

## Honeynet Maintenance Procedures and Tools

Carlos Henrique P. C. Chaves, Lucio Henrique Franco, Antonio Montes  
Renato Archer Research Center - CenPRA/MCT  
13069-901 - Campinas, SP - Brazil  
{carlos.chaves, lucio.franco, antonio.montes}@cenpra.gov.br

### *Abstract*—

**As part of an effort to improve honeynet's maintenance process, several procedures and tools automating high-interaction honeypot management tasks have been developed. Among the advantages of the adoption and use of these procedures and tools are the documentation of the maintenance procedures, the standardization of the collected data structure, the elimination of errors during the maintenance of honeypots, the automatization of the tasks that are executed and the reduction of the time between the deactivation and activation of a compromised honeypot.**

### I. INTRODUCTION

A Honeynet is not a product, it is a highly controlled network used to contain and analyze attackers in the wild. The primary purpose of a Honeynet is to gather information about threats that exist. New tools can be discovered, attack patterns can be determined, and attackers motives studied [1]. Honeynets are composed of an administration subnet and many hosts called honeypots, which are security resources designed to be probed, attacked or compromised, and to register these activities [2].

In the daily operation of a honeynet, it is not uncommon that one has to deactivate a honeypot, clean it up and put it back in operation. In addition, it is very convenient to have procedures and tools in place to rapidly install a new honeypot when required. This is particularly true if the honeynet has many honeypots or when many members of the project share the management task [3].

The procedures discussed in this paper applies to high-interaction honeypots in a research honeynet. It includes zeroing the disk, choosing the operating system, installing services, and finally configuring the data collection mechanisms and the preservation of the artifacts stored in the honeypot. These procedures help to avoid human errors, which could cause contamination of the capture data or reveal the presence of the honeytrap to attackers.

Together with these procedures, a bootable iso-image of a system based on the Slax Linux distribution<sup>1</sup> has been developed to automate the process of deploying a new hon-

eypot or reinstalling a compromised one. The image has scripts to perform a variety of tasks that will be described in the subsequent sections.

The remainder of this paper is organized as follows. Section II presents a brief description of honeypots and their classification. The procedures employed are discussed in Section III and Section IV presents the scripts created for their implementation. Section V concludes this paper.

### II. HONEYPOTS CLASSIFICATION

A honeypot is very different from most traditional security mechanisms. They are security resources that have no production value; no person or resource should be communicating with them. As such, any observed activity is suspected by default. Any traffic sent to the honeypot is most likely a probe, scan, or attack. Any traffic initiated by the honeypot means the system has most likely been compromised and the attacker is making outbound connections [4].

Honeypots can be classified by their level of interaction. High-interaction honeypots are hosts with real operating systems and services. They can be compromised and fully controlled by the attackers. As a result, they can be configured to collect a large amount of information, such as which vulnerability is being exploited, which tool is being used, and what is the attacker's motivation. This makes them very useful as a research tool. On the other hand, they can pose a serious threat to the environment, since the attacker can try to use the honeypot as a launching point of scan and attacks against other hosts in the network. This is the main reason for avoiding using high-interaction honeypots in production network. Instead one may use low-interaction honeypots, which can be as simple as a listening port or a full blown emulated network. This type of honeypots can be easier to install and should be less risky, since the attacks would be directed to fake services or operating systems. As a consequence, the collected data will consist mainly of scans and connections to emulated services. Despite this, it is becoming common to use low-interaction honeypots as a complement to intrusion detection systems in production network, where they

C. H. P. C. Chaves and L.H. Franco are also with National Institute for Space Research - São José dos Campos, SP - Brazil.

<sup>1</sup><http://slax.linux-live.org/>

can be particularly useful in the detection of internal worm attacks.

The management of high-interaction honeypots is a complex task, since there must be no traces of previous installations and of the monitoring tools, and in order to facilitate analysis the activities logging should be configured to follow the same standards independently of the operating system. These requirements lead to the need of developing procedures for the first time deployment, monitoring and redeployment of honeypots.

The present paper describes the procedures and tools used to maintain the honeynets of the Brazilian HoneyNet Project, made up of high-interaction honeypots [5], [6]. The architecture of these honeynets are based on the GenII architecture [7] and follows strictly the HoneyNet Project requirements [8]. These procedures will be detailed in the next section.

