

Sebek 3: tracking the attackers, part two

Raul Siles, GSE 2006-02-13

Introduction

In [part one of this series](#), we discussed the current Sebek development and its integration with GenIII Honeynets. In this article, we take it a step further and focus on best practices to deploy Sebek inside a GenIII Honeynet, as well as the new Sebek "write" patch. This patch is a cutting edge improvement that makes it possible for a security professional to watch all the attacker's activities in real time, in a similar way that one records his family's evolutions with the video camera.

Best practices to install, configure and run Sebek clients

Installation of Sebek version 2, plus its configuration and usage are extensively covered on the original Sebek KYE paper. [\[ref 1\]](#) This article discusses best practices to deploy Sebek version 3 on a GenIII Honeynet environment and provides details about the different Sebek Linux and Windows versions currently available.

The Sebek 3 client installation and configuration process is very similar to those from former versions. Figure 1 illustrates the standard procedure to build and configure Sebek for Linux.

```
Build process:
# ./configure
# make
# make install      (optional)

Configuration process:
# vi sbk_install.sh

Clean up process:
# cd ../; rm -rf sebek*
# rm -f /usr/local/bin/sbk_install.sh /usr/local/bin/gen_fudge.pl
```

Figure 1. Sebek build, configuration and clean up process on Linux.

The compilation process generates a binary tar file (sebek-linux-3.0.3-bin.tar or sebek-lin26-3.1.2b-bin.tar.gz) that must be copied and installed into the target honeypot. This file only contains the scripts (parameters.sh and sbk_install.sh) and kernel modules required to install Sebek.

It is recommended that one compile the module on a separate system running the same Linux kernel and deploy Sebek using this binary tar file. This is to reduce the traces about Sebek's existence in the production honeypot. If Sebek is installed in the honeypot itself,

through the "make install" command, it is advisable to remove the Sebek source code directory and the two scripts installed under "/usr/local/bin", as was illustrated in Figure 1. Additionally, the user shell history files for many shells (such as Bash) typically contain the commands executed during the compilation process; these should be removed too.

To install Sebek on a Windows platform, the two executable files, "Configuration Wizard.exe" and Setup.exe must be copied to the honeypot. The kernel driver installer, Setup.exe, guides you through the kernel driver installation process. The wizard installs the driver on the standard Windows location, C:\WINDOWS\system32\drivers on Windows XP or C:\winnt\system32\drivers on Windows 2000, using the default driver name SEBEK.SYS. From a security point of view, the installer should be run with the "/N=NAME" command line argument to specify a non-standard driver name.

Figure 2, below, illustrates the Sebek Windows client setup wizard where a different driver name, SCISICTRL, has been specified. As in Linux, a defensive countermeasure is not to keep a copy of the configuration files on the honeypot itself once Sebek has been installed.

