Security and Results of a Large-Scale High-Interaction Honeypot

J. Briffaut, J.-F. Lalande, C. Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
Ensi de Bourges
88 bd Lahitolle, 18020 Bourges Cedex, France
{jeremy.briffaut, jean-francois.lalande, christian.toinard}@ensi-bourges.fr

***Abstract*— This paper[1] presents the design and discusses the results of a secured high-interaction honeypot. The challenge is to have a honeypot that welcomes attackers, allows userland malicious activities but prevents system corruption. The honeypot must authorize real malicious activities. It must ease the analysis of those activities. A clustered honeypot is proposed for two kinds of hosts. The first class prevents a system corruption and never has to be reinstalled. The second class assumes a system corruption but an easy reinstallation is available. Various off-the-shelf security tools are deployed to detect a corruption and to ease analysis. Moreover, host and network information enable a full analysis for complex scenario of attacks. The solution is totally based on open source software and has been validated over two years. A complete analysis is provided using the collected events and alarms. First, different types of malicious activities are easily reconstructed. Second, correlation of alarms enables us to compare the efficiency of various off-the-shelf security tools. Third, a correlation eases a complete analysis for the host and network activities. Finally, complete examples of attacks are explained. Ongoing works focus on recognition of complex malicious activities using a correlation grid and on distributed analysis.**

***Index Terms*— High-Interaction Honeypot, Attack Monitoring, Intrusion Detection System**

This paper presents a framework for securing high-interaction honeypots, an analysis of the malicious activities resulting from one year and a half of experimentation, an evaluation of the efficiency of the security mechanisms and complete examples of complex attacks. Deploying high-interaction honeypot is a challenging research activity and few works address this problem. High-interaction honeypots use real operating systems and services while low-interaction honeypots simulate services and thus limit the malicious activities. The difficulty of using high-interaction honeypot is that hackers can gain a complete control of the system. A high-interaction honeypot is more efficient because it authorizes real malicious activities. Thus, the analysis of the results is closer to real systems. Since a high-interaction honeypot presents more vulnerabilities, they need frequent reinstallations and advanced monitoring of the activities.

This paper presents a clustered approach to provide a high-interaction honeypot. Then, it gives a comprehensive analysis of the attacks that occurred during almost two years of experimentation. Section I presents the state of the art related to honeypot. A motivation follows in order 1) to introduce our work and 2) to show mainly that our approach allows intruders to execute more attacks without suspecting that they are in a honeypot.

Section III describes the proposed honeypot architecture. A first cluster offers two types of real vulnerable hosts, 1) secured Operating Systems that let the attackers use the target system but avoid the compromising of the system and 2) unsecured Operating Systems with strong monitoring and reinstallation facilities. Monitoring of activities is available at various host levels (i.e. from system call level to shell level) but also at various network levels. A second cluster allows to collect, store and analyze the malicious activities. A third cluster provides a correlation framework in order to rebuild the sessions of malicious activities. Reuse of the events, coming from the different levels, enables a reconstruction of the full session using both host and network information. The section IV describes briefly how the security of the targeted hosts is achieved.

However, simply securing the honeypot does not give much valuable knowledge. Section V provides a further analysis about the attacks themselves using a study of the intrusions carried out during during one year and a half of experimentation. Our study enables to learn more about 1) the vulnerabilities of current systems and 2) the efficiency offered by off-the-shelf security mechanisms. That study produces a better knowledge of real malicious activities. Thus, the resulting analysis gives a good justification of our high-interaction honeypot. Some examples of attacks are given in section VI.

Section VII concludes this paper and gives perspectives for the automation of the malicious activities analysis.

I. State of the art

Two types of honeypots are distinguished in the literature. The low level honeypots emulate a limited part of the services, mainly the network protocols. They allow to compute statistical results and to model the attack activities [1]. On the other hand, the high-interaction honeypots deploy real operating systems that allow to capture more complete information. Some high-interaction honeypots use VMware to emulate the hosts [2] but hiding a hypervisor is very hard to achieve. Finally, the attackers

[1]This work is supported by the French National Research Agency through funds of the Security Challenge Program.

may give up their attacks because the operating system will probably be reinstalled. For the papers that focus on the low interaction honeypots, or on the exploit of services [3], the problem is simpler than ours since there are less security risks.

In different white papers about honeypots [4], [5], the different generations of honeypots and their architectures are described. Our proposed honeypot is part of the second generation of honeypots described by [4]. Several toolkits are presented as the well-known Honeyd [6] which is used by Leurre.com, a distributed honeypot. These white papers cover the network configuration, host sensors and the modus operandi for collecting data at the date of year 2003.

The problem of configuring the honeypots is difficult because one have to find a trade-off between a real production host and an adaptive honeypot that let the attackers enter the system. If the honeypots are too easy to enter, the attacker could guess that the host is not a production host. Moreover, one cannot get any results if the honeypot is hidden or too hard to attack. A large variety of papers try to propose new kinds of honeypots that solve parts of this problem.

Paper [7] presents how to configure honeypots dynamically based on network scans. It improves the initial work of Hieb and Graham to adapt the honeypot behavior (the opened ports) to the detected attacks. In [8] the authors tries to build a honeypot that can be plugged into a local network and that find automatically unused IP. These works allow a system administrator to deploy a honeypot according to his network configuration and to let the honeypot evolves and adapts the services with the requests of the attackers. These goals are crucial when deploying some honeypots on a large scale network with thousands of hosts but the choice of the operating system and the security of this operating system remains an opened question that we want to address in this paper.

Some papers address how to detect that a host is a honeypot [9], [10]. Those solutions are more or less related to a kernel analysis that allows to detect a User-mode Linux or a VMware host. This is precisely these papers that show that deploying real operating systems with real services is better for capturing attackers, but is possibly more dangerous for other hosts on the network.

