A proposal for securing a large-scale high-interaction honeypot

J. Briffaut, J.-F. Lalande, C. Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
Université d'Orléans, rue Léonard de Vinci
45067 Orléans, France
{jeremy.briffaut,jean-francois.lalande,christian.toinard}@ensi-bourges.fr

KEYWORDS

High-Interaction Honeypot, Attack Monitoring, IDS

ABSTRACT

This paper presents the design of a secured high-interaction honeypot. The challenge is to have a honeypot that welcomes attackers, allows userland malicious activities but prevents from system corruption. The honeypot must be scalable to authorize a large amount of malicious activities and to analyze those activities efficiently. The hardening of the honeypot is proposed for two kinds of host. The first class prevents system corruption and has never to be reinstalled. The second class assumes system corruptions but easy reinstallation is available. A first cluster enables to deploy a wide range of honeypots and security sensors. A second cluster provides an efficient auditing facility. The solution is totally based on open source software and has been validated during one year. A statistical analysis shows the efficiency of the different sensors. Origin and destination of attacks are given. Moreover, the complementarities of the sensors are discussed. Ongoing works focus on recognition of complex malicious activities using a correlation grid.

INTRODUCTION

This paper proposes an architecture for securing high-interaction honeypots. The main objective is to welcome attackers and to provide different operating systems. The difficulty of high-interaction honeypot is that hackers can gain a complete control of the system. So, the risk is high that attackers use the honeypot to carry out severe attacks. Deploying high-interaction honeypot is a challenging research activity and few works address this problem with real operating systems and services. Classical approaches use low interaction honeypot but they limit the malicious activities. That is why high interaction is required. Currently, they need frequent reinstallations and advanced monitoring of the activities. The main objective is to prevent high-interaction honeypot from frequent reinstallation. 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; 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: 1) An attacker can exploit a vulnerability in order to obtain administration privileges and stop the monitoring mechanisms; 2) The operating system can be stopped or broken; 3) The attacker can use the honeypot to attack other hosts on the Internet; 4) A large number of operating systems have to be deployed and monitored to offer a large amount of vulnerabilities; 5) The malicious activities generate a large amount of traces and analysis become difficult; 6) 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 external analysis and is complex. Moreover, an attacker can discover that the operating system has been cleaned between two connections and possibly can understand that he is connected to a honeypot.

In this paper, a clustered honeypot is proposed that offers vulnerable systems accessible by public Internet addresses. Our high interaction honeypot provides two types of hosts, 1) secure hosts that let the attackers use the target system but avoid the compromising of the sys-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

tem and 2) classical systems with strong monitoring and reinstallation facilities. A second cluster allows collecting and analyzing the activities. Monitoring is available at various host levels (i.e. from system call level to shell level) but also at various network levels. The cluster is safe since it minimizes the possibility of using the honeypot to attack outside hosts.

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. It allows to compute statistical results and to model the attack activities (Kaaniche et al., 2006). On the other hands, the high-interaction honeypots deploy real operating systems that allow capturing more complete information. Some high-interaction honeypots use VMware to emulate the host (Alata et al., 2006) but hiding the hypervisor is very hard to achieve: attackers 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 (Diebold et al., 2005), the problem is simpler than ours since there is less security aspects addressed.

In different white papers about honeypots (Nguyen et al., 2005; Baumann and Plattner, 2002), the different generations of honeypots and their architectures are described. Our proposed honeypot is part of the second generation of honeypots described by (Nguyen et al., 2005). Several toolkits are presented as the well known Honeyd (Provos, 2004) 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. On the other way, no results can be obtained 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 (Hecker et al., 2006) 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 (Kuwatly et al., 2004) 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 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 with this paper.

Some papers address the techniques to detect that a host is a honeypot (Holz and Raynal, 2005; Innes and Valli, 2005). These techniques are more or less related to kernel analysis that allows detecting 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 (Anagnostakis et al., 2005), 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. These types of solution are complex to deploy and our purpose is to collect more information by letting intruders attack directly the host.

HIGH INTERACTION HONEYPOTS

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

Network management

The honeypot hosts are directly connected on the Internet. Each honeypot has a public IP. All the connections from Internet to the honeypots are forwarded through a Honeywall host. The goals of the Honeywall are the following: 1) It limits the bandwidth usage from the inside to the outside to 10MB/hour; 2) It limits the number of TCP connection from the inside to the outside to 100 connections per hour; 3) It limits the bandwidth usage from the outside to the inside to 100MB/hour.