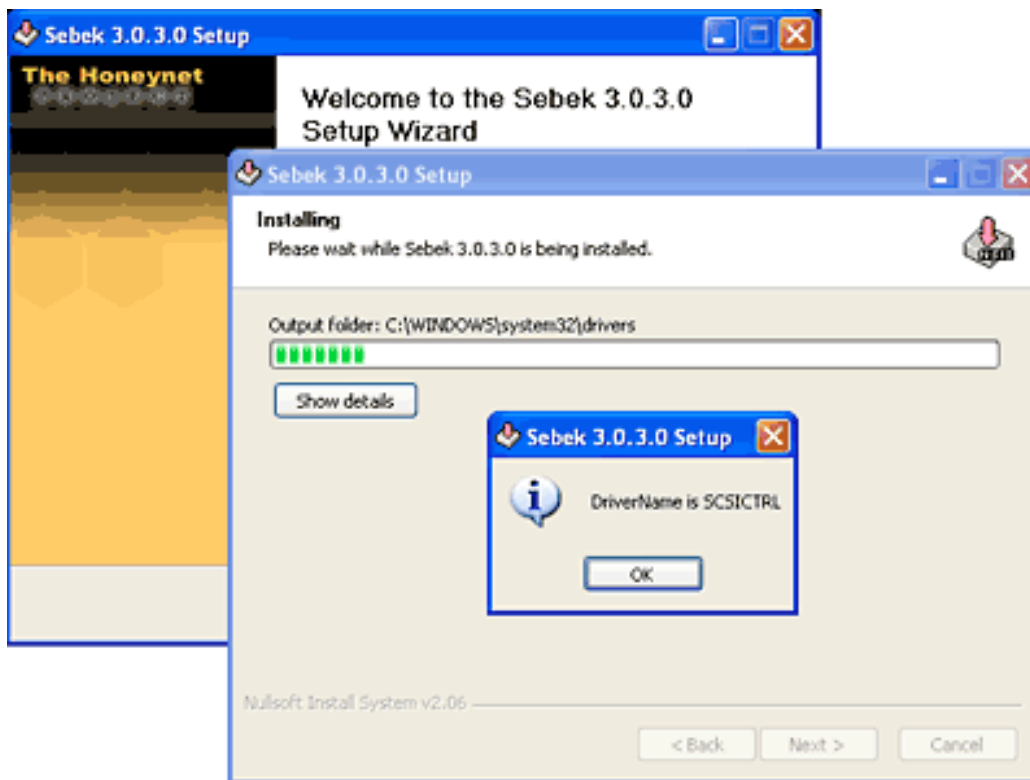


Figure 2. Sebek installation process on Windows.

The following recommendations detail best Sebek configuration practices for GenIII Honeynets from a security perspective. Sebek is configured on Linux by editing the "sbk_install.sh" script, which contains the different Sebek configuration variables: DESTINATION_IP and DESTINATION_MA. These variables should contain the default gateway IP and MAC addresses. If by chance the intruder could recover the Sebek configuration, these values won't disclose the real Sebek server identity. Besides, if the intruder could see Sebek packets these won't look suspicious. The packets addressed to the default gateway look like any other traffic and won't help to identify where the Sebek server resides. In a GenIII Honeynet, the stealthy Honeywall acting in bridge mode will collect all Sebek traffic no matter what values these variables have been set to.

It is also important to ensure that the default gateway does not generate ICMP port unreachable messages - it would tip off the intruder about Sebek. Below is a brief description of some of the key configuration variables:

- **DESTINATION_PORT**: All honeypots on the same honeynet must use the same destination port. This is because this port is used by the Sebek server to identify interesting traffic and by the Sebek clients (together with the magic value) to hide other honeypots Sebek traffic.
- **MAGIC_VAL**: The magic value is used in conjunction with the destination port to determine which packets to hide in the honeynet. It must be set to a difficult to guess value.
- **KEYSTROKE_ONLY**: This variable should be set to zero (the default value is one) in order to collect all read syscalls and be able to recover entire files exchanged through, for example, SCP. Enough disk space is required in the Honeywall to save the large amount of information generated by the honeypots when running in this mode.
- **SOCKET_TRACKING**: Socket activity must be monitored (by default) to be able to establish the process and network relationships showed through Walleye in the [first part of this article](#).
- **TESTING**: In a production environment, this value should be set to zero (by default), so that the Sebek module is hidden and can't be removed using the "rmmod" command.
- **MODULE_NAME**: The Sebek module name must be set to a non-suspicious value not related with Sebek at all, such as "scsictrl" (SCSI controller kernel module). Setting this option is even preferable to leaving this variable blank, which would thereby select a random name for the module. The random name could look suspicious to an intruder if he is able to see it.
- **WRITE_TRACKING**: This option is only available in the Linux 2.6 version (3.1.2b) and allows one to activate the experimental feature implemented by the Sebek "write" patch, covered later in this article.

Some variables, such as **INTERFACE** or **SOURCE_PORT**, have not been mentioned above because there is no special security recommendations for GenIII Honeynets that must be associated to them. More detailed information about the variables meaning is available on the tool documentation and script files. [[ref 1](#)] [[ref 2](#)]

Sebek is set up on Windows through a graphical tool called "Configuration Wizard.exe". This wizard guides you through the Sebek customization process to define all Sebek configuration variables. These are similar to the Sebek Linux client variables explained previously, except that there is a lack of granularity when defining Sebek's capabilities, as illustrated below in Figure 3. The keystroke, socket, write or testing modes cannot be specified. Besides, the Sebek Windows version does not allow one to set the source port. It always uses the default "1101" value.

The Sebek Windows client provides an additional variable that contains the name of the process entitled access to the Sebek driver. Once the Sebek driver has been installed, it is hidden and can only be accessed and reconfigured using a program with a specific name, the one specified in this variable. It is recommended to change the name of the default

"Configuration Wizard.exe" tool used to manage Sebek, and therefore the name of this variable, to a non-standard value.

Figure 3, below, compares the main differences between the Sebek Linux and Windows clients.