In [11], the authors propose an advanced hybrid type of honeypots: shadow honeypots. They analyze the network traffic in order to redirect suspicious packets to a shadow honeypot. This shadow honeypot is a replication of the protected software that can be used to analyze the received attacks. Those solutions are complex to deploy. Our purpose is to collect more information by letting intruders attack directly the host.

A honeypot can also help to defend against attackers. Collected data can drive countermeasures dynamically using the frontal servers of the considered organization. This has been demonstrated in [12] with spammers that are detected, then slowed by the system and eventually completely blocked. The honeypot can also be hidden in a real production system and can provide legal information when performing forensic operations on compromised hosts [13].

II. Motivation

The main objective is to prevent a high-interaction honeypot from being frequently reinstalled. The second objective is to efficiently monitor the malicious activities and to maintain a cluster of operating systems connected to public Internet addresses.

Usually low-interaction honeypots do not authorize the attacker to gain a login shell on the real system. In low-interaction honeypot all the services are emulated and even the login shell is emulated. The main drawbacks of these low-interaction honeypots are:

- The attacker can discover that he is connected to a fake system;
- The services are partially emulated;
- The vulnerabilities of these services are missing, but efforts are made to address this issue [14];
- Host based attacks are impossible to capture.

A high interaction honeypot is required to capture host based attacks and to offer the vulnerabilities of the target system. As each operating system, distribution, service, software can contain different vulnerabilities according to their version, the deployment of a large heterogeneous cluster of honeypots is required to increase the number of possible attacks.

As attackers have a direct access to a real system and can exploit the vulnerabilities of the target system, they can gain administrator privileges and compromise easily the system. The main problems related to high interaction honeypots are:

- An attacker can exploit a vulnerability in order to obtain administration privileges and stop the monitoring mechanisms;
- The operating system can be stopped or broken;
- The attacker can use the honeypot to attack other hosts on the Internet;
- A large number of operating systems have to be deployed and monitored to offer a large amount of vulnerabilities;
- The malicious activities generate a large amount of traces and analysis becomes difficult;
- The volume of data, that has to be stored, is even larger when auditing system calls and when using a great number of complementary sensors.

The easiest solution to prevent the preceding problems is to reinstall frequently the operating system. First, it is not feasible for a large scale honeypot. Second, attacks are lost and monitoring is corrupted. Finally, the decision of reinstalling the system requires a complex analysis. Moreover, an attacker can discover that the operating system has been cleaned between two connections and has the means to understand that he is connected to a honeypot. The proposed solution addresses the problems related to high-interaction honeypots while 1) minimizing

a system corruption, 2) detecting a system corruption and 3) providing automation for the reinstallation of the corrupted systems.

Existing studies focus on low-interaction honeypots and network analysis. Our study goes one step further since it supports high-interaction honeypots and provides a full analysis of a complete attack i.e. including both host and network information. It gives a comprehensive analysis of 1) malicious activities and 2) efficiency of the security mechanisms. Since complete analysis is supported, full examples of sessions can be given using a correlation of the events and alarms coming from our honeypot. This kind of analysis is required to improve the knowledge of malicious activities and to test the efficiency provided by off-the-shelf security mechanisms.

III. High Interaction Honeypots

Figure 1 describes the global architecture of the clustered high interaction honeypots. That architecture is separated into four parts: an Internet access management, the clustered honeypots, a cluster for auditing and storage, and finally a correlation grid. This section details each part of the proposed architecture, the objectives, the installed software and their configuration. Intentionally, the section puts focus on technical aspects to make possible reproducing this clustered architecture.

A. Network management

The honeypot hosts are directly connected on the Internet. Each honeypot has a public IP address. All the connections from Internet to the honeypots are forwarded through a Honeywall host. The goals of the Honeywall are the following ones:

- It limits the bandwidth usage from the inside to the outside to 10MB/hour;
- It limits the number of TCP connection from the inside to the outside to 100 connections per hour;
- It guarantees a bandwidth limit of 100MB/hour from outside to inside.

The Honeywall has two network interfaces in bridge mode (no IP, only forwarding of packets). This way, it cannot be seen by intruders. Iptable rules limit the bandwidth and the connections in order to control the capabilities of the successful attacks:

- Denial of Service attacks (DOS) from the inside to the outside;
- Port scans from the inside to the outside;
- Denial of Service attacks (DOS) from the outside to the inside.

There is no firewall blocking some ports or services on the Honeywall host. Thus, the intruders see the honeypots as frontal hosts on the Internet. Those honeypots can be used to attack from the inside other hosts on Internet but with very limited resources.

In order to analyze the network traffic outgoing from the Honeywall, a hub forwards the packets to a network analyzer. That host includes a network Intrusion System (Snort IDS) that analyzes the traffic and sends the alerts to a local OSSIM agent. It keeps also a TCP dump (PCAP file) of all the incoming and outgoing network traffic. Traces and OSSIM alerts are sent to the second cluster for auditing and storage. Two other network IDS are installed in order to detect the operating system type of the attacker (P0F) and the services (PADS). The analysis is passive and thus invisible from the intruder side. The generated alerts are sent to the OSSIM agent.

The network analysis and the Honeywall are separated because an attacker can possibly exploit a vulnerability against one of the network IDS. This way, the network analyzer can be compromised or disabled. In this case, as from one of the honeypots, it can be used as a starting point to attack outside hosts. Nevertheless, the limitation guaranteed by the Honeywall limits drastically the ammount of attacks against internet hosts.

In the network management part of our architecture, control of the malicious network traffic is supported and monitoring is available, using network IDS, that provides alerts and event data for the correlation phase.

B. Clustered honeypots

The second part of our architecture is the honeypot hosts. Three kinds of hosts are represented in figure 1:

- hosts with Mandatory Access Control enforcing the operating systems (MAC) [15];
- "Classical" hosts without MAC but Discretionary Access Control systems (DAC) [16];
- Microsoft Windows operating systems.