The Honeywall has two network interfaces in bridge mode (no IP, just forwarding packets) and this way cannot be seen by intruders. Limitation of bandwidth and connections is achieved by iptables rules in order to limit the impact of these possible attacks: 1) Denial of Service attacks (DOS) from the inside to the outside; 2) Port scans from the inside to the outside; 3) 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 replicates the packets to a network analyzer host. This host contains a network Intrusion Detection System (Snort IDS) that analyzes the traffic and

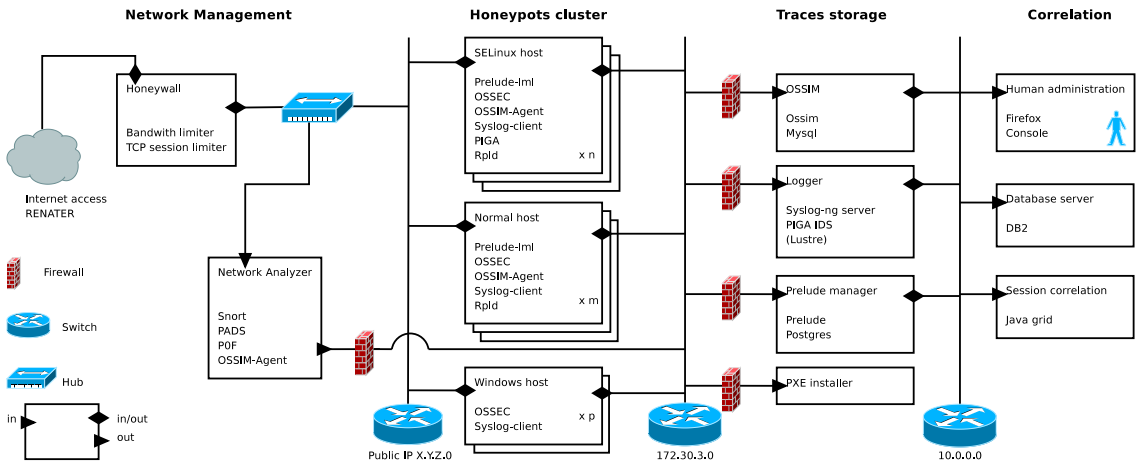

Figure 1: Clustered high interaction honeypots architecture

possibly send alerts to the 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 but cannot be used as starting point to attack outside hosts because of the limitation guaranteed by the Honeywall.

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

Clustered honeypots

The second part of our architecture is the honeypot hosts. Three kinds of honeypots are represented in figure 1: 1) Mandatory Access Control enforced the security of the operating systems (MAC) (Bell and La Padula, 1973); 2) "Classical" hosts without MAC but Discretionary Access Control systems (DAC) (Harrison et al., 1976); 3) Microsoft Windows operating system.

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 an 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 home directory. The created account is persistent and authorizes the intruder to come later with this login/password and to continue his attack.

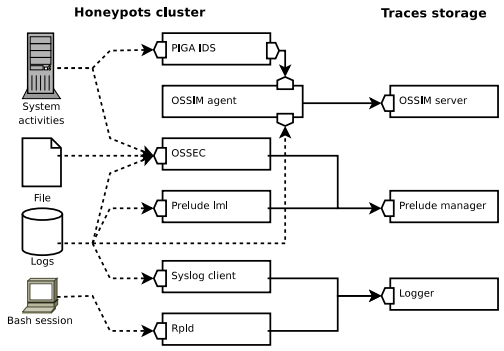

Figure 2: IDS and tools monitoring host activities

In order to capture the activities of the intruders, several host IDS and 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 logs of kernel and daemons, and the bash sessions. All these information are complementary and enable correlation between network and host data. On each host, the installed IDS and tools are the followings: 1) Prelude-lml is a system log analyzer that reports system activities (connections, daemon logs like apache, ...). 2) OSSEC checks the integrity of the system files, rootkit installation and modification of the registry and logs. 3) OSSIM agent collects the alarms generated by PIGA IDS and analyzes system logs. 4) PIGA IDS is a policy based IDS that de-

tect violation of security properties (Blanc et al., 2006). 5) Syslog client forwards all kernel and system logs to the Logger host. 6) Rpld captures the shell activities performed by the attackers and allows to replay them.

