A Novel Architecture for Real-time Automated Intrusion Detection Fingerprinting using Honeypot

Muhammad Shoaib Siddiqui, Choong Seon Hong
Dept. of Computer Engineering, Kyung Hee University

Abstract

As the networking and data communication technology is making progress, there has been an augmented concern about the security. Intrusion Detection and Prevention Systems have long being providing a reliable layer in the field of Network Security. Intrusion Detection System works on analyzing the traffic and finding a known intrusion or attack pattern in that traffic. But as the new technology provides betterment for the world of the Internet; it also provides new and efficient ways for hacker to intrude in the system. Hence, these patterns on which IDS & IPS work need to be updated. For detecting the power and knowledge of attackers we sometimes make use of Honey-pots. In this paper, we propose a Honey-pot architecture that automatically updates the Intrusion's Signature Knowledge Base of the IDS in a Network†.

1. Introduction

Advances in computer and communication technologies have resulted in highly distributed systems that allow users to access information and resources from all over the world. This interconnectivity emphasizes the longstanding problem of providing security by introducing many more possible attacking points. We are witnessing a rapid increase in the number of reported intrusions, break-ins and computer thefts [1] results in a growing need for effective computer and network security measures. Even with the most advanced and state of the art protection, computer networks are still not completely secure. In fact, most security experts agree that, given user-desired features such as network connectivity, we'll never achieve the goal of a completely secure system.
As a result, we must develop Intrusion Detection & Prevention techniques and systems to discover and react to network attacks. So as a positive protection technology, IDS/IPS becomes the focus in the field of network security at present [2].

1.1. Intrusion Detection & Prevention Systems

An intrusion detection system (IDS) generally detects unwanted manipulations to computer systems, through the Internet and the local network. The manipulations may take the form of attacks by skilled malicious hackers, or script kiddies using automated tools.
An IDS is composed of several components: Sensors which generate security events, a Console to monitor events and alerts and control the sensors, and a central Engine that records events logged by the sensors in a database and uses a system of rules to generate alerts from security events received. The rules define the intrusions and need to be updated by the period of time because new techniques and tools for intruding and attacking are being evolved all the time. In an IDS, the sensors are located at choke points in the network to be monitored, often in the demilitarized zone (DMZ) or at network borders. The sensor captures all network traffic and analyzes the content of individual packets for malicious traffic. The traffic is then analyzed and compared with the known malicious knowledge. If a malicious activity is found then the concerned authorities or systems are made known of the intrusion in the network or a specified action is taken.

1.2 Honey Pot

Honey pot is basically a "decoy" system [A honey pot can be defined as [3]] that has a non-hardened operating system or one that appears to have several vulnerabilities for easy access to its resources [4]. The decoy system is set up in a similar manner to those of the production servers in the corporation and should be loaded with numerous fake files, directories, and other information that may look real. By making the honey pot appear to be a legitimate machine with legitimate files, it leads the hacker to believe that they have gained access to important information. In a word, honey pot provides an environment where intruders can he trapped or vulnerabilities accessed before an attack is made on real assets.

1.3 Motivation

Honey pot is a very good tool for collecting information about the attackers, hackers and intruders. It gives the knowledge about the abilities of the attackers and also about the tools available to them. Honey pot maintains a log of the activities the attacker performed, so, the algorithm used by the attacker is also visible in this log. Hence, we can extract the attack pattern or the intrusion signature out of the attackers' activity logs. These patterns can be used by the IDS or IPS to defend against the attackers. If we perform this task in real-time then we have a higher probability of detection intrusions as the knowledge base of the IDS/IPS would always be up-to-date.

† This paper was supported by MIC and ITRC Project.

2. Related Work

Many different approaches to building detection models have been proposed. Several architectures are proposed for the Intrusion Detection Systems but most of these approaches use a database of known signatures or algorithms to determine what production traffic is and what malicious activity is. However, information overload, unknown activity, false positives and false negatives can make analyzing and determining activity extremely difficult.

Tian et all [5] have given an architecture for intrusion detection using honey-pot in which they have used a honey pot to find unknown intrusions. But the honey pot is used to identify the system vulnerabilities but not the knowledge base of the IDS is updated.

More work has been done on Worm Fingerprinting to automatically detect the newly spreading worms and stop their propagation. Singh et all [6] has discussed the previous techniques and proposed their own automated worm fingerprinting architecture.