The MAC and DAC systems are deployed on clustered hosts with five GNU/Linux distributions: debian, gentoo, fedora and ubuntu. The Microsoft Windows systems are: NT 4.0, 2000, 2003, XP. Each host has two network interfaces, one with a public IP address and one with a local address 172.30.3.x. This way, all the honeypot hosts are connected to a local network that enables an intruder, when connected on one of these hosts, to attack other honeypots. No firewall is installed on the 14 honeypots.

Each GNU/Linux host has an OPENSSH modified service that facilitates the open of sessions for the intruders: when an attacker attempts to log on one of the honeypots with a ssh brute force scan, our openssh service creates randomly accounts with a probability of 1% using the attempted login/password. It gives the intruder a real account on the system with a regulat home directory. The created account is persistent and authorizes the intruder to come later with this persistent account. Thus, he can easily continue his attack.

In order to capture the activities of the intruders, several IDS hosts and security tools are deployed on the GNU/Linux and Windows systems. These IDS monitor four types of information sources: the system activities (system calls, processes), the integrity of the file systems, the kernel and daemon logs, and the bash sessions. Those complementary information are required to carry out a correlation of host processes and network traffic. On each host, installed IDS and security tools are as

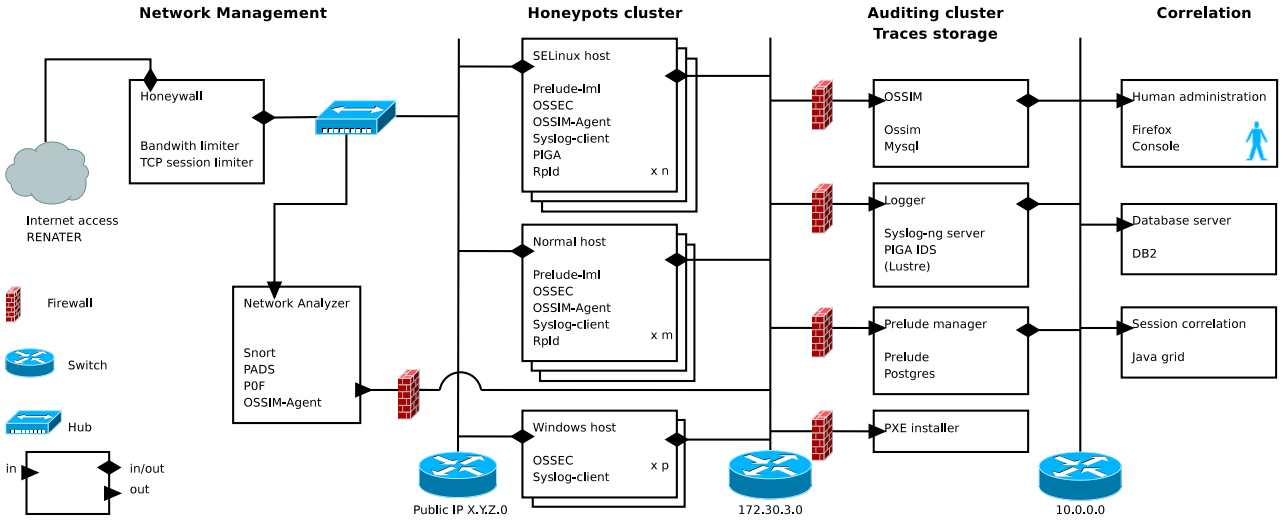

Figure 1. Clustered high interaction honeypots architecture

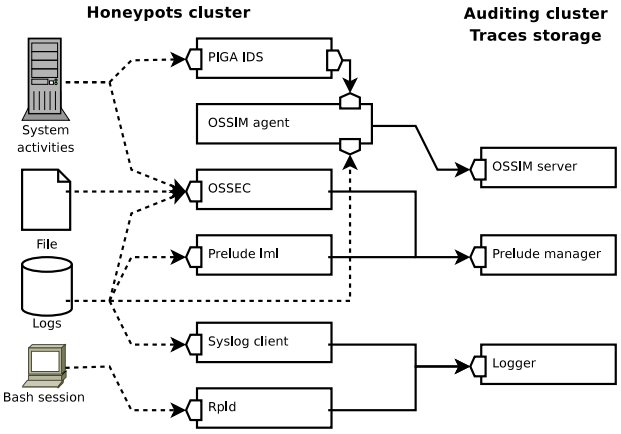

Figure 2. IDS and tools monitoring host activities

follows: 1) Prelude-lml, a system log analyzer that reports system activities (connections, daemon logs like apache, ...), 2) OSSEC for checking the integrity of the system files, rootkit installation, registry and logs modification, 3) PIGA, a policy based IDS that detects violation of security properties [17], 4) OSSIM that collects the alarms generated by the PIGA IDS and analyzes the system logs, 5) Syslog that forwards all kernel and system logs to the Logger host and 6) Rpld that captures the shell activities of the attackers.

Figure 2 sums up the data used by each security agent (dotted lines). All the collected alarms, logs and sessions are sent to the next part of our architecture i.e. the cluster for the storage of traces.

C. Storage of traces

The third part of our architecture stores and analyzes the collected alarms and logs. Storage hosts are are connected to the 172.30.3.0 network. Those hosts are protected by firewalls allowing only the incoming traffic. The opened ports are those used by the OSSIM, Prelude and syslog servers (loggers).

Three analysis frameworks are used in order to collect all events and reports alerts readable by humans: 1) OSSIM: it provides a framework to manage security information. It generates reports, aggregates alerts, send incident tickets, ... It stores the collected data into a mysql server (eventually a clustered mysql server). 2) Prelude: it aggregates the collected information and enables to visualize them on a website. All events are stored using the IDMEF standard. A postegreSQL server is used (eventually a clustered postegreSQL server). 3) Syslog (logger): it stores all the syslog traces of the honeypots on a distributed file system (LUSTRE).