Sebek version 3	Linux 2.4	Linux 2.6	Windows
Survives reboots?	No	No	Yes
Advanced configuration granularity: <ul style="list-style-type: none"> • WRITE_TRACKING (experimental) • SRC_PORT • KEYSTROKE_ONLY • SOCKET_TRACKING • TESTING 	Yes (w/ patch) Yes Yes Yes Yes	Yes Yes Yes Yes Yes	No No (1101) No No No
Real rootkit hiding capabilities (*)	No	No	Yes
Additional hiding module required	Yes (cleaner. o)	No (built-in)	No (built-in)
Option not to replace the raw socket implementation (at compile time)	No	Yes	No
Complementary reconfiguration tool	No	No	Yes

(*) These capabilities were removed on purpose from the initial Sebek Linux versions to reduce chances of misuse. Sebek was based on the Adore kernel rootkit.

Figure 3. Sebek Linux and Windows client's capabilities and differences,

As can be observed in Figure 3, there are still some pending improvements in both platforms. For instance, it would be desirable to have the capability to survive reboots on Linux and the fine-grained configuration capabilities on Windows.

Once the client has been installed and configured, Sebek is started by running the "sbk_install.sh" script on Linux or by rebooting the honeypot on Windows.

Best practices to run the Sebek server

The Sebek 3 server (v3.0.3) is installed and configured by default on the Roo Honeywall, so the details about how to compile it won't be covered. It follows the Linux standard build

procedure.

The Sebek server tools [ref 2] manage the network traffic logged by the Sebek clients. The main extraction tool is called "sbk_extract" and it is based on libpcap. [ref 3] It simply sniffs traffic in promiscuous mode from the Honeywall internal interface, filtering by the Sebek destination port. No matter who the Sebek traffic is addressed to, it will be captured by the Sebek server. Sebek records are processed by this tool and their binary representation is sent to the standard output. Typically, this output is piped to the input of the "sbk_ks_log.pl" utility, which processes Sebek's records and writes detailed keystroke logs to the standard output, as illustrated below in Figure 4.

```

root@localhost tmp1# sbk_extract -i eth0 -p 29985 2>/dev/null | sbk_ks_log.pl
[2005-11-27 23:26:47 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#uname -a: id
[2005-11-27 23:26:52 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#ls /
[2005-11-27 23:27:05 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#cat /etc/shadow | grep root
[2005-11-27 23:27:13 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#ps -ef | grep syslog
[2005-11-27 23:27:27 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#netstat -ant
[2005-11-27 23:27:38 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#psud
[2005-11-27 23:27:48 Host:192.168.100.66 UID:0 PID:1176 FD:0 INO:78497 COM:bash
|#scp root@192.168.100.66:/usr/bin/nc .
[2005-11-27 23:27:49 Host:192.168.100.66 UID:0 PID:15451 FD:3 INO:15231 COM:ssh
|#SSH-1.99-OpenSSH_3.5p1
[2005-11-27 23:27:52 Host:192.168.100.66 UID:0 PID:15451 FD:4 INO:78495 COM:ssh
|#yes
[2005-11-27 23:27:56 Host:192.168.100.66 UID:0 PID:15451 FD:4 INO:78495 COM:ssh
|#verysecret
[2005-11-27 23:27:56 Host:192.168.100.66 UID:0 PID:15458 FD:7 INO:15229 COM:scp
|#C8755 18236 nc
-

```

Figure 4. Sebek standard usage: sbk_extract and sbk_ks_log.pl.

Another tool, called "sebekd.pl", can be used to upload the Sebek data into a MySQL database, such as the one available inside the Roo Honeywall. This tool invokes "sbk_extract" internally and inserts Sebek records as database records. "sebekd.pl" replaces the "sbk_upload.pl" command available in previous versions.

In order to avoid potential vulnerabilities on libpcap or the tools themselves, and be independent of the client implementation, it is recommended that you take proper security precautions to protect the Sebek server and minimize the risk of a Sebek compromise.

From a security perspective, the "sbk_extract" tool must run in a chroot environment. Figure 5 illustrates an example of how to invoke a chrooted instance of the tool. Both options, -u and -c, must be used simultaneously in order to avoid root to escape from the chroot environment. [ref 4] [ref 5]

```
# sbk_extract -i eth1 -p 29905 -u eric -c /tmp/chroot | sbk_ks_log.pl
```

Figure 5. Sebek server inside a chroot environment

Additionally, the environment should be protected with a system call policy enforcement application, such as Systrace [ref 6] and with stack protection software, such as stack randomization [ref 7] and ExecShield. [ref 8] [ref 9] These two stack security mechanisms are active by default on the Roo Honeywall Linux 2.6 kernel, based on the Fedora Core 3 distribution.