2.1. Automated Fingerprinting

Wherever IDS/IPS is used, it requires a knowledge base of known intrusion patterns and signature to identify an attack on the network system. Experts of network security identify these patterns and build the knowledge base for the IDS/IPS. Hence, it requires a lot of time while finding vulnerability or a backdoor in a system and documenting it or identifying and building an intrusion pattern out of it. Then more time is spent in updating the knowledge base of the IDS/IPS.

There a better option is to opt for an automated system that identify the intrusion pattern and document it to be available for updates for IDS/IPS. Automated Fingerprinting is used to identify the spreading worms in the Internet and the new techniques used by attackers and hacker to compromise a system [8]. Honey pot can serve nicely in this regards, as they attract an attacker or a worm and due to its log maintenance, the intrusion pattern can be identifies easily [10].

3. Architecture for Automated Intrusion Fingerprinting

We propose architecture for implementing a secure network, which updates the knowledge base of its IDS/IPS by identifying intrusion patterns in the attackers' activities.

3.1 Architecture

Our proposed architecture is very simple and also cost effective. A typical secure network has an IDS/IPS, firewalls and DeMilitarized Zone (DMZ) gateway. Our architecture just introduces a system which would serve as a honey pot in the DMZ as shown in figure 1.

Figure 1 shows a network based on our architecture. We have an IDS inside our network behind the firewall, which is a usual line of defense. Whatever traffic comes into the network is monitored by the IDS, which analyze the traffic and tries to find known intrusion pattern in the network traffic. Whenever an intrusion is detected; an alert is alarmed like IDS usually does.

The new thing in our system is the Honey-pot that is placed

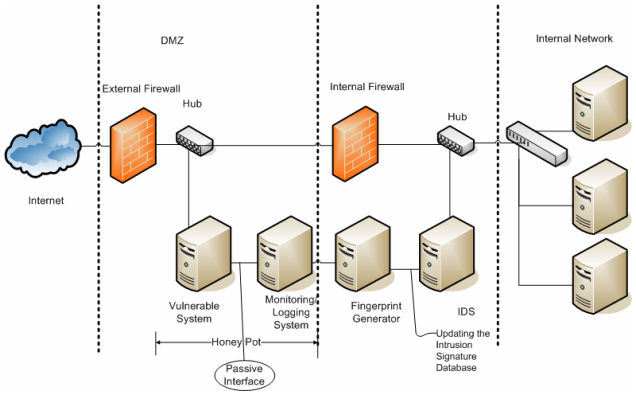

Figure 1: Automated Intrusion Fingerprinting Architecture for IDS

inside the Demilitarized Zone. In figure 1, the Honey-pot has two components which are shown to run on separate systems but both of these tasks can be performed by a single system. This honey-pot is a vulnerable system that is placed for the intruders to attack. Attacker would like to intrude in this system because it is decoyed as a DMZ gateway and for any network the DMZ gateway is the first line of defense. While the attacker is trying to compromise this system all the activities would be logged on the system. The honey-pot system is connected to the IDS system by a passive interface, so all these logs can be delivered to the IDS. Due to this passive interface, the attacker cannot access the IDS and would have no knowledge of it. We have also placed a firewall outside the demilitarized zone because we don't want the attacker to identify the honey-pot.

When IDS receive the log, it performs an intrusion pattern detection algorithm on this log and if there is any intrusion in the honey-pot system, an automated intrusion signature creation algorithm is applied on the activity log. When an intrusion fingerprint or signature is successfully created, the knowledge base of the IDS is updated accordingly.

4. Automated Signature Generation

After an Intrusion activity is detected in the Honey-pot; we perform the signature creation algorithm on the activity log. Our proposal would only work if we are able to produce automatic intrusion patterns or signature in real-time. For this the critical necessity is the signature creation algorithm. Diebold et all [9] use honey pots to trace unknown intrusion but our approach is quite similar to Kreibich et all [8].

We apply our algorithm to newly detected intrusion patterns based on previous intrusion signatures by analyzing the network traffic. Our approach is not limited to a certain communication layer protocol; we treat each and every packet equally. Each received packet initiates the same sequence of activities, which are quite similar to the approach discussed in Honeycomb [8]. Figure 2 illustrates the algorithm. Each activity is briefly explained in more detail in the following sections.