All network and system events/alarms are stored and can be visualized by an administrator. As the three frameworks do not use the same standards, we need these three servers for the correlation phase. Another goal of these servers is to prevent an attacker from deleting the log/traces/activities generated on the honeypots.

D. Correlation

This last part of our architecture enables to visualize the alarms and to test correlation algorithms between all kinds of IDS alarms. Indeed, the main goal is to characterize attacks using network IDS alarms, host IDS alarms, system events logged in traces. These algorithms use our OSSIM, Prelude and syslog databases through the private network 10.0.0.0. Event standardization is provided using a common format for the storage into the various databases. The correlation algorithms, that are outside of the scope of that paper, use a Java grid to improve the computation time.

IV. SECURITY OF HONEYPOTS

The main drawback of our architecture is to provide real systems, thus allowing an attacker to get administrator privileges. In this section we detail MAC and DAC hosts.

A. MAC hosts

With a MAC host, a policy guarantees that the root (staff role) user does not have the privileges of the super administrator (admin role). If an attacker exploits a vulnerability and becomes root, the policy limits his privileges on the system as a normal user. The only way of becoming super administrator is to exploit a kernel vulnerability or to attack the MAC mechanism (SELinux). In this case, the system is compromised and have to be reinstalled using the PXE server. Nevertheless, during one year and a half of deployment, we never observed such an attack.

The main advantage of these hosts is that they are time persistent. An attacker can come back later in order to finish an attack whereas a reinstallation will suggest that the host is a honeypot. Moreover attackers can explore the different home directories of other attackers and possibly reuse/delete/download the uploaded scripts. All these shell activities are logged by the Rpld server.

B. DAC hosts and modifications

This classical system can easily be compromised by an attacker. When a user gets administrator privileges, the host have to be reinstalled with the PXE server. The drawback of these honeypots is that they need more administrative tasks. Alerts have to be daily monitored and an administrator decides if the system is "too" compromised. We implemented a cron job that computes the differences that appear during the time. When a honeypot file differs from the corresponding file stored on the PXE, an alert is sent by OSSEC integrity analyzer and the difference is stored on the PXE SERVER for further manual analysis.

C. Auto-installation

The PXE server contains boottp and a tftp server in order to deploy fresh images of our distributions. It contains 2 linux hosts (Debian, Gentoo) with SELinux MAC mechanism, 2 DAC linux hosts (Debian, Gentoo) and 4 Microsoft Windows systems, NT 4.0, 2000, 2003, XP. This server is used to reinstall a compromised honeypot using the PXE protocol on the 172.30.3.0 network.

V. STATISTICAL RESULTS

The presented statistics aggregate activities from February 27th 2007 to August 23th 2008 for 2 MAC honeypots (Debian, Gentoo) and 2 DAC honeypots (Debian, Gentoo) and 1 windows (NT 2000).

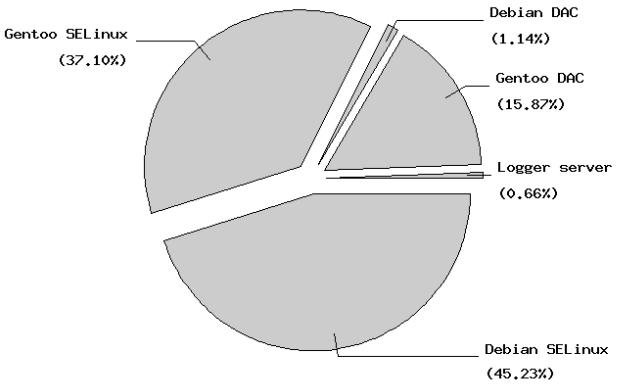

Figure 3. Alarms per host

A. Per host results

Figure 3 represents the distribution of alerts per host. There are two reasons that explain the low number of alerts on the DAC host. First, machines got public IP adresses starting from September 2007: before that date, attacks have been achieved against DAC hosts from our private network and thus Snort did not report anything for DAC hosts during those six first months. Second, PIGA IDS cannot be deployed without a MAC mechanism, reducing the number of alerts. Note also that the logger host has also been attacked even if this host has no public IP address and is protected by a firewall.

There is no real difference between a Debian or Gentoo distribution because attackers do not target one distribution. It can be explained because a large part of the reported alarms are networks alarms that do not target a specific host.

B. Statistics for IDS tools

In the database server 9,879,363 events and 388,467 alarms have been stored i.e. 760 events per hour and 29 alarms per hour. An event is an information like a user connection and an alarm (e.g. a malformed packet) is a malicious event i.e. usually a subset of a complete attack. Numerous alarms are monitored, but the major part is generated by Snort and corresponds with false positives.

Figure 4 shows the distribution of the network and host IDS alarms. Snort reports a large amount of false positives. Indeed, it uses a signature base and detects attacks in packets using pattern matching: it cannot know if the attack succeeded or may succeed. This is the main drawback of network IDS. False positives are eliminated using PIGA IDS that detects only 55,860 opened sessions by scan robots and only 3,116 sessions performing activities on the corresponding host.

Table I reports the main types of alarms and the sensors that detect it. The most frequent alarm deals with the ssh daemon and is reported by Prelude-lml: it is the first step to enter our honeypots. For this purpose, they try brute passwords via ssh and generate a lot of alarms.

