

Attack Methodology Analysis By Using Honeypot Technology

Ms. Mandeep Kaur¹, Deepinderdeep Singh²

1. Department of Information & Technology, U.I.E.T., Panjab University, Chandigarh (India)

2. M.E. (Student), Department of Information & Technology, U.I.E.T, Panjab University, Chandigarh

E-Mail ID: deepinderdeep@yahoo.com

Abstract—A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. This paper presents the deployment of high interaction honeypot in a highly controlled network where every packet entering or leaving is monitored, captured, and analysed in order to know the attack methodology. A Honeynet is a network of high-interaction honeypots.

Honeynets have become an important tool for researchers and network operators. However, their effectiveness has been impeded by a lack of a standard unified honeynet data model which results from having multiple un-related data sources, each with its own access method and format.

In this paper we propose a new data collection Gen-III architecture that addresses the need for both rapid comprehension and detailed analysis by providing data access method: a relational model based fast path

I. INTRODUCTION

A Honeynet is a network of high interaction honeypots. High interaction honeypots are quite different from low interaction honeypots such as Honeyd for they provide a full operating system and set of software for an intruder to interact with. This high level of interactivity is a desired because it allows researchers the ability to observe the behaviour of an intruder in a live system, and not a simulation. As a result, high interaction honeypots are well suited to capture new or unanticipated activity. However, high interaction honeypots collect a larger volume detailed data from multiple data sources making it difficult to manage honeynets and make sense of the collected data.

To help facilitate honeynet deployments, Gen-III architecture is used. The major Objective of Gen-III are:

The data model should be directly based on this ontology and strive to be independent of data source. In the Gen-II model, adding a new data capture instrument results in a new data format, and every analysis tool needs to be modified to use the new data source. While it is unlikely that we can eliminate the need to modify tools to fully benefit from a new data source, we feel that we can limit the amount of work by keeping the data access and encoding methods consistent. The need for interoperability between functionally equivalent tools provides another situation where having a source independent standard data model is necessary.

The data model should be capable of expressing the relationships between the different types of information in the

model. As an example, a network flow is going to be related to a host and process, the model should make it easy to identify that relationship. If we were monitoring system call activity on a honeypot we should be able to relate a network flow to a system call made by a process on one of the hosts. This type of monitoring is key to programmatic identification of incident causality.

The data should be added to the composite data store on a continual basis as close to real-time as practical. This allows the system to pre-process the data into the data store as the data comes in rather than trying to do it at the moment an analyst asks for data, thus reducing the work required from multiple examinations of an event.

Lastly, it is necessary that a programmatically consistent data access methods is provided. In order to allow researchers to automatically share data between Honeynets, it is not sufficient to simply standardize how we collect data, but also the methods of representing and sharing data.

II. IMPLEMENTATION

The implementation described is the basis for a Honeynet data capture and analysis system under development Gen-III generation Honeywall. For this paper Gen-III generation data capture and data analysis capabilities are implemented to know the attack methodology of attacker, the heart of which are at the core of this paper.

A. Data Collection

Key to make use of the proposed model but absent from the GenII design are multiple data types which are not explicitly collected within the GenII architecture. This section addresses how we collect each of these new types of data and why they are important to create a contiguous composite view of intrusion sequence.

Within the GenII architecture IPTables is used to approximate network flow monitoring with its connection tracking capability. However, while providing some aspects of the flow concept, this capability does not provide valuable details such as quantity of data transferred, bi-directionality information or end time of a network connection. Such information is desired to estimate the interest of a connection, for example one might be more interested in a bidirectional flow lasting 10 seconds than a flow lasting under a second and consisting of a single

packet.

To improve the collection of flow data, we turned to Argus, a tool designed to provide flow monitoring as defined by the IP Performance Measurement Working Group. Argus provides network flow records which contain summary detail not provided by IPTables connection tracking, such as the start and end of the flow, number of bytes transmitted in each direction and number of packets transmitted in each direction. Similar in design to snort, Argus collects the canonical network traffic data using libpcap, and it then processes this data to create a derived data source. This data source is used to populate flow related fields in the relational model. Unlike the Connection Tracking logs, this data does describe the quantity of data and duration of each flow.