### III. DEVELOPED PROCEDURES

Honeynets usually contain several honeypots with different configurations, running a variety of operating system and local or network services. For their maintenance, it is convenient to develop methodologies and elaborate procedures. As mentioned above, this is particularly relevant if the honeynet has many honeypots or when many members of the project share the management task [3]. These procedures can be automated using scripts which reduce the time between the honeypot's deactivation and activation. The procedures and scripts contain key steps in a honeypot configuration, like zeroing the entire disk before installing the operating system from installation media [9] or removing traces of the honeypot configuration to avoid information leaks. Another important procedure describes a database for the storage of information about the honeypots of a honeynet, including type and version of the operating system, type and version of network services, compromises, etc. This will ease the process of gathering information for status reports and the maintenance of the honeypot's timeline.

#### A. Honeypot Deployment Procedure

The first step in the honeypot deployment is the definition of the operating system and services (including versions) that will be installed. Next, some basic procedures must be followed during the installation process:

1. The hard disk must be written with a constant data pattern (usually zeros). This procedure also has the benefit that disk image copies compress better, and that deleted files are easier to find [9].
2. The chosen operating system is installed, usually using the default installation options.
3. The chosen services are installed, configured and initialized.

4. The honeypot's configuration script is executed to create new users, configure the network settings, etc.
5. In order to monitor the attacker's steps, keystroke recording system are configured in the honeypot. These systems can send the commands typed by an attacker to a loghost, using the syslog system [10], and broadcast this information directly into the network [11].
6. A script to generate the MD5 and SHA1 hashes of the honeypot's files is executed. The hashes can be used later to discover which files have been changed by an attacker.
7. A script to generate the system's status is executed. This is composed of the output of some Unix commands like *ps*, *netstat*, *lsof*, *socklist* and *df*.
8. The MD5 and SHA1 hashes and the system's status files are stored in a loghost in the administration subnet which stores also the system's images.
9. Traces of the steps (5-8) above are erased.
10. The system's image is generated and exported to the loghost.
11. The new honeypot is connected in the honeynet and monitored until the moment when it will be turned off.

#### B. Post-deployment Procedure

After the honeypot installation, a standardized deployment log of every honeypot must be created and sent to the project members by e-mail. The log is based on the HoneyNet Project model [12].

The log file name follows a standard, which is HN-[org]-DEPLOY-[date]. The [org] tag must be changed by the name of the organization where the HoneyNet has been deployed, and the [date] tag must be changed by the actual date. Figure 1 shows an example of a log file created for a deployed honeypot.

```

HONEYPOT-BR SUMMARY:
-----
GenII HoneyNet with a class C IP address block exposed directly to
the Internet. Typical small organization /24 network.

IP RANGE: xxx.xxx.xxx.1-254

ACTIVE HONEYPOTS:
-----

System Name: Hp1
System IP: 10.0.0.36
System OS: RH 7.2, default, no patches
Date Deployed: 2004, Nov 24
Time Deployed: 10:50am
Responsible: Lucio Henrique Franco
System Summary:
- minimal installation, with portmap(4.0-38), with xinetd(2.3.3-1)
- kernel version: 2.4.7-10
Modifications to System:
- installed ntpd(4.1.0-4), httpd(1.3.20-16 with ssl & with
mod_perl 1.2 4_01-3), php(4.0.6-7), wu-ftpd(2.6.1-18)
- installed static bash patch
- installed smartl
- add default accounts
Status Change:

```

Fig. 1. Honeypot deployment log file.

Also when a honeypot is compromised, a new log file is created, containing the information about the attack, including what was the time of the first successful access, the explored service, the attacker's IP address and its *whois* information. Figure 2 shows an example of a log file created for a compromised honeypot. Finally, if the honeypot is deactivated, time and date of end of operation is appended in the Status Change field of the log file (see also Figure 2).

```

HONEYPOT-BR SUMMARY:
-----
GenII Honeynet with a class C IP address block exposed directly to
the Internet. Typical small organization /24 network.

IP RANGE: xxx.xxx.xxx.1-254

ACTIVE HONEYPOTS:
-----
System Name: Hp1
System IP: 10.0.0.36
System OS: RH 7.2, default, no patches
Date Deployed: 2004, Nov 24
Time Deployed: 10:50am
Responsible: Lucio Henrique Franco
System Summary:
- minimal installation, with portmap(4.0-38), with xinetd(2.3.3-1)
- kernel version: 2.4.7-10
Modifications to System:
- installed ntpd(4.1.0-4), httpd(1.3.20-16 with ssl & with
mod_perl 1.2 4_01-3), php(4.0.6-7), wu-ftp(2.6.1-18)
- installed static bash patch
- installed smartl
- add default accounts
Status Change:
- First succesfull attack: Sat Dec 04 15:43:50 GMT 2004 (Resp. Cae)
- Attack description: OpenSSL Buffer overflow exploit
- Attack origin: 10.10.0.104
- Whois: 10.10.0.0 - 10.10.255.255 -> Foobar Technologies
-> Brazil
- Offline on 12-10-2004 at 08:30am (Lucio)

```

Fig. 2. Compromised honeypot log file.