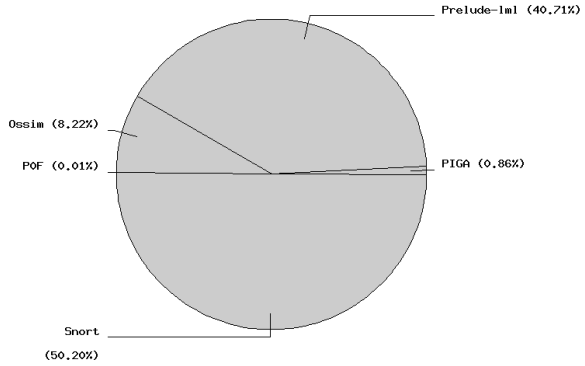

Figure 4. Alarms per sensor

Sensor	Description	Occurences
Prelude-lml	SSHd: Root login refused	498,468
Snort	Destination udp port not reachable	452,011
Prelude-lml	SSHd: Bad password	109.221
OSSIM	SSHd: Possible brute force tentative	53,444
Prelude-lml	SSHd: Invalid user	43,311
PIGA	Integrity: system file modification	41,063
Prelude-lml	FTP bad login	21,366
Snort	Potential outbound SSH scan	19,983
PIGA	Confidentiality: information flow	16,191
	...	
Snort	Attempt to access an non-existent file.	1.324
Snort	Common web attack.	406
Snort	Mutiple web server 400 error codes from same source ip.	361
Snort	Multiple attempts to access non-existent files (web scan)	251
Snort	XSS (Cross Site Scripting) attempt.	162
Snort	Alarm for signature k	< 1,000
Snort	Alarm for signature k+1	< 1,000
	...	

TABLE I.
MAIN TYPES OF ALARMS

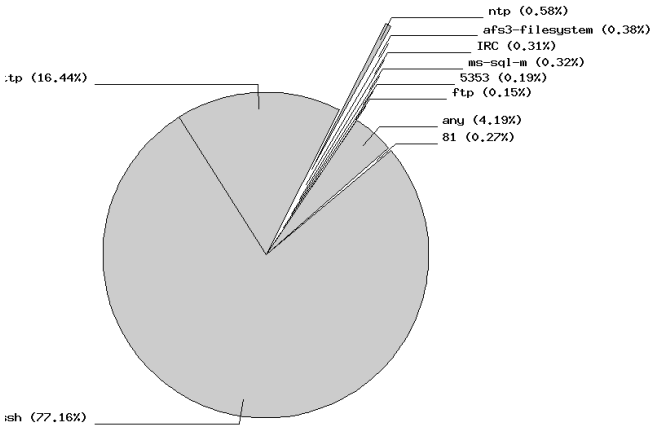

Figure 5. Alarms per port

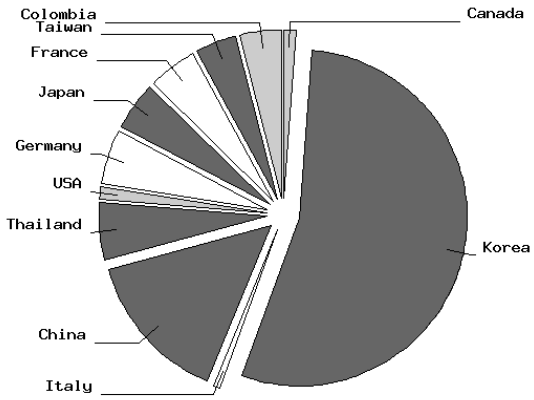

Figure 6. Alarms per country - incoming

Note that ftp accounts are also targeted by these brute force attacks. For outgoing traffic, Snort reports 19,983 ssh scans because the intruders use gained accounts to attack outside hosts.

PIGA IDS detected the modifications of the configuration files that a normal user should not access. It detected also information flows mainly from those configuration files to the user space. Moreover, information provided by specific files like /etc/shadow or /etc/apache/httpd.conf have been stolen by the attackers.

The total amount of Snort alarms is 50,20%. It includes 642,008 UDP port scans plus a lot of alarms with lower rates associated to different IP addresses.

Figure 5 is consistent with table I: the ssh port dominates the number of alarms. The http port is mainly attacked from the outside in order to install phishing websites. 10% of attacks are malicious ICMP packets. Classical ports are used in order to exploit classical windows vulnerabilities (afs3 filesystem, ms-sql-m): these attacks are worms trying to propagate themselves. The IRC port is used by IRC bots installed on the honeypots in order to establish outside connections with IRC channels. Those bots can be controlled by attacker's orders using those specific channels.

C. Results per country

Figures 6 and 7 give the ratio of reported alarms per country. Incoming and outgoing alarms are separated: the first ones are alarms generated when the intruders penetrate our honeypots. The second ones are alarms generated when the intruders use our honeypots as a base to launch attacks against outside or local hosts.

The world region that launches most of the attacks is Asia (grey). The other attackers are mainly from Europe (white). Note that one cannot find any attack coming from the United States whereas outgoing attacks are more than 40% against the USA. There is no guaranty that the attackers are really incoming from the claimed country: they may use proxies that should be compromised hosts in order to have a large bandwidth for performing island hopping attacks.

On figure 7 we added the attacks against our local hosts. The ratio between attacks against external host or local host is biased because of the Honeywall that limits outgoing bandwidth and connections. The most attacked

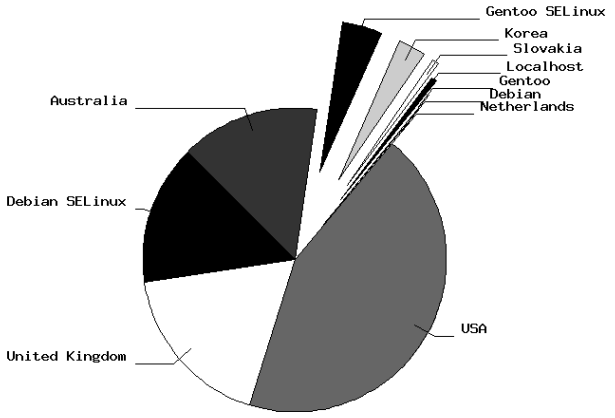

Figure 7. Alarms per country - outgoing

Occurrences in /home	Occurrences in /tmp	Type of malwares
186	188	ELF 32-bit LSB executable dynamically linked
140	98	ASCII C program text
34	44	shell script
17	11	ASCII Pascal program text
7	21	ELF 32-bit LSB executable statically linked linked
3	0	Mach-O executable ppc
2	13	C++ program text
Occurrences in /home	**Occurrences in /tmp**	**Type of data**
181	578	ASCII text
64	70	data
10	25	gzip/tar compressed data
7	1	HTML document text
1	0	ISO 9660 CD-ROM filesystem

