International Journal of Intelligent Information Systems
2014; 3(6-1): 23-27
Published online October 20, 2014 (http://www.sciencepublishinggroup.com/j/ijiis)
doi: 10.11648/j.ijiis.s.2014030601.14
ISSN: 2328-7675 (Print); ISSN: 2328-7683 (Online)

Improving honeyd for automatic generation of attack signatures

Motahareh Dehghan, Babak Sadeghiyan

Department of Computer Engineering and Information Technology, Amirkabir University of Technology (AUT), Tehran, Iran

Email address:
Motahareh479@aut.ac.ir (M. Dehghan), basadegh@aut.ac.ir (B. Sadeghiyan)

To cite this article:
Motahareh Dehghan, Babak Sadeghiyan. Improving Honeyd for Automatic Generation of Attack Signatures. *International Journal of Intelligent Information Systems.* Special Issue: Research and Practices in Information Systems and Technologies in Developing Countries. Vol. 3, No. 6-1, 2014, pp. 23-27. doi: 10.11648/j.ijiis.s.2014030601.14

Abstract: In this paper, we design and implement a new Plugin to Honeyd which generates attack signature, automatically. Current network intrusion detection systems work on misuse detectors, where the packets in the monitored network are compared against a repository of signatures. But, we focus on automatic signature generation from malicious network traffic. Our proposed system inspects honeypot traffic and generates intrusion signatures for unknown traffic.The signature is based on traffic patterns, using Longest Common Substring (LCS) algorithm. It is noteworthy that our system is a plugin to honeyd - a low interaction honeypot. The system's output is a file containing honeypot intrusion signatures in pseudo-snort format. Signature generation system has been implemented for Linux Operating System (OS) but due to the common use of Windows OS, we implement for Windows OS, using C programming language.

Keywords: Honeypot, Honeyd, Signature, Intrusion Detection System (IDS), Longest Common Substring (LCS) Algorithm

1. Introduction

Nowadays, in order to reduce the effects of network attacks and prevent network intrusion, several security equipments are designed and implemented. One of them is honeypot.

Honeypot offers a variety of services and attracts attackers.

In this paper, we acquire patterns from honeypot's traffic on basis of packets sent to multiple hosts from attackers, which have approximately similar content. Then, from these patterns we generate signatures. The system's output is a file containing intrusion signatures in pseudo- snort format. As illustrated in Figure 1, our proposed system is a plugin to honeyd-low interaction honeypot that designs appropriate responses based on these signatures [1]. Also, these signatures can be used to filter the traffic directed towards the honeypot, in order to reduce the amount of traffic needed to be processed by the honeypot sensors.

2. Data and Materials

2.1. Honeypot

As mentioned by Grønland [2], honeypot is a system which is built and set up in order to be hacked. Besides this, honeypot is also a trap system for the attackers which is deployed to counteract the resources of the attacker and slow him down, thus he wastes his time on the honeypot instead of attacking the production systems.

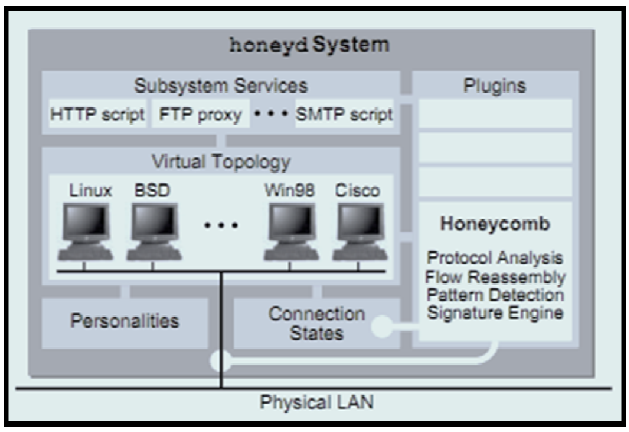

Figure 1. *Honeyd and Proposed System.*

- Types of honeypots include [3,4] Honeypots in terms of reality Physical honeypots
- A physical honeypot is a real machine in network which

has a particular IP address.

- Virtual honeypots A virtual honeypot is simulated by another machine.
- Honeypots in terms of interaction by attacker Low interaction honeypots A low-interaction honeypot will typically run or emulate a small number of services on a real or emulated operating system.
- High interaction honeypots A high-interaction honeypot is often a real computer running a real operating system (Figure 2).