Finally, to decrease the likelihood of detection using the methods pointed out in the [first part of this article](#), it is recommended that one modify the default Sebek client configuration variables as previously covered. A further security step would be to modify Sebek's source code to make it different from the publicly available version.

Continued...

Sebek's kernel "write" syscall patch

Sebek is the most advanced honeynet-based data capture tool available. However, the [first part of this article](#) hinted at one of the current Sebek limitations: it is possible to gather what the attacker typed but not the response received.

To avoid this constraint, a Sebek patch was developed with the goal of capturing the whole interaction between the honeypot and the intruder. This new version allows us to see not only what the attacker typed by capturing the "read" syscall activities, but also the response the attacker received by capturing the "write" syscall activities too. The patch intercepts the following "write" system calls: "__NR_write", "__NR_writew" and "__NR_pwrite".

The patch was developed for the current Linux 2.4 Sebek client version, 3.0.3, and the corresponding Sebek server 3.0.3 version. The "write" functionality was experimentally included in the first official Sebek client beta version for Linux 2.6 (version 3.1.2b), released on October 2005. The Linux 2.6 version uses a new configuration variable to switch on and off the "write" capabilities, called WRITE_TRACKING, as has been explained in the first section of this article.

Once the Sebek "write" patch has been downloaded [ref 10], it can be applied by following the standard Linux patching procedures. There are two patch tar files associated to the Sebek client and server, respectively. The client patch is installed by copying the four available source files to the standard Sebek client directory, "sebek-linux-3.0.3", and by running the commands illustrated below in Figure 6. The patch differences are contained in the "sebek_write.h.patch" and "sebek_write.c.patch" files, and the new files contents are in the "sebek_write.h" and "sebek_write.c" files. These will replace the standard "sebek.h" and "sebek.c" files with the corresponding new functionality that includes the "write" syscall changes.

```
On the honeypot:  
# tar xvzf sebek-linux-3.0.3-write.tar.gz
```

```
# cp sebek-linux-3.0.3-write/sebek* sebek-linux-3.0.3
# cd sebek-linux-3.0.3
# patch -p0 < sebek_write.h.patch
# patch -p0 < sebek_write.c.patch
```

Figure 6. Sebek "write" client patch installation.

On the Roo Honeywall that is acting as the Sebek server, the new customized data analysis application, called "sbk_viewer.pl", must be copied to the "/usr/sbin" directory where the Sebek tools reside. This tool is available on the server patch tar file. This tar file also contains patches for the Sebek server source code, but they are not required when the GenIII Roo Honeywall is used. The tool requires execution permissions to run. Figure 7 illustrates the server installation steps.

```
On the Honeywall:
# tar xvzf sebekd-3.0.3-write.tar.gz
# cp sebekd-3.0.3-write/sbk_viewer.pl /usr/sbin
# chmod 550 /usr/sbin/sbk_viewer.pl

# sbk_extract -i eth1 -p 29905 | sbk_viewer.pl
```

Figure 7. Sebek "write" server patch installation and usage.

Once the standard Sebek Linux client has been patched and the new tool has been added to the Honeywall, the Sebek "write" functionality can be used by installing the Sebek kernel module on the honeypot and by invoking the new tool on the Honeywall. The tool uses a similar method to the one used by the standard tools, where the output from the "sbk_extract" tool is piped to "sbk_viewer.pl", as illustrated in Figure 7.

Figure 8, below, illustrates how the tool, acting as a video camera, allows the security professional sitting at the Honeywall console to watch in real time the attacker's actions performed in the honeypot. By default, the attacker's input is displayed in red and the response received is presented in blue. When no command line arguments are used, the tool shows just "read" and "write" syscalls related with the attacker keystrokes, only focusing on the "stdin", "stdout" and "stderr" file descriptors.

```

[root@localhost tmp]# sbk_extract -i eth0 -p 29985 2>>/dev/null | sbk_viewer.pl

#uname -a; id
Linux localhost.localdomain 2.4.28-8 #1 Thu Mar 13 17:54:28 EST 2003 i686 i686 i
386 GNU/Linux
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel)
[root@localhost root]#
#ls /
bin dev home lib misc opt root tmp var
boot etc initrd lost+found mail proc sbin usr
[root@localhost root]#
#syslogd -v
syslogd 1.4.1
[root@localhost root]#
#netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
[root@localhost root]#
#scp root@192.168.100.66:/usr/bin/nc .
#SSH-1.99-0openSSH_3.5pl
#yes
Warning: Permanently added '192.168.100.66' (RSA) to the list of known hosts.
#verysecret
#C8755 18236 nc
nc                               100% |*****| 10236      00:00 [
[root@localhost root]#
#ls -l nc
-rwxr-xr-x 1 root root 10236 Nov 28 08:34 nc
[root@localhost root]#
#nc -l -p 666
-

```

Figure 8. Watching the attacker's actions through sbk_viewer.pl.