TABLE II.
MAIN TYPES OF MALWARES

Occurences	Bash commands
539	ls
531	cd
136	wget
50-100	./start tar cat ps rm w
20-40	mv ./a passwd mkdir chmod exit nano uname
10-20	pico ./scan ./exploit-2.6 kill ./v who ./x id perl info ftp ./unix history ping
5-10	curl export ./init ./mass ./muh screen unzip ./net php /sbin/ifconfig uptime help quit vi netstat ./s ssh

TABLE III.
MAIN EXECUTED COMMANDS BY ATTACKERS

local host is the Debian SELinux host mainly because it has a public IP address, suggesting that attackers do not try to discover hosts that do not have an Internet address. They prefer to attack a second host that have an Internet address which is the Gentoo SELinux host.

D. Time results

Figure 8 shows the evolution of the number of events (solid lines) and alarms (dotted lines) during one year and a half of experiments. We can observe a hole of events during summer because of a large power supply failure of 3 weeks. In December 2007, Prelude-lml was added to our architecture and 4 new hosts increased the number of events and alarms. When omitting the failure problem and the arrival of Prelude-lml, the number of alarms is constant, confirming thus that attackers are using random IP addresses on Internet.

We also computed the number of alarms per hour and we observed that the variations of events/alarms are not related to specific hours. It varies randomly from 230,000 to 750,000 events and from 9,000 to 20,000 alarms when considering a specific hour. As intruders use robots in order to attack the honeypots, we cannot observe any privileged hours.

E. Malware and activities results

Table II is an analysis of the malwares installed by attackers for a period of time of 6 months. It computes the number of time a type of file is found in */home* and */tmp*. Attackers mainly try i386 binaries (probably C programs) that are dynamically linked, which is surprising. They also try C programs that are recompiled on the compromised hosts. Some exotic attacks use Pascal or C++ programs, and some shell scripts can perform some automatic operations for launching/compiling C programs. Some extra files are uploaded, mainly documentation for the malwares.

The Table III gives a statistical overview of the typed commands logged in *.bash_history* files by all the attackers on the period of 6 months. The most used commands suggest that most of the time, real humans are using the shell script to explore the entered accounts (IT) and browse the home-directory or the created directories (*cd*). Then, the highest activity is to download a malware using wget (136 occurrences). The commands that occur between 50 and 100 times suggest that the attackers unpack the downloaded malware (*tar*) or inspect the system (*ps, cat*). Then, if possible, the attacker tries to prepare an attack, which is suggested by the occurrences 20 to 40 (passwd, chmod, nano, uname). The remaining commands are less clear but some non standard commands suggest that the attacker launches the downloaded malware to compromise the system, install a bot, or scan the network (*./scan, ./exploit-2.6, ./v, ./x, ./unix, ./mass, ./muh, ./net, ./s*).

Table IV gives the file extensions of the downloaded malwares. Most of the downloaded files are archives (*tgz, gz, tar*) which mainly contain the C program files to be compiled on the targeted host. Some multimedia files (*jpg, mp3*) suggest that some exploits will be tried or that the host will be used as a storage server.

F. Discussion

Figure 9 shows the number of malicious activities seen by each sensor. Snort detects the brute force attacks (more than 850,000) but cannot decide if sessions are

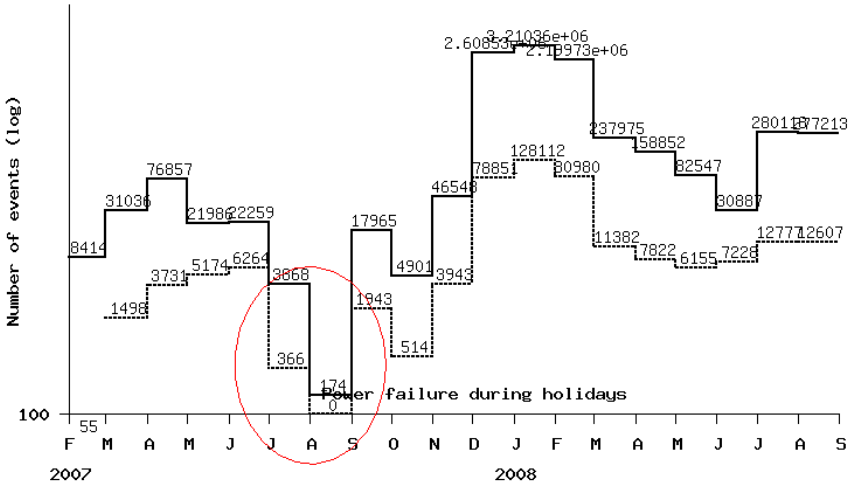

Figure 8. Number of events during a year and a half (logarithmic)

Occurrences	Downloaded file extensions
45	.tgz
24	.gz
15	.tar
10	.jpg
5	.php
4	.mp3
3	.pl
1	.txt

TABLE IV.
EXTENSIONS OF THE DOWNLOADED MALWARES

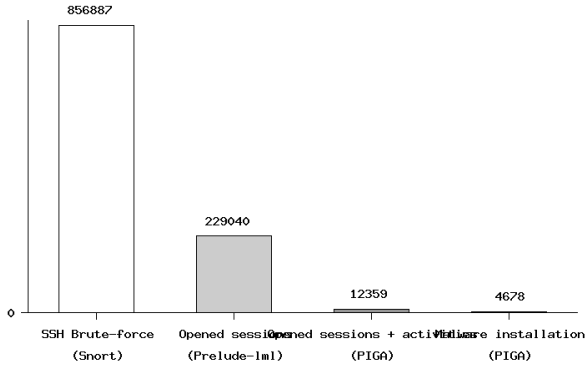

Figure 9. Number of malicious activities seen by sensors

opened. Prelude-lml reports that 230,000 sessions have been opened. In theory the system offers 1 session over 100 brute force attacks. In practice, 229,000 sessions have been detected instead of the 85,000 theoretical sessions because the attackers use similar dictionaries and the tried logins have probably already been created. Moreover, the attackers can come back later with the created accounts which increases the number of sessions seen by Prelude.