### C. Honeypot Monitoring Procedure

The operating system and applications running on the honeypots can have failures, caused by hardware failures, or problems in the filesystem, or simply the interruption of a service or all system as a result of an attack. These issues raise the necessity of a monitoring procedure. Once a week (period defined by the project), all honeypots are tested according to the procedure described below and showed in Figure 3.

The operation monitoring procedure is performed as follows:

1. The honeypot is disconnected from the Honeynet.
2. The system is first tested by trying to login from the console.
3. If the login is successful, a script is executed to obtain the honeypot's status (see Subsection IV-F). Otherwise, try to login again.
4. If the login fails twice, deactivate the honeypot.

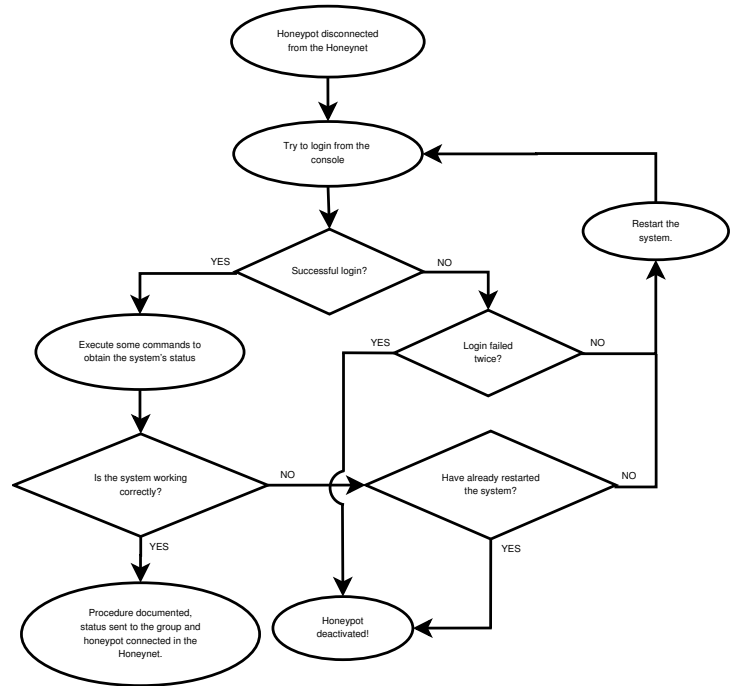


Fig. 3. Honeypots monitoring procedure.

5. Analyze the honeypots's status, and if it is working correctly, the results are documented, the status sent to the group and the honeypot is connected back to the Honeynet. If the honeypot is not working correctly and have not been restarted, restart the system and try to login again. If it has already been restarted, deactivate the honeypot.

### D. Honeypot Deactivation Procedure

From the moment the honeypot is deployed in the honeynet, it is constantly monitored by the administration subnet. Once it has been compromised, alerts are generated and sent to the project members by e-mail. The commands, actions and other information generated by the attacker are collected and sent to a loghost.

At that moment, one has to decide if it will be deactivated or not. Usually it has to be deactivated if some critical part of the operating system has been damaged, or if the attacker has made so many modifications to the system that it becomes nonfunctional.

The deactivation procedure includes the following steps:

1. Disconnect the honeypot from the Honeynet.
2. Collect and analyze the system status.
3. Generate the system's image and export it to the host containing the initial image (described in step 8 of the installation procedure).
4. The honeypot is ready to be reinstalled (procedure described at section III-A).

As mentioned before, some honeypots may be so damaged that one is unable to login from the console. In this

case, the honeypot must be rebooted using a live CD-ROM or floppy disk of a operating system (like Slax Linux distribution) to be able to generate the hard disk's image and export it to another host. The image generated is then analyzed and compared with the initial image. This process can reveal what has been modified in the honeypot by the attacker. Also the status files and file hashes can be compared with the initial ones.