The tool implements multiple command line options to manage the information displayed. It can monitor all the "read" and "write" syscalls (not only the three file descriptors previously mentioned) using the verbose "-v" option. Additional verbose information provided by the "open" and "socket" syscalls is available through the "-V" option. Both options can be used simultaneously for even more verbose output. Figure 9 illustrates the tool's verbose capabilities, showing the commands typed, the response obtained, and its correlation with timestamps, client hosts, users ids, processes, file descriptors, inodes and commands.

```

12885-11-28 08:12:16 Host:192.168.100.66 UID:0 PID:1176(1178) FD:0 INO:78497 COM
:bash |uname -a; id
12885-11-28 08:12:16 Host:192.168.100.66 UID:0 PID:1176(1178) FD:2 INO:78497 COM
:bash |#
12885-11-28 08:12:16 Host:192.168.100.66 UID:0 PID:15482(1176) FD:1 INO:78497 CO
M:uname |#
Linux localhost.localdomain 2.4.28-8 #1 Thu Mar 13 17:54:28 EST 2003 i686 i
386 GNU/Linux
12885-11-28 08:12:16 Host:192.168.100.66 UID:0 PID:15483(1176) FD:1 INO:78497 CO
M:id |#
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel)
12885-11-28 08:12:16 Host:192.168.100.66 UID:0 PID:1176(1178) FD:2 INO:78497 COM
:bash |#
[root@localhost root]#

```

Figure 9. sbk_viewer.pl verbose output.

The Sebek client intercepts all the system calls executed on the honeypot; therefore, the

data collected and displayed by the "sbk_viewer.pl" tool could contain information belonging to various users and their multiple processes. Figure 10 illustrates the default tool output, intermixing data of the activities associated to two different sessions held by a single attacker on a honeypot. This scenario also occurs when multiple attackers have compromised a honeypot and are logged simultaneously.

```
[root@localhost root]# sbk_extract -i eth0 -p 29985 2>/dev/null | sbk_viewer.pl

#psd
/tep
[root@localhost tmp]#
#tty
/dev/tty1
[root@localhost tmp]#
#uname -a
Linux localhost.localdomain 2.4.28-8 #1 Thu Mar 13 17:54:26 EST 2003 i686 i686 i
386 GNU/Linux
[root@localhost tmp]#
#psd
/root
[root@localhost root]#
#tty
/dev/tty2
[root@localhost root]#
#id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel)
-
```

Figure 10. sbk_viewer.pl output from multiple sessions intermixed.

In this scenario, when the default command line options are used, it is not possible to segregate the data associated to each session. In order to avoid this constraint, the tool provides several filtering options to focus on specific honeypot events. This helps to reduce the amount of information displayed and to discriminate the information associated to each process or user.

In order to define the display filters, the tool must be run in verbose mode, as previously illustrated in Figure 9. This verbose output provides the information (or metainformation) required to set up the filtering conditions, such as the process identifier (PID 1176, "PID:" column). Once the process number has been identified, another "sbk_viewer.pl" instance can be executed simultaneously on a different Honeywall console filtering by the process id (PID and PPID) by using the "-p" option. Figure 11 illustrates how this filter helps to get only the "read" and "write" syscalls associated to a specific process, like the shell used by the attacker ("bash", "COM:" column from Figure 9).

```
[root@localhost tmp]# sbk_extract -i eth0 -p 29985 2>/dev/null | sbk_viewer.pl -
p 1176
filtering by PID/PPID number (1176)

#netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80               0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
-
```

Figure 11. sbk_viewer.pl filtering by PID (1176).

A similar procedure applies to the other filtering mechanisms. It is possible to filter by

syscall type ("-t" option) or by command string ("-c" option). Figure 12 illustrates the syscall filter, where only the read (type 0) system calls are displayed. This example shows just what the attacker types, in a similar way to the standard "sbk_ks_log.pl" tool.

```
[root@localhost root]# sbk_extract -i eth0 -p 29985 2>>dev/null | sbk_viewer.pl
-t 0
filtering by SYSCALL type (0)

#uname -a; id
#ls /
#cat /etc/shadow | grep root
#netstat -ant
-
```

Figure 12. sbk_viewer.pl filtering by syscall type (type 0, read).