On figure 2 we sum up the data used by each IDS (dotted lines). All the collected alarms, logs, sessions are sent to the next part of our architecture, the traces storage clusters.

Traces storage

The hosts of the third part of the proposed architecture are used to store and analyze the alarms and logs captured on honeypots. The hosts are connected on the 172.30.3.0 network and protected by firewalls allowing only incoming traffic. The opened ports are those used by the OSSIM server, Prelude server, and syslog server (logger).

Three analysis frameworks are used in order to collect all events and generate reporting 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 (possibly a clustered mysql). 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 (possible a clustered postegreSQL). 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 needs this three servers for the correlation phase. Another goal of these servers is to prevent an attacker to delete the log/-traces/activities generated on honeypots as the reported events are transmitted in real time (it needs the time for the daemon to send the information to the server).

Correlation

This last part of our architecture is used 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 send request to the database of OSSIM, Prelude and syslog using the private network 10.0.0.0. As the format of the events is not standard, we merge them in a neutral format into the database server used by our correlation algorithms. These algorithms, not described in this paper, are written in java and have high CPU and memory consumption. These algorithms cannot be done in real time and use a java grid to distribute the computation.

SECURITY OF HONEYPOTS

The main drawback of our architecture is to provide real systems with vulnerabilities that possibly allow an attacker to get administrator privileges. In this section we details MAC and DAC hosts.

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 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 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.

DAC hosts and modifications

This classical system can easily be compromised by an attacker. When an 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 a decision has to be taken to know when 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.

Auto-installation

The PXE server contains boottp and a tftp server in order to deploy fresh images of our distributions. It contains Linux images, Debian, Gentoo, Fedora and Ubuntu with and without SELinux MAC mechanism and 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.

STATISTICAL RESULTS

The presented statistics aggregate activities from February 27th 2007 to February 21th 2008 for 2 MAC honeypots (Debian, Gentoo) and 2 DAC honeypots (Debian, Gentoo) and 1 windows (NT 2000).

Per host results

Figure 3 represents the distribution of alerts per host. There are two reasons that explains the low number of alerts on the DAC host. First, we give public IP addresses in September 2007: attacks on these hosts are achieved on the private network before this date and Snort did not report anything during the six first months. Second, PIGA IDS cannot be deployed without a MAC mechanism, reducing the number of alerts. Note also that the logger

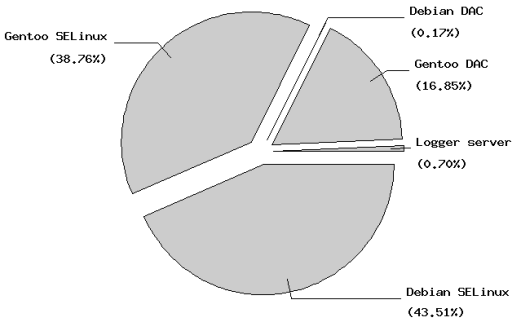

Figure 3: Alarms per host

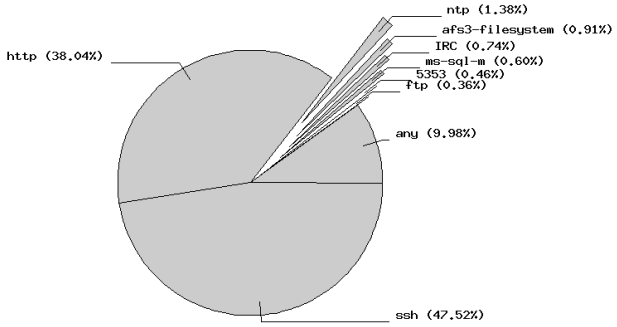

Figure 5: Alarms per port

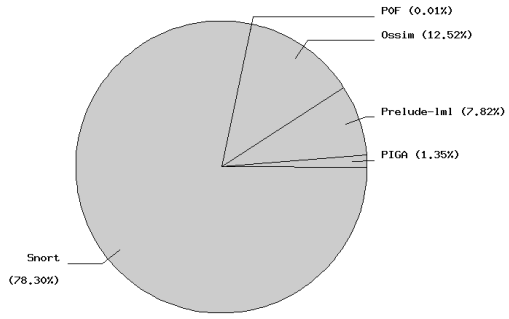

Figure 4: Alarms per sensor

Sensor	Description	Ocurences
Prelude-lml	SSHd: Root login refused	498,468
Snort	Destination udp port not reachable	452,011
Prelude-lml	SSHd: Bad password	49,329
OSSIM	SSHd: Possible brute force tentative	43,989
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	Alarm for signature k	< 1,000
Snort	...	< 1,000
Snort	Alarm for signature k+1	< 1,000
	...	