In the next section the scripts generated to automate the procedures discussed above will be described.

#### IV. DEVELOPED TOOLS

The procedures described in the previous section are complex, and mistakes can be made easily. Moreover, it is interesting to automate the processes in order to make them faster and avoid human errors. These assertions raised the necessity of creating scripts to automate the procedures of deploying a honeypot. In order to be portable, they were developed using shell scripts.

##### A. Honeypot Deployment Script

This is the central script for the deployment of a honeypot. It has three options: (1) erase the hard disk and create a new partition; (2) erase one specified partition and use it for the installation of the operating system; and (3) generate and export the new system image.

In the case of installing a new honeypot, the first option must be used because the hard disk must be erased (written with zeros) and a new system partition must be created. In the case of restoring the honeypot and installing the same system that was used (from an image), the second option must be chosen because the same partitioning must be used. The first two options differ only at the beginning of the process. After that, the deployment process follows the same sequence, starting with the script asking for the location of the image to be copied to the honeypot disk. Then it uses SSH<sup>2</sup> to transfer the system's partition image from a remote host to the honeypot. Afterward, the swap partition is enabled and the LILO boot loader is installed using the configuration file (`/etc/lilo.conf`) from the image copied to the honeypot. The script finally prepares the new system to call the honeypot configuration script when the new system is booted and root logs in.

The third option is used to generate the image of the system's partition and export it to another host using the network. The script has the option of using Netcat<sup>3</sup> or SSH to transfer the image. Netcat is faster than SSH, but the second is more adequate to preserve the image's confidentiality and integrity.

It is important to mention that the scripts were originally developed to deal with Red Hat Linux honeypots (but they

were designed to be used with other distributions and operating systems).

##### B. Honeypot Configuration Script

The honeypot configuration script is used after the installation of the operating system, described above, and the reboot of the system. The script has two options: (1) configure a new honeypot; or (2) generate the system's status, the MD5<sup>4</sup> and SHA1<sup>5</sup> hashes, clean up all traces and export the image of the installed system to another host.

The first option must be used when installing a new honeypot. In this case, the image copied to the honeypot disk must be from a system installation iso-image. The first step in this option is to configure the root password and the honeypot's network (IP address, netmask, broadcast address, gateway, dns server, NTP server, honeypot's name and domain name). The script then automatically detects which is the operating system being installed to configure it properly. The next step is to create the user's accounts. After that, the script asks if any post-installation is required. In affirmative case, the script will return a root shell to let the user install and configure any other applications. The user will have to run the honeypot cleanup script manually after he finishes configuring the applications. Otherwise, the honeypot cleanup script will be executed automatically.

The second option will apply when restoring the honeypot from a compromise and re-installing the same system. In this case, the only thing that the honeypot configuration script will do is to execute the honeypot cleanup script.

##### C. SHA1 and MD5 Hash Generation Script

The function of this script is to generate the MD5 and SHA1 hashes of all files in a given directory. The script can have many directories as input, and all hashes generated will be stored in a file for each directory. Finally, these generated files are compressed.

The name of generated files is composed of the honeypot name, plus the directory path and the suffix "md5" or "sha1".

This script was developed for Linux, FreeBSD, and Windows (using cygwin<sup>6</sup>). It uses the command "`uname -s`" to detect which operating system is being used in order to set up the right parameters.

##### D. Honeypot Cleanup Script

The main goal of the honeypot cleanup script is to generate the MD5 and SHA1 hashes of the files in the selected directories, generate the system's status, clean up all traces and export the generated information to the host that will store the image of the system.

<sup>2</sup><http://www.openssh.com/>

<sup>3</sup><http://netcat.sourceforge.net/>

<sup>4</sup><http://www.ietf.org/rfc/rfc1321.txt>

<sup>5</sup><http://www.ietf.org/rfc/rfc3174.txt>

<sup>6</sup><http://www.cygwin.com/>

The first operation performed is to execute the MD5 and SHA1 hashes generation script to generate the SHA1 and MD5 hashes of the files in the following directories: /bin, /sbin, /dev, /etc, /usr/bin, /usr/sbin, /usr/local/bin and /usr/local/sbin. There might be others, depending on the operating system and applications.

Next the script generate the system's status using Unix commands such as *ifconfig*, *netstat*, *ps*, *lsof*, *lsmod*, *rpcinfo*, *df*, *socklist*, *rpm* and *last*. The applications output are stored in text files, using the name of the application plus the suffix ".txt".