To augment our understanding of the network activity and the hosts at either side of a communication, we added passive operating system fingerprinting capability, provided by the pOf tool. POf is also a pcap based monitor that provides an estimate of the operating system(OS) used by host that initiates a TCP connection. This data is useful for two reasons. First, across flows it allows one to see if the apparent host OS is changing for a given IP source providing an indication that the host might be behind a NAT. Second, OS identification can improve the accuracy of IDS events through the process of passive alert verification.

The addition of the Argus and pOf data to the Snort and packet capture data provides a more comprehensive representation of events than provided in the GenII design. Further this new data can be organized around the concept of a network flow. However additional data sources are needed to bridge the relational gap between the network flows and processes on a host.

To bridge this gap we enhanced Sebek to monitor network activity from the host's perspective. Sebek is a kernel based data capture tool designed to be installed on high interaction honeypots. Balas modified Sebek to monitor socket, process and file activity. These modifications provided three necessary capabilities.

First, Sebek was enhanced to monitor socket activity. Whenever a honeypot accepts or creates a network connection, Sebeck records the IP level attributes as well as the corresponding host, process and inode. This allows us to relate a network flow to the specific open inode and file descriptor used by a process to service the connection. This data is integral to providing a composite view of the incident that transcends flow and host data. Once a network connection associated with an intrusion attempt is observed, we immediately know which inode and process the intrusion was tied to. Using this data we can quickly identify related information such as the keystrokes captured by Sebek. Second, Sebek was enhanced to monitor process creation. This monitoring allows us to relate one process to another, rebuilding the process tree. This is important in intrusion analysis for it allows us to track the intrusion forward from the point of intrusion identifying all processes created, and any other causally related system activity, such as outbound network connections. The same capability can be used in

reverse, if we see an outbound connection on a honeypot, we can back track to identify the point of intrusion.

Lastly, the ability to monitor the opening of files was added. Coupled with the process tree this allows us to identify all files accessed as part of an intrusion. This knowledge can in turn be used to prioritize data analysis efforts.

B. Data Fusion

Hflow was developed to combine each of these data sources into a composite relational model. It continually consumes data from each source, fusing it based on identifiable relationships and it then loads this data into a database.

Hflow receives Argus flow, Snort IDS, pOf OS fingerprints and Sebek data. This data once combined is then inserted into a database.

Flow related data, such as Argus and Snort, are correlated based on corresponding tuples consisting of the IP protocol number, the source and destination IP addresses and if applicable port numbers which fall within the same time period. This is similar in approach to the way an operating system determines which socket a packet is related to.

Process data is gathered by Sebek and organized based on the process ID. For every activity monitored by Sebek both the process ID and parent process ID are recorded. This combination allows Hflow to recover the process tree. One concern with the process ID is that it is a counter that can roll over. While the process ID value eventually rolls over, during the same period in time no two processes on a single host will share a process ID. Further, in the unlikely event that 2 processes share the same process id in close time proximity, it is even more unlikely that the 2 processes will have the same combination of process id and parent process id.

File data is linked to process data by monitoring system calls that provides us with the process ID of the acting process, and the inode number of the file being acted on. By monitoring the open and read system calls, we are able to map process activity to a specific file.

The result of Hflow data fusion is a unified data set with identifiable relationships between the logical categories. This data is exported in the necessary format to be uploaded into a relational database in a continual basis as seen in Fig:1.