Table 1: Main types of alarms

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 does not target a specific host.

Statistics for IDS tools

In the database server 8,206,382 events and 302,543 alarms are stored i.e. 950 events per hour and 35 alarms per hour. An event is an information (as a user connection) and an alarm is a malicious event possibly a part of an attack (as a malformed packet). It is a lot of alarms to monitor, but the major part of these alarms is generated by Snort and is mainly 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 45,590 opened sessions by scan robots and only 2,219 sessions performing activities on the corresponding host.

Table 1 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. Intruders use ssh scan tools and try to brute force passwords that generates a lot of alarms. Note that ftp accounts are also targeted by these scan 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 78%. It includes 452,011 UDP port scans plus a lot of alarms with lower rate associated to different IP addresses.

The figure 5 is consistent with table 1: the ssh port dominates the number of alarms. The http port is mainly attacked from the outside in order to install fishing 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 be connected on outside IRC channels. The bots can be controlled by attacker's orders on the channel and could be used to launch DOS attacks.

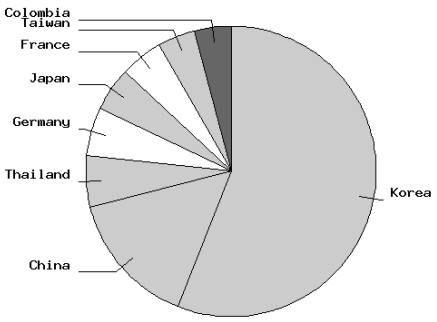

Figure 6: Alarms per country - incoming

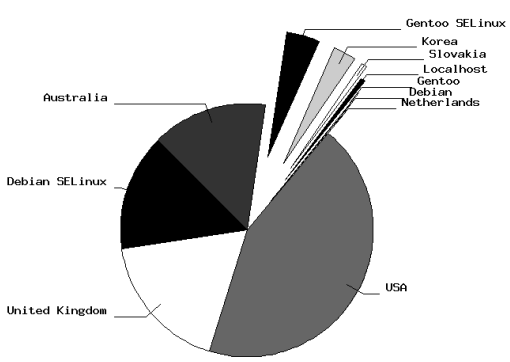

Figure 7: Alarms per country - outgoing

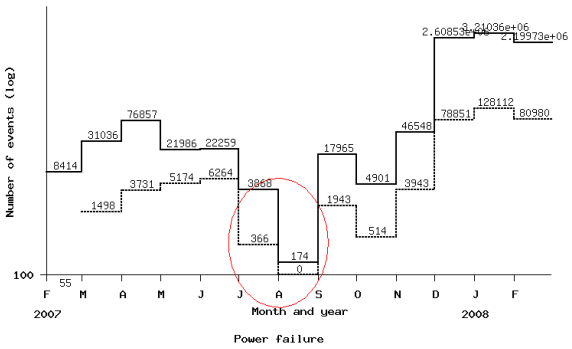

Figure 8: Number of events during a year (logarithmic)

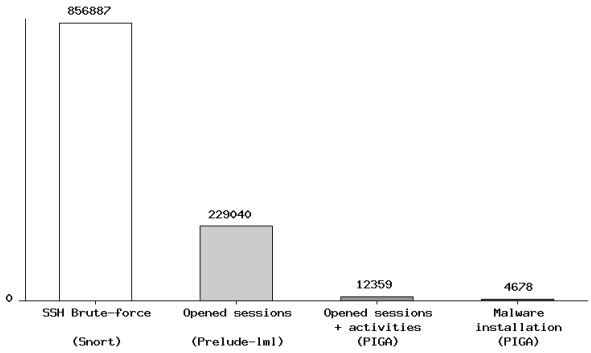

Figure 9: Number of malicious activities seen by sensors

Results per country

Figures 6 and 7 give the ratio for the total 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 no attack is incoming from the United States whereas the target of outgoing attacks on figure 7 is more than 40% against the USA.