Once the SHA1 and MD5 hashes have been generated and the system status has been collected, the honeypot information is ready to be exported to another host (it is suggested that this information be stored together with the new honeypot image). Again, SSH is used to transfer the files.

Finally, the last step is to remove all traces left from the configuration of the honeypot. To ensure that the deleted files won't be able to be restored, the application "shred"<sup>7</sup> is used to overwrite the files repeatedly in order to make it harder for even very expensive hardware probing to recover the data.

#### E. High-Interaction Honeypot Deployment CD-ROM

At the beginning of this section, the need to create scripts to automate the procedures for honeypot deployment is discussed. The development of the scripts described above has partially automated the process, but up to this point it is still necessary to copy and run some scripts in the honeypot manually.

This situation lead to the creation of a tool that automates the manual processes that remained. A high-interaction honeypot deployment CD-ROM has been created using the Slax Linux distribution. It includes all the scripts described in the previous subsections and automates all the required tasks. The CD-ROM executes the following steps:

1. The machine that will host the honeypot has to be booted from the CD-ROM and when root logs in, the honeypot deployment script is executed and the deployment process starts.
2. As described in Subsection IV-A, after the copy of the honeypot's operating system image to the hard disk, the other scripts are copied to the /root directory of the honeypot, and a call to the honeypot configuration script is added in the file /etc/profile of the honeypot.
3. A reboot is performed to boot the honeypot's own system. When root logs in, the configuration script is executed.
4. After all configuration and cleaning up, the system is rebooted using the Slax's CD-ROM to generate and export the honeypot's image.

<sup>7</sup><http://www.gnu.org/software/fileutils/>

5. Finally, the system must be rebooted again. Only now the CD-ROM must be ejected in order to allow the honeypot to boot from its own operating system.

#### F. Operation Monitoring Script

The operation monitoring script was created to automate and standardize the process of generating the honeypot's status. This shell script uses a CD-ROM containing statically compiled binaries of the applications used to perform the task. The script must be copied and run from a floppy disk, where the output of the script will also be stored. The honeypot operation monitoring script performs the following steps:

1. The floppy disk containing the script is mounted.
2. The script is executed from the floppy disk using the CD-ROM mounting point and the output directory as parameters.
3. The output directory is created in the floppy disk using the current date as its name.
4. The commands that generate the status of the honeypot are executed and their output is appended to the status file, located in the output directory.
5. The CD-ROM and the floppy disk are unmounted.

Figure 4 shows the status file generated for the honeypot hp1 that was stored in /mnt/floppy/15\_03\_2004/hp1. Some lines of the file were trimmed to make the figure fit in the available space.

## V. CONCLUSION

The creation of procedures and tools for the deployment of a high-interaction honeypot has great relevance to honeynet researchers. They automate the processes and improve the information gathering methods. Standards are created to be followed by all Honeynet administrators and this avoid human error like deactivating a honeypot without generating its final status. This kind of error would harm the comparison of the honeypot information immediately after installation and after the compromise. Also the time that a honeypot stay out of operation is reduced considerably.

The tools were being developed and used since the beginning of the Brazilian Honeynet Project, in december of 2001, when the definition of the honeypots configuration procedures have begun. The difficulties in the management of the honeynets showed the importance of developing tools to automate and document the processes of honeypot configuration, monitoring and deployment.

## VI. ACKNOWLEDGMENTS

The authors would like to thank Cristine Hoepers, Klaus Steding-Jessen and Marcelo Chaves for useful discussions.