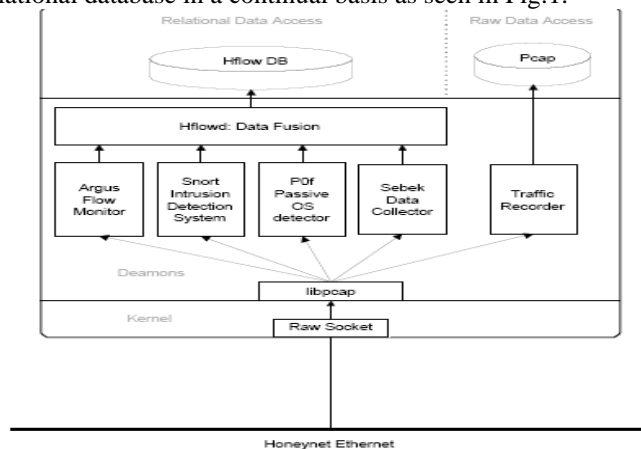


Fig: 1 Data Collection and Fusion

C. Data Analysis

To demonstrate the utility of the fast path and slow path data model, Walleye, a web based Honeynet data analysis interface. The purpose of the interface is not to be a comprehensive or monolithic analysis platform. It is designed to facilitate intrusion sequence comprehension through the presentation of a sequence-centric, composite view of the data. By this we mean that the view of the data we provide is a composite of each of the raw data sources. If successful, what the analyst perceives is the greater than any with any single source of data. It is hoped that the composite view provided by Walleye, which spans multiple data source will improve the analyst's capability to quickly perceive the intrusion sequence. This allows the analyst to look not just at an IDS event, but look at that event in the context of the effects of the event on system activity and side effects. This context exists today but requires manual effort to identify. By using the relational model we can automatically identify this context and provide it to the analyst, ideally improving accuracy and efficiency, by removing the need to manually identify relations and compiling the composite event view.

The honeypot used in research consisted of a single honeypot and honeywall differently. The honeywall was implemented using a minimal installation of Fedora Core 2 Linux as a base, with the kernel recompiled to support filtering with iptables in bridging mode. Data controls and logging were implemented on this machine using Generation III techniques described above. With bridging implemented on the two network interfaces, network-facing and honeypot-facing, and no IP address being claimed, the honeywall is practically invisible to the honeypot, aside from the actions the honeywall may take to intercede in attempts to attack non-honeypot systems.

The honeypot was implemented as an installation with VMWare with two Operating Systems, one guest OS has Red Hat Linux version, 9.0, without any security updates and other has Windows 2000 Advanced Server. It was hypothesized that such a machine would give attackers plenty of opportunity for compromising it, while still running an operating system new enough to be found "in the wild", making it a likely target for scanning. The popular Red Hat and Microsoft distribution was chosen since most publicly available exploits that contain offsets for exploiting various operating systems usually include offsets for versions of Red Hat and Microsoft. Table I contains a list of the services that were left open on the Linux honeypot. Table II contains a list of services that were left open on the Windows honeypot. A large number of services were left running to improve the chances that it would be compromised. A number of user accounts were added to the system with weak passwords, and files were placed on the honeypot's filesystem to give the appearance that the system was intended to be a class project's server.

III. RESULT AND DISCUSSION

The honeypot was online, unfiltered by firewall for 45 days

over a period of 1.5 months (taken off-line during times that there would not be anyone around to monitor it and during times that it was being analyzed). During this time a large number of probes and attacks were observed. A summary of these attempts is listed in Table III.

Port	State	Services
21/tcp	Open	FTP
22/tcp	Open	SSH
23/tcp	Open	Telnet
80/tcp	Open	HTTP
111/tcp	Filtered	SUN-RPC
135/tcp	Open	Msrpc
3306/tcp	Open	Mysql

Services Running On Linux Honeygot
TABLE-I

Port	State	Services
21/tcp	Open	FTP
23/tcp	Open	Telnet
25/tcp	Open	SMTP
42/tcp	Open	Nameserver
53/tcp	Open	Domain
80/tcp	Open	HTTP
135/tcp	Filtered	Msrpc
3306/tcp	Open	Mysql

Services Running On Window - Honeygot
TABLE II

Figure 2 shows the current representation of a flow. From the image we can observe several features: (i) This is a bidirectional TCP flow, whose initiator is XX.XX.XX.227, (ii) 1 alert generated i.e. "WEB-IIS view source via translate header", (iii) 6 packets were sent to destination IP and 5 packets were sent in response to attacking IP during this flow and (iv) the OS fingerprint is Linux for this connection. This chart combines information from four different data sources: Argus, POF, Snort and Sebek presents them to the analyst in a related and aggregated fashion.