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 local host is the Debian SELinux host mainly because it has a public IP address (and not the Debian host): it suggest that attackers does not try to discover local hosts that have not an Internet address; they prefer to attack a second host that have an Internet address which is the Gentoo SELinux host.

Time results

Figure 8 shows the evolution of the number of events (solid lines) and alarms (dotted lines) during one year of experiments. We can observe a hole of events during summer because of a large power supply failure of 3 weeks. In December 2007 we added Prelude-lml to our architecture and 4 new hosts which increased the number of events and alarms. If we omit the failure problem and the arrival of Prelude-lml the number of alarms is constant with time which confirms 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 scripts in order to attack the honeypots, we cannot observe specific activities on some precise hours.

Discussion

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

ated login/passwords which increase the number of sessions seen by Prelude.

The third sensor, PIGA IDS, detects opened session with real activities such as shell commands for exploring the host, download of software, execution of binaries or scripts. The number of activities (12,000) shows that 95% of 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 installation. For this purpose, PIGA factorizes a download, installation, execution of a software as a malware 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 it efficiently, system calls must be monitored. In this direction PIGA IDS proved his efficiency to detect complex scenarios of attacks.

CONCLUSION AND PERSPECTIVES

During one year of experiments, MAC honeypots never needed reinstallation despite of the detection of 229,000 and 12,359 malicious activities. This result shows the robustness of the proposed architecture. The DAC hosts have been reinstalled only three times in three months. A good monitoring was proposed to both MAC and DAC system. Automation of reinstallation have been completed. Our study shows the need of complementary network and host IDS. Some tools generate a large number of false positive whereas other tools are more precise in the analysis of intrusions.

The decision of the reinstallation of DAC hosts was taken manually. In order to make DAC hosts as much persistent as possible, correlation between the different collected data is required to take the reinstallation decision. Ongoing works will compute a risk level using correlation results. Moreover, the data of a compromised system have to be manually analyzed using the modification files such as described in section . Future works will study how to associate those modifications to a known malware. This way unknown malware could be highlighted.

The next step of this work is to correlate events and alarms between all sensors in order to model complete session of attacks. This work has been done for the system events of a MAC host. The major difficulties are related to the correlation with network events and with distributed attacks.

REFERENCES

Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., and Herrb, M. (2006). Lessons learned from the deployment of a high-interaction honeypot. In *6th European Dependable Computing Conference*, pages 39–44, France. IEEE Computer Society.

Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., andE. Markatos, K. X., and Keromytis, A. D. (2005). Detecting targeted attacks using shadow honeypots. In *14th USENIX Security Symposium*, pages 129–144, Baltimore, MD.

Baumann, R. and Plattner, C. (2002). White paper: Honeypots.

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

Blanc, M., Briffaut, J., Lalande, J.-F., and Toinard, C. (2006). 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.

Diebold, P., Hess, A., and Schäfer, G. (2005). A honeypot architecture for detecting and analyzing unknown network attacks. In *14th Kommunikation in Verteilten Systemen 2005*, Kaiserslautern, Germany.

Harrison, M. A., Ruzzo, W. L., and Ullman, J. D. (1976). Protection in operating systems. *Communications of the ACM*, 19(8):461–471.

Hecker, C., Nance, K. L., and Hay, B. (2006). Dynamic honeypot construction. In *10th Colloquium for Information Systems Security Education*, University of Maryland, USA.

Holz, T. and Raynal, F. (2005). Detecting honeypots and other suspicious environments. In *Information Assurance Workshop*, pages 29–36, University of Maryland, USA.

Innes, S. and Valli, C. (2005). Honeypots: How do you know when you are inside one? In Valli, C. and Woodward, A., editors, *4th Australian Digital Forensics Conference*, Perth, Western Australia. School of Computer and Information Science, Edith Cowan University.

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

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

Nguyen, H. Q., Pouget, F., and Dacier, M. (2005). White paper: Integration of honeypot data into an alert correlation engine. Technical report, Institut Eurecom, France.

Provos, N. (2004). A virtual honeypot framework. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, Berkeley, CA, USA. USENIX Association.