The third sensor, PIGA IDS, detects opened sessions with real activities such as shell commands for exploring the host, downloading softwares and executing binaries or scripts. The number of activities (12,000) shows that 95% of the sessions are never exploited by the intruder. Indeed, a bruteforce of ssh creates several accounts (because of the 1% creation policy). A successful intruder will only use one of the created accounts. PIGA reports the number of malware installations. For this purpose, PIGA factorizes download, installation, execution of a software as a single global alarm. We conclude that 37% of the sessions deals with malwares such as IRC bots or ssh scanners.

These results show the complementarity of those sensors. Snort enables to know the different intrusion attempts but cannot decide if the attempts is successful. Snort cannot analyze further the attempt and complementary tools are needed. Moreover, for ssh connections, a network IDS is useless to analyze the use of the connection. Though, other host IDS are required to monitor the next steps of the attack. In order to do an efficient detection, system calls must be monitored. In this direction PIGA IDS proved his efficiency to detect complex scenarios of attacks.

VI. ATTACK EXAMPLES

This section presents examples for complete sessions of attacks. The proposed analysis reuses the result of our correlation mechanism. The proposed examples show the relationships between the host/network activities and the generated events/alarms.

A. Inspection and exploit attempts

The first example presents the attempt of an attacker to acquire more privileges by exploiting system vulnerabilities. The following listing shows the attacker bash commands during this session. First, the attacker build a ssh brute-force in order to obtain a user access (line 1-2). Next, the attacker analyzes the system and his privileges (line 3-5). Then, he downloads a set of exploits hidden in a file with a *.jpg* extension (more difficult to detect by an

IDS) (line 6-9). Finally, several exploits are tried in order to compromise the system and the attacker verifies if each exploit succeeds (line 10-15).

```
(ssh bruteforce)                                  1
(ssh connection)
cat /proc/cpuinfo                                 3
who
ps -x                                             5
wget casperel.3x.ro/cas.jpg
tar zxvf cas.jpg                                  7
rm -rf cas.jpg
cd boti                                           9
./init
uname -a                                          11
w
./kswapd                                          13
./entity-gen
ps -aux                                           15
```

During this malicious session, several events and alarms are generated by the sensors. The table V summarizes, for each line of the previous listing, the number of occurrences, the name, the source (the name of the sensor), the type (event, alarm or correlation) and the source level (which part of our honeypot architecture is involved) of each information.

When the attacker tries to brute-force the ssh service, he generates a lot of bad connections (bad username or bad password). The `ossim-agent`, that analyzes the authentication log (`/var/log/auth.log`) on the honeypot host (level 2), generates an event for each bad connection with the name `SSHd: Failed username or password`. In this case, the attacker tried 48 usernames/passwords before opening a valid session. Moreover, the `OSSIM` server, at level 3, correlates these events as a `Possible SSH brute force login attempt against` this host. When the user session is opened, the `ossim-agent` generates an event (`pam_unix: authentication successful`), and `PIGA` detects a forbidden transition to the user context : `Transition Violation (sshd_t->user_t)`. These two events are correlated as a global alarm: `User connection Network Scan` at the level 4.

When the attacker is connected, he obtains some information from the host with system enumeration commands (line 3-5). Each command generates a forbidden information flow from the System level to the User level. These flows are detected by `PIGA` and they correspond to the event `Information flow (System->User)`.

Next, the attacker downloads an archive containing a set of exploits for the targeted Operating System (line 6) corresponding to a correlation between a `PIGA` event (`Information flow (Network->User)`) and a `Snort` event (`Wget : activity from REMOTE_IP to local host`). He extracts and deletes this archive and generates a violation of the integrity of the user domain (line 7, 8).

Finally, the attacker runs some exploits from this archive (line 10, 13, 14). Each execution corresponds to a violation of the modification and execution privileges (`Duty Separation violation (Write -> Execute)`). Indeed, in order to protect the honeypot host, a user cannot modify (write) and then execute a file. This rule corresponds to the the *duty separation* security property. When this rule is violated, `PIGA` denies the final access (execution) and generates an alarm. This kind of property allows to protect the Operating System and to prevent system level corruption.

The previous alarms or events are generated at level 2 or 3 by different security mechanisms. All these events and alarms allow to generate a correlation alarm corresponding to the entire session of the attacker. This alarm corresponds to a `User exploit attempts` and needs the apparition of the three information : *User connection*, *Information flow (Network->User)*, *Duty Separation violation*.

B. IRC bot and Outgoing Network Scan

The second example shows a typical install of an IRC bot by an attacker. The commands are probably automatically launched when the ssh succeeds. The *./start* command is a script that launches the `EnergyMech IRC Bot` software.