## REFERENCES

- [1] H. Project, "Know your enemy: Honeynets," November 2003. <http://www.honeynet.org/papers/honeynet/index.html>.

```

$cat /mnt/floppy/15_03_2004/hp1

### hostname ###
hp1.localdomain

### df -h ###
Filesystem Size Used Avail Capacity Mounted on
/dev/sda1 1.7G 628M 1.0G 38% /
none 46M 0 46M 0% /dev/shm
/dev/fd0 1.4M 3.5k 1.3M 1% /mnt/floppy
/dev/cdrom 358M 358M 0 100% /mnt/cdrom

### w ###
12:46pm up 32 days, 0 min, 1 user, load average: 0.28, 0.42, 0.25
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root tty1 - 12:32pm 3.00s 4.00s 0.27s /bin/sh ./statu

### last ###
root tty1 Tue Jul 27 12:32 still logged in
root tty1 Fri Jun 25 12:53 - 12:54 (00:01)
reboot system boot 2.4.7-10 Fri Jun 25 12:52 (31+23:54)
root tty1 Wed Jun 16 10:24 - 10:25 (00:00)
reboot system boot 2.4.7-10 Wed Jun 16 10:22 (9+02:26)

### netstat -na ###
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:1024 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:21 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:11223 0.0.0.0:*
udp 0 0 0.0.0.0:3049 0.0.0.0:*
udp 0 0 0.0.0.0:111 0.0.0.0:*
udp 0 0 10.0.0.0:36:123 0.0.0.0:*
udp 0 0 127.0.0.1:123 0.0.0.0:*
udp 0 0 0.0.0.0:123 0.0.0.0:*
(...)

### ps -auwwwx ###
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.3 1384 340 ? S Jun25 0:31 init [3]
root 2 0.0 0.0 0 0 ? SW Jun25 0:00 [keventd]
root 3 0.0 0.0 0 0 ? SW Jun25 0:00 [kapm-idled]
root 8 0.0 0.0 0 0 ? SW Jun25 0:39 [kupdated]
root 468 0.0 0.6 1452 584 ? S Jun25 0:05 syslogd -m 0
root 473 0.0 1.0 2036 980 ? S Jun25 0:00 klogd -2
root 32404 0.0 0.0 0 0 ? Z Jul11 0:00 [sh <defunct>]
root 32403 0.0 0.0 948 32 ? T Jul11 0:00 /bin/sh -c sal
root 950 0.0 0.7 1580 684 ? S Jul12 0:00 CROND
(...)

### vmstat ###
procs -----memory----- -swap- --io-- -system- --cpu---
r b w swpd free buff cache si so bi bo in cs us sy id
0 0 0 1956 3096 1540 37336 0 0 0 0 8 11 0 1 6

### rpcinfo -p 0 ###
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100024 1 udp 1024 status
100024 1 tcp 1024 status

### uptime ###
12:46pm up 32 days, 0 min, 1 user, load average: 0.28, 0.42, 0.25

### printenv ###
HOSTNAME=hp1.localdomain
SHELL=/bin/bash
TERM=linux
HISTSIZE=1000
USER=root
(...)

```

Fig. 4. Summary of a status file.

- [2] L. Spitzner, "Hosus (honeypot surveillance system)," *login: Magazine of Usenix and Sage*, vol. 27, December 2002. <http://www.usenix.org/publications/login/2002-12/pdfs/spitzner.pdf>.
- [3] L. H. Franco and A. Montes, "Procedimentos e Ferramentas para Manutenção de Honeypots de Alta Interatividade," in *Anais do I WorkComp Sul - Unisul - Universidade do Sul de Santa Catarina (WorkComp Sul'2004)*, (Florianópolis, SC), May 2004. ISBN 85-86870-25-0.
- [4] L. Spitzner, *Honeypots: Tracking Hackers*. Addison Wesley, 2002.
- [5] A. B. Filho, A. S. M. S. Amaral, A. Montes, C. Hoepers, K. Steding-Jessen, L. H. Franco, and M. H. P. C. Chaves, "HoneyNet.BR: Desenvolvimento e Implantação de um Sistema para Avaliação de Atividades Hostis na Internet Brasileira," in *Anais do IV Simpósio sobre Segurança em Informática (SSI'2002)*, (Sao Jose dos Campos, SP), pp. 19-25, November 2002.
- [6] H. Project, "Status report: January - august 2004," September 2004. <http://www.honeynet.org.br/status-report/>.
- [7] H. Project, "Know your enemy: Genii honeynets," November 2003. <http://www.honeynet.org/papers/gen2/index.html>.
- [8] H. Project, "Honeynet definitions, requirements, and standards," October 2004. <http://project.honeynet.org/alliance/requirements.html>.
- [9] W. Venema and D. Farmer, "Being prepared for intrusion," *Dr. Dobb's Journal*, April 2001. <http://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/tct/being-prepared.html>.
- [10] H. Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison Wesley, 2001.
- [11] L. G. C. Barbato and A. Montes, "SMaRT - Session Monitoring and Replay Tool," in *Apresentado no Grupo de Trabalho em Segurança de Redes (GTS'02.2003)*, (Rio de Janeiro, RJ), December 2003.
- [12] H. Project, "Honeypot deployment log," December 2001. <http://www.honeynet.org/alliance/AppendixA.txt>.