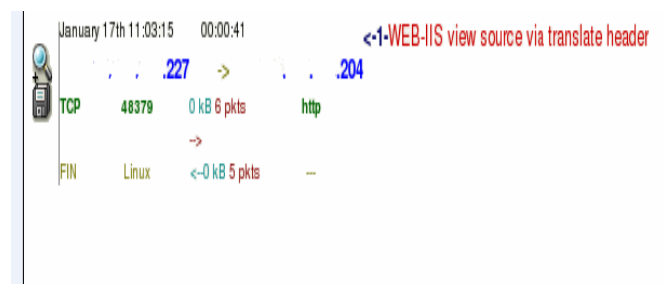


Fig: 2 Flow Abstraction used by Walleye

Protocol	Port	Alert Generated	Probes
FTP	21	Vsftpd Login Attempt	94
SSH	22	Sshd Authentication	515
HTTP	80	-Cross-site Scripting(XSS) Attack	110
		-HTTP response Splitting Attack	10
		-Remote File Access Attempt	18
		-System Command Injection	45
		-Session Fixation	6
		-Proxy Access Attempt	20
		-WEB-IIS view source via translate header	141
		-Other	112
msrpc	135	ICMP Ping Attempt	657
netbios-ssn	139	NETBIOS SMB IPC unicode share access	38
snmp	161	SNMP request tcp	24
snmptrap	162	SNMP trap tcp	18
Microsoft- ds	445	NETBIOS SMB-DS IPC unicode share access	27
ms-sql-m	1434	MS-SQL version overflow attempt	40

Probes observed on Honeypots

TABLE-III

REFERENCES

- [1] T. H. Project, Know Your Enemy. Addison-Wesley, 2nd ed.,2004.
- [2] N. Provos, "Citi technical report 03-1: A virtual honeypot framework," tech. rep., Center for Information Technology Integration, University of Michigan, 2003.
- [3] T. H. Project, "Know your enemy:genii honeynets."
- [4] "Netfilter homepage." <http://www.netfilter.org>, 2005.
- [5] M. Roesch, "Snort-lightweight intrusion detection for networks," in Proceedings of LISA'99 Systems Administration Conference, 1999.
- [6] T. H. Project, "Know your enemy sebek." <http://project.honeynet.org/papers/sebek.pdf>, November 2003. Last access: Feb 2005.
- [7] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen, "Enriching intrusion alerts through multi-host causality," in Proceedings of the 2005 Network and Distributed System Security Symposium (NDSS), February 2005.
- [8] T. H. Project, "Know your enemy:Honeywall cdrom." <http://project.honeynet.org/papers/cdrom/index.html>, Feb 2005. Last access: Feb 2005.
- [9] "Argus project," 2004.

[10] "Rtfm: New attributes for tra c flow measurement," 1999.

[11] "Framework for ip performance metrics," 1999.

[12] M. Zalewski, "passive os fingerprinting tool." <http://lcamtuf.coredump.cx/p0f.shtml>, 2004.

[13]"Real-time network awareness." <http://www.sourcefire.com/products/downloads/secured/sfRNA.pdf>, 2004.

[14] R. Gula, "Correlating ids alerts with vulnerability information." <http://www.tenablesecurity.com/images/pdfs/va-ids.pdf>, 2002.

[15] M, "Cve-2002-0392." <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0392>, 2002.

[16] E. Balas, "Honeynet data analysis: A technique for correlating sebek and network data," in Digital Forensics Research Workshop, June 2004.

[17] L. Spitzner, "Honeytokens: The other honeypot." <http://www.securityfocus.com/infocus/1713>, Jul 2003. Last access: Feb 2005.

[18] P. Biondi, "Shellforge g2: Shellcodes for everybody and every platform." http://www.cartel-securite.fr/pbiondi/conf/shellforgeG2_csw04.pdf, 2004.

[19] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," Journal of Computer Security, vol. 6, no. 3, pp. 151-180, 1998.

[20] N. Provos, "Improving host security with system call policies," in 12th USENIX Security Symposium, USENIX, 2003.