In addition to the previous example, the attacker comes in a second time to recover the result of a ssh scan launched from the local host. In this case, for the line 10, `PIGA` generates an alarm corresponding to an `Information flow (User->Network)`. Finally, the four information *User connection*, *Information flow (Network->User)*, *Duty Separation violation* and *Information flow (User->Network)* are correlated to the correlation alarm : `User Outgoing Network Scan`.

```
(ssh bruteforce)                                  1
(ssh connection)
wget members.lycos.co.uk/??/linuxx.tar.gz         3
tar zxvf linuxx.tar.gz
rm -rf linuxx.tar.gz                              5
cd .al/
./start                                           7
(quit)
(ssh connection)                                  9
scp ./al/scanresult test@w.x.y.z:~/
```

VII. Conclusion and perspectives

During about two years of experiments, MAC honeypots never needed reinstallation despite the detection of 229,000 sessions and 12,359 malicious activities. This result shows the robustness of the proposed architecture. The DAC hosts have been reinstalled only three times. A good monitoring was proposed for both MAC and DAC systems. Automation of reinstallation has been completed. Our study shows the needing of complementary network and host IDS. Some tools generate a large number of false positive whereas other tools authorize a more precise analysis of the intrusions. In order to make DAC hosts as much persistent as possible, correlation between the different collected data is required to take the reinstallation decision.

Line	Nb.	Name	Source	Type	Level
1	48	SSHd: Failed username or password	ossim-agent (syslog)	Event	2
1	1	directive_event: Possible SSH brute force login attempt	Ossim	Correlation	3
2	1	SSHd: Login sucessful, Accepted publickey	ossim-agent (syslog)	Event	2
2	1	pam_unix: authentication successful	ossim-agent (syslog)	Event	2
2	1	Transition Violation (sshd_t->user_t)	PIGA	Event	2
2	1	User connection	(PIGA,OSSIM)	Correlation	4
3-5	2	Information flow (System->User)	PIGA	Event	2
6	1	Information flow (Network->User)	PIGA	Event	2
6	1	Wget : activity from REMOTE_IP to local host	Snort	Event	2
6	1	Network Information flow	(PIGA,Snort)	Correlation	2
7-8	2	Integrity violation (User Home)	PIGA	Event	2
10,13,14	3	Duty Separation violation (Write -> Execute)	PIGA	Alarm	2
11,12,15	3	Information flow (System->User)	PIGA	Event	2
	1	User exploit tentatives (User connection, Information flow, Duty Separation violation)	Session Correlation	Correlation alarm	4

TABLE V.
ALARMS AND EVENTS RESULTS

An automatic reconstruction of malicious sessions uses host and network information to analyze complex attacks. It eases the complete analysis of complex attacks and provides a good knowledge about those malicious activities. Moreover, the correlation between host and network information provides a better knowledge about the false positive and false negative alerts that are generated by the different off-the-shelf security tools. Thus, a precise comparison between the various security tools is proposed. Finally, complete examples of attacks are presented using the session reconstruction and the different generated alerts.

Ongoing works will compute a risk level using correlation results. A higher level of analysis is required. For this purpose, automatic recognition of scenarios of attacks is required. Moreover, the major difficulties are related to the correlation between network events in order to detect and analyze distributed attacks.

REFERENCES

[1] M. Kaaniche, Y. Deswarte, E. Alata, M. Dacier, and V. Nicomette, "Empirical analysis and statistical modeling of attack processes based on honeypots," in *Workshop on Empirical Evaluation of Dependability and Security (WEEDS)*, Philadelphia, USA, June 2006, pp. 119–124.

[2] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," in *6th European Dependable Computing Conference*. France: IEEE Computer Society, 10 2006, pp. 39–44.

[3] P. Diebold, A. Hess, and G. Schäfer, "A honeypot architecture for detecting and analyzing unknown network attacks," in *14th Kommunikation in Verteilten Systemen 2005*, Kaiserslautern, Germany, February 2005.

[4] H. Q. Nguyen, F. Pouget, and M. Dacier, "White paper: Integration of honeypot data into an alert correlation engine," Institut Eurecom, France, Tech. Rep., Jun 2005.

[5] R. Baumann and C. Plattner, "White paper: Honeypots," February 2002. [Online]. Available: http://security.rbaumann.net/download/whitepaper.pdf

[6] N. Provos, "A virtual honeypot framework," in *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2004.

[7] C. Hecker, K. L. Nance, and B. Hay, "Dynamic honeypot construction," in *10th Colloquium for Information Systems Security Education*, University of Maryland, USA, June 2006.

[8] I. Kuwatly, M. Sraj, Z. A. Masri, and H. Artail, "A dynamic honeypot design for intrusion detection," in *ICPS '04: Proceedings of the The IEEE/ACS International Conference on Pervasive Services (ICPS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 95–104.

[9] T. Holz and F. Raynal, "Detecting honeypots and other suspicious environments," in *Information Assurance Workshop*, University of Maryland, USA, June 2005, pp. 29–36.

[10] S. Innes and C. Valli, "Honeypots: How do you know when you are inside one?" in *4th Australian Digital Forensics Conference*, C. Valli and A. Woodward, Eds. Perth, Western Australia: School of Computer and Information Science, Edith Cowan University, 2005.

[11] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. X. andE. Markatos, and A. D. Keromytis, "Detecting targeted attacks using shadow honeypots," in *14th USENIX Security Symposium*, Baltimore, MD, August 2005, pp. 129–144.

[12] L. Oudot, "Fighting spammers with honeypots," November 2003.

[13] Y. Manzano and A. Yasinsac, "Honeytraps, a valuable tool to provide effective countermeasures for crime against computer and network systems," in *7th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, July 2003.

[14] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, *The Nepenthes Platform: An Efficient Approach to Collect Malware*. Springer Berlin / Heidelberg, 2006, vol. 4219/2006.

[15] D. E. Bell and L. J. La Padula, "Secure computer systems: Mathematical foundations and model," The MITRE Corporation, Bedford, MA, Technical Report M74-244, May 1973.

[16] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[17] M. Blanc, J. Briffaut, J.-F. Lalande, and C. Toinard, "Distributed control enabling consistent mac policies and ids based on a meta-policy approach," in *Workshop on Policies for Distributed Systems and Networks*. Canada London: IEEE Computer Society, 2006.