4.1. Connection Tracking

We assert all the connection, either TCP or UDP that the system has made with the attacker's machine or a zombie machine. We maintain the state of each connection as we don

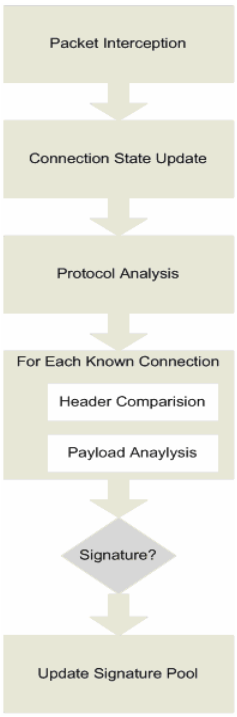

Figure 2: Flow chart of the signature (fingerprint) creation algorithm

not know which one is responsible for the intrusion. When a connection is terminated, we do not release it at once, but also keep track of it until we are sure that it will not benefit us in detecting the intrusion pattern.

4.2. Protocol Analysis

After updating connection state, we create an empty signature record for the flow and starts inspecting the packet. Each signature record has a unique identifier and stores discovered facts (i.e., characteristic properties) about the currently investigated traffic independently of any particular NIDS signature language. The signature record is then augmented continuously throughout the detection process, maintaining a count of the number of facts recorded.
We perform protocol analysis at the network and transport layers for IP, TCP and UDP packet headers. Instead of correcting detected anomalies, we record them in the signature, for example invalid IP fragmentation offsets or unusual TCP flag combinations.

4.3. Pattern Detection in Flow Content

After protocol analysis, we perform intrusion pattern detection in flow control because the attacker can use intrusion pattern in more than one packet or connection. First we perform horizontal intrusion pattern detection i.e. in different messages of a same connection.
After that vertical intrusion pattern detection is performed, in which we try to find the pattern in different connection with similar messages. In either case, if a common substring is found that exceeds a configurable minimum length, the substring is added to the signature as a new payload byte pattern.

4.4. Signature Output

The signature is updated whenever a susceptible intrusion pattern is found. The signature is made according to IDS standards such as of SNORT [11]. If no intrusion pattern is found or a similar signature to one already known is found then this signature is discarded after a certain period of time when there is no connection tracking is maintained for that signature.

5. Conclusion & Future Work

We have proposed an architecture which makes use of honey pot in learning the unknown intrusion patterns and then update these patterns in the knowledge base of the IDS/IPS. Our proposed system makes use of a signature creation algorithm which is automated system and work in real-time. As the system is real-time, hence, no time in wasted in updating the knowledge base of the IDS/IPS and further enhances the reliability of the IDS/IPS implemented in the system.

References

[1]. Stefan Axelsson. “Intrusion Detection Systems: A Survey and Taxonomy”. Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.
[2]. William Stallings, “Network Security Essentials”, Third Edition, Prentice Hall, July 2006;
[3]. David Klug, “Honey Pots and Intrusion Detection” September 13,2000
[4]. http:/lwww.sans.org/rr/intrusion/honey pots. php.
[5]. Lance Spitzner. “Honeypots: Tracking Hackers”, Addison-Wesley, 2003.
[6]. Zhi-Hong Tian, Bin-Xing Fang and Xiao-Chun Yun, "An Architecture For Intrusion Detection Using Honey Pot" in the proceedings of International Conference on Machine Learning and Cybernetics, 2-5 Nov. 2003 Volume: 4, On page(s): 2096-2100
[7]. Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage, "Automated Worm Fingerprinting," in Proceeding of Usenix Symposium of Operating System Design and Implementation, Usenix Association, 2004, pp. 45–60.
[8]. S. Singh, C. Estan, G. Varghese, and S. Savage. “The EarlyBird System for Real-time Detection of Unknown Worms” Technical Report CS2003-0761, CSE Department, UCSD, Aug. 2003.
[9]. C. Kreibich and J. Crowcroft, “Honeycomb - Creating Intrusion Detection Signatures Using Honeypots” in 2nd Workshop on Hot Topics in Networks (HotNets-II), 2003.
[10]. Patrick Diebold, Andreas Hess, Günter Schäfer, “A Honeypot Architecture for Detecting and Analyzing Unknown Network Attacks” in proceeding of 14th Kommunikation in Verteilten Systemen (KiVS2005), Kaiserslautern, Germany, February 2005.
[11]. V. Paxson, “Bro: A System for Detecting Network Intruders in Real-Time” in Computer Networks, Amsterdam, Netherlands, 1999, vol. 31, no. 23-24, pp. 2435.2463.
[12]. Snort: Open Source Network Intrusion Detection System. www.snort.org, 2002.