability is a big issue for this solution.

**Intrusion Detection System**

IDS and firewall are widely used to monitor and detect suspicious events in the network.

**Drawbacks**

The false alarms and the large volume of raw alerts from IDS are two major problems for any IDS implementations.

Many attack graph based alert correlation techniques have been proposed recently. L. Wang et al. devised an in-memory structure,
called queue graph (QG), to trace alerts matching each exploit in the attack graph.

**Drawbacks**

The implicit correlations in this design make it difficult to use the correlated alerts in the graph for analysis of similar attack scenarios.

Roschke et al. proposed a modified attack-graph-based correlation algorithm to create explicit correlations only by matching alerts to specific exploitation nodes in the attack graph with multiple mapping functions, and devised an alert dependencies graph (DG) to group related alerts with multiple correlation criteria.

**Attack countermeasure tree**

Roy et al. 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.

**Drawbacks**

However, their solution focuses on a static attack scenario and predefined countermeasure for each attack.

N. Poolsappasit et al. proposed a Bayesian attack graph (BAG) to address dynamic security risk management problem and applied a genetic algorithm to solve countermeasure optimization problem.

**ANALYSIS OF THE STUDY**

**Proposed System**

1) NICE (Network Intrusion detection and Countermeasure sElection in virtual network systems) is proposed to establish a defense-in-depth intrusion detection framework.
2) For better attack detection, NICE incorporates attack graph analytical procedures into the intrusion detection processes.
3) The design of NICE does not intend to improve any of the existing intrusion detection algorithms; instead, NICE employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs.
4) 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 towards the collaborative attack goals, NICE will decide whether or not to put a VM in network inspection state.
5) 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.
6) 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.
7) NICE enables the cloud to establish inspection and quarantine modes for suspicious VMs according to their current vulnerability state in the current SAG.
8) 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.

**Advantages**

1) 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.
2) 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.
3) 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.
4) NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures.
5) 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.

**System Architecture**

![System Architecture Diagram]

**Hardware Specification**

- **Processor**: Any Processor above 500 MHz.
- **Ram**: 128Mb.
- **Hard Disk**: 10 GB.
- **Input device**: Standard Keyboard and Mouse.
- **Output device**: VGA and High Resolution Monitor.

**Software Specification**

- **Operating System**: Windows Family.
- **Pages developed using**: Java Server Pages and HTML.
- **Techniques**: Apache Tomcat Web Server 5.0, JDK 1.5 or higher
- **Web Browser**: Microsoft Internet Explorer.
- **Data Base**: MySQL 5.0

**Modules**

**Cloud cluster**

Consider a cloud cluster, which consists of two cloud server. It shows the NICE framework within one cloud server cluster. Major components in this framework are distributed and light-weighted 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 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. The network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer.

**Attacker module**

Attacker module is designed to produce DDOS attack on cloud server. The user who is giving more requests to cloud server is considered to be attacker. Intrusion detection alerts are sent to control center when suspicious or anomalous traffic is detected.

**Intrusion detection**

When there is a considerable anomalous traffic is there, then there is an intrusion can be detected. When the traffic exceeds the threshold level, there is considered to be the intrusion in the network.

**Attack graph**

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.

**Attack counter measure**

As there are number of counter measure can be taken depending on the attack severity, we here established the anomalous traffic to cloud server and attack the cloud server. When the cloud server is attacked, the client request on the queue will be dropped. To avoid this, the clients requests to be redirected to another existing cloud server. The client request will be processed by second virtual machine.

**CONCLUSION**

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.

**Future Enhancements**

NICE only investigates the network IDS approach to counter zombie explorative attacks. In order 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**