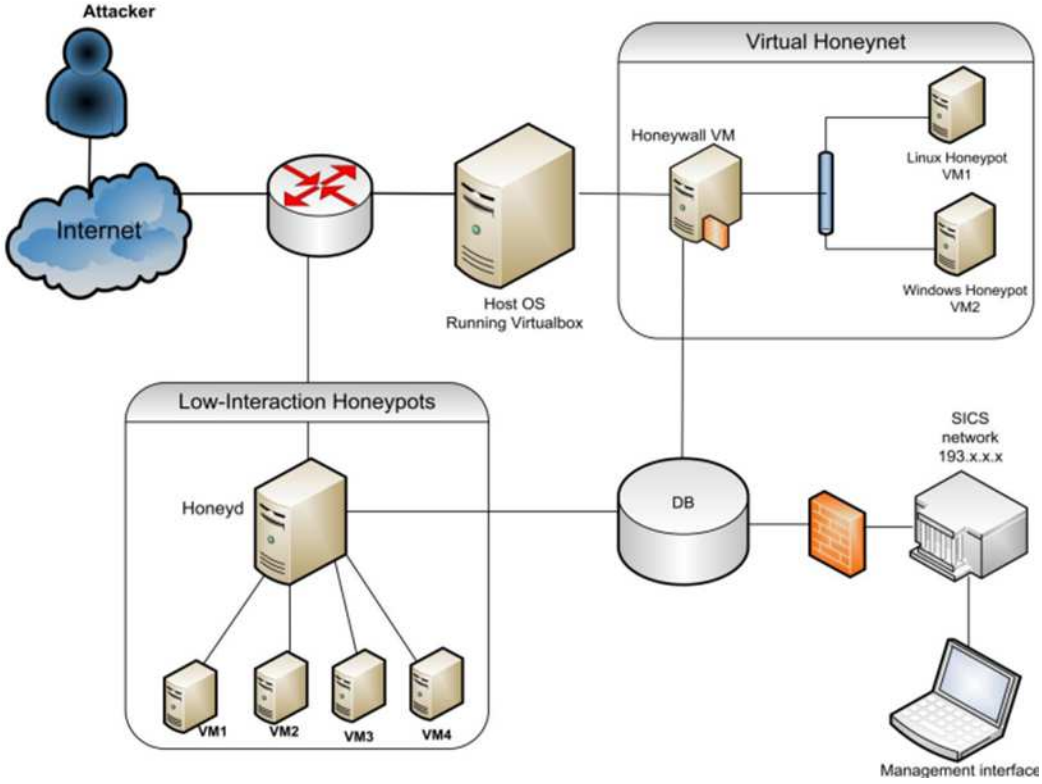

Figure 2. *Types of Honeypots.*

2.2. Honeyd

Honeyd is an Open Source low-interaction honeypot implemented for UNIX and Windows Operating Systems (OS) [5]. Every attacker which intends to communicate network with useless Internet Protocol (IP) Address, is disconnected and interacted by honeyd disconnects this connection and interacts with him. Honeyd is a framework for virtual honeypots which allows thousands of IP addresses to communicate with virtual machines. Thus, it should be able to simulate network topology. Honeyd is a central machine which captures the traffic directed towards the virtual honeypots and simulates appropriate responses [6].

2.3. Longest Common Substring of Two Strings

The longest common substrings of a set of strings can be found by building a generalised suffix tree for the strings, and then finding the deepest internal nodes which have leaf nodes from all the strings in the subtree below it [7]. The figure on the right is the suffix tree for the strings "ABAB", "BABA" and "ABBA", padded with unique string terminators, to become "ABAB$0", "BABA$1" and "ABBA$2". The nodes representing "A", "B", "AB" and "BA" all have descendant leaves from all of the strings, numbered 0, 1 and 2.

2.4. Cygwin

Cygwin is a UNIX-compatible environment that runs on Windows systems. It consists of cygwin1.dll, a library that takes POSIX calls and translates them into Win32 calls; a shell (GNU BASH, the shell used on most Linux systems, is the default); an implementation of the X Window System as well as GCC [8].

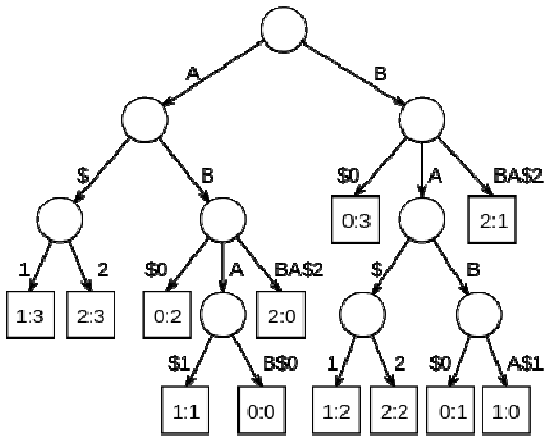

Figure 3. *Generalized Suffix Tree of a set of strings.*

3. Research Methodology

3.1. System Architecture

The proposed system architecture consists of following parts:

- Local Control Unit
  - Analysis Unit
  - Communication Unit
  - Database
- Known-Attack Filter
- Network Intrusion Prevention System
- Global Control Unit

Each unit is described as follow:

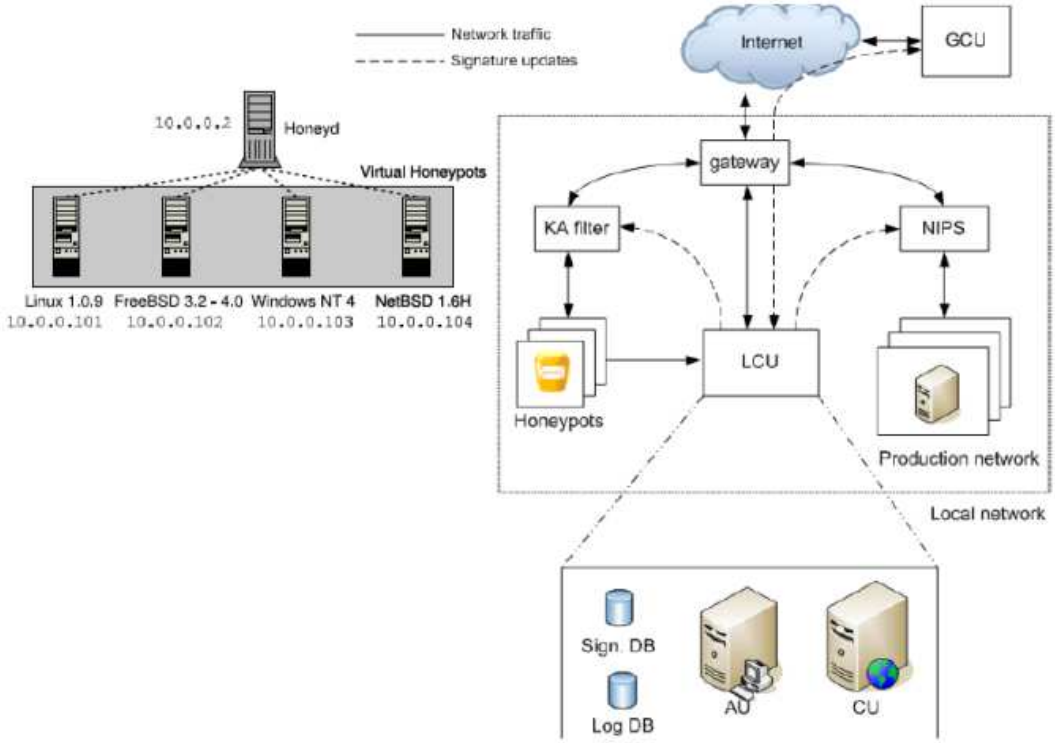

Figure 4. *Proposed System Architecture*

- *Local Control Unit*

This unit has a simplified version that is only able to receive signature updates from the Global Control Unit (GCU) and use these in NIPS to protect the production network. But in a complicated version, LCU consists of behind units:

- *Analysis Unit – (AU)*

The AU's main task is to correlate the incoming honeypot events and create signatures for possible worms. When receiving new events from a honeypot, the following procedure is executed:

- Step 1: The incoming events are stored in the log database and correlated with older events. If a similar chain of events has been received a certain number of times before, it is assumed that the events are caused by a worm and step 2 is carried out. If not, the events are simply stored and the AU returns to idle state.
- Step 2: The network packets causing the same chain of events are compared. If a common substring (larger than a given threshold) is found between these traffic traces, a signature is created.
- Step 3: Before storing the newly generated signature in the database, it is compared with the already existing ones. It can then either be stored directly in the database as a new entry or help to improve one of the older ones.

- *Communication Unit – (CU)*

The CU's main purpose is to exchange signatures with the Global Control Unit (GCU) as well as issuing signature updates to the Known-Attack (KA) filter and Network Intrusion Prevention System (NIPS).

- *Databases*

The signature database is used to store locally generated as well as received signatures. The log database is used to store the logged events along with relevant data.

- *Known-Attack Filter – (KA Filter)*

The main purpose of the KA filter is to look for known attacks (based on the signatures received from the LCU) in the traffic directed towards the honeypots.

- *Network Intrusion Prevention System – (NIPS)*

The NIPS is placed in the system to protect the production network. It can filter traffic that is unwanted based on certain ports as specified by the network administrator, as well as traffic that have been declared malicious as a result of signature updates from the LCU. Similar to the KA filter, it is also possible for the NIPS to report back to the LCU on the activity level of the received signatures.

- *Global Control Unit – (GCU)*

The GCU serves as a central signature storage and distribution unit. It receives signature updates from the

distributed LCUs and is able to correlate received data from different locations to compose improved signatures. Based on the received data, it issues periodic updates to the LCUs. As the GCU is a potential single point-of-failure and the effects can be catastrophic if it is compromised, the requirements regarding security are strict. All communication between the GCU and LCUs should be authenticated and encrypted in order to avoid forged signature updates.

3.2. Implementation

The implementation is based on traffic patterns, using Longest Common Substring (LCS) algorithm. Our system output is a file containing honeyd intrusion signatures in pseudo-snort format to filter unwanted production network traffic. Also the proposed architecture introduces the use of a Known-Attack (KA) filter. The main purpose of this filter is to remove known attacks from the traffic directed towards the honeypots. This filter reduces the amount of traffic needed to be processed by the honeypot sensors.

Signature generation system is implemented in Linux OS, but due to the common use of Windows OS, our implementation is also in Windows OS, using C programming language.

According to [5, 9, 10], for implementing the system we use below items:

1 Winpcap to capture packets
2 Cygwin compiler
3 Libevent library: The libevent API provides a mechanism to execute a callback function when a specific event occurs on a file descriptor or after a timeout has been reached [11].
4 Libdnet library: libdnet provides a simplified, portable interface to several low-level networking routines [12].
5 Libstree library: libstree is a generic suffix tree implementation, written in C [13].
6 Python

4. Results and Analysis

Signatures are periodically reported to an output module which implements the actual logging of the signature records. At the moment, there are modules that convert the signature records into pseudo-Snort format and a module that dumps the signature strings to a file. Our proposed system generated 53 signatures during a roughly 18-hour period and we captured 224 KB of traffic, comprising 560 TCP connections, 120 UDP connections and 24 ICMP pings. 25 signatures were created containing flow content strings. These are relatively long; on average they contain 136 bytes. The signatures format is as follow:

alert tcp 61.0.0.0/8 any → 129.241.196.0/24 80 (msg: "Hello!!! "; flags: PA+; flow: established; content: "GET http://lookfreebies.com/prx1.php HTTP/1.0|0D 0A|Accept: */*|0D 0A|Accept-Language: en-us|0D 0A|User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)|0D 0A|Host: lookfreebies.com|0D 0A|Connection: Keep-Alive|0D 0A 0D|";) As mentioned in [14], rule options are:

- The direction operator → indicates the orientation, or direction of the traffic that the rule applies to. The IP address and port numbers on the left side of the direction operator is considered to be the traffic coming from the source.
- The generated signature has TCP protocol that the source IP address is 61.0.0.0/8 and the destination IP address is 129.241.196.0/24 and its port is 80.
- The msg rule option tells the logging and alerting engine the message to print along with a packet dump or to an alert.
- The content keyword is one of the more important features of Snort. It allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.
- The flags keyword is used to check if specific TCP flag bits are present.

5. Conclusion

In Summary, an automated signature generation system for Windows OS designed and implemented. This system considered as honeyd plugin. Our tests show the proposed system is particularly good at generating attack signatures.

Acknowledgements

I would like to thank Mr. Erfan Khosravian for valuable feedbacks and Comments.

References

[1] Vusal Aliyev, “Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network”. Master of Science Thesis in the Master Degree Programme, Secure and Dependable Computer Systems, Department of Computer Science and Engineering Division of Computer Security. Goteborg, Sweden, 2010.

[2] Grønland, Vidar Ajaxon. "Building IDS rules by means of a honeypot". Master’s Thesis, Master of Science in Information Security, Department of Computer Science and Media Technology Gjøvik University College, 2006.

[3] Noordin, Yusuff, Mohamed. "HONEYPOTS REVEALED". IT Security Officer. Specialist Dip. Info Security, MA. Internet Security Mgmt.

[4] Mark Meijerink, Jonel Spellen. "Intrusion Detection System honeypots". Master Program System and Network Administration, University of Amsterdam, 2006.

[5] Baumann, Reto. "Honeyd – A low involvement Honeypot in Action". Originally published as part of the GCIA (GIAC Certified Intrusion Analyst) practical, 2003.

[6] Provos, Niels. "Honeyd- A Virtual Honeypot Daemon". Center for Information Technology Integration, University of Michigan. 2003.

[7] Sung, Wing-Kin; Melvin, Zhang Zhiyong. "Suffix Tree and Suffix Array". Knowledge Discovery and Data Mining Conference,2005.

[8] Moody, George. "An Introduction To Cygwin". Harvard-MIT Division of Health Sciences and Technology.

[9] Provos, Niels; Mathewson, Nick. "Libevent – an event notification library", 2011. URL: http://libevent.org/

[10] Van Rossum, Guido; "Introduction to Python". LinuxWorld, New York City, Documented in https://www.python.org/doc. 2002.

[11] Libevent – an event notification library: http://libevent.org.

[12] Libdnet: http://libdnet.sourceforge.net.

[13] Christian Kreibich; libstree: http://www.icir.org/christian/libstree .

[14] Roesch,Martin; Green, Chris. "SNORT Users Manual 2.8.5", The Snort Project (https://manual.snort.org), 2009.