Figure 13 below illustrates the command name filter, where only the data from the processes instantiating a "bash" shell is being displayed. This allows us to get just the activities associated to a specific type of processes identified by the command the attacker executes. This functionality can be used for the analysis of individual vulnerable services running on the honeypot, allowing to display the events associated to, for instance, the Web server related to all the "httpd" processes.

```
[root@localhost root]# sbk_extract -i eth0 -p 29985 2>>dev/null | sbk_viewer.pl
-c bash
filtering by COMMAND string (bash)

#ls -l /
[root@localhost root]#
#ls -l /root
[root@localhost root]#
#echo "You've been hacked!"
You've been hacked!
-
```

Figure 13. sbk_viewer.pl filtering by command (bash).

These three different filters can be combined in a single command. The tool uses the AND logical operator to create the composite filter. This allows one to filter, for example, all the "read" syscalls associated to process id 1176: "sbk_viewer.pl -p 1176 -t 0".

The current Sebek "write" patch in its initial development has some limitations. The main one is that it introduces some performance issues on the client side. The patched version works fine on a Linux honeypot without a graphical environment. If it is run on a honeypot running X-windows, the amount of work required to capture and log through the network all the "write" events taking place at the kernel level will freeze the honeypot. The "write" functionality is still in beta testing phase in the official Sebek Linux 2.6 branch due to instability problems.

Finally, it is well worth mentioning how important would be to have the Sebek "write" patch functionality in the official Honeywall. It would be very interesting to see a future integration of this patch with the Walleye's data analysis graphical capabilities introduced in the first part of this article.

Concluding part two

In the first article of the series, I described the latest Sebek 3 version capabilities, its protocol specification and how this tool integrates with GenIII Honeynets. The article also pointed out some of the current Sebek limitations that should be solved in future versions. Now in the second part, some Sebek deployment best practices have been shown, along with some details about how the default Sebek functionality has been improved with a "write" patch, to display the interaction between an attacker and a honeypot. The main goal of a Honeynet is to learn about the tools, tactics and motivations of the attacker community. This requires us to capture as much data as possible of the attacker's activities. Sebek provides security professionals with the information required to analyze all the intruder's actions details, and works admirably to fool an attacker into providing enough information to become vulnerable.

References

[ref 1] "Know Your Enemy: Sebek. A kernel based data capture tool". The Honeynet Project. November, 2003.

<http://www.honeynet.org/papers/sebek.pdf>

[ref 2] "Sebek Homepage". The Honeynet Project. 2005.

<http://www.honeynet.org/tools/sebek/>

[ref 3] "Libpcap". LBL. Tcpdump. 2005.

<http://www.tcpdump.org>

[ref 4] "How to break out of a chroot() jail". Simes. 2002.

<http://www.bpfh.net/simes/computing/chroot-break.html>

[ref 5] "Best Practices for UNIX chroot() Operations". Steve Friedl's Unixwiz.net Tech Tips. 2002.

<http://www.unixwiz.net/techtips/chroot-practices.html>

[ref 6] Systrace for Linux. Neils Provos, Marius Aamodt Eriksen. 2004.

<http://www.systrace.org>

[ref 7] Linux 2.6 Stack Protection. xwing's. 2005.

<http://kerneltrap.org/node/5783>

[ref 8] "Dealing with an Overflowing Buffer in Red Hat Fedora Linux 3". J. Hall, P. G. Sery. 2005.

<http://www.dummies.com/WileyCDA/DummiesArticle/id-2900.html>

[ref 9] "New Security Enhancements in Red Hat Enterprise Linux v.3, update 3". Arjan van de Ven. 2004.

http://www.redhat.com/f/pdf/rhel/WHP0006US_Execshield.pdf

[ref 10] Sebek "write" patch. Raul Siles. December 2005.

http://www.raulsiles.com/docs/Sebek_write_patch.html

Credits

To Monica and Eric - infinite thanks for making my life so fascinating and special.

The Sebek "write" patch was developed as part of a HoneyNet research project between Hewlett-Packard and Telefonica Moviles in Spain. Thanks to David Perez for his collaboration on this project. Thanks to the HoneyNet Project, especially to Lance Spitzner and Ed Balas, and to all my colleagues of the Spanish HoneyNet Project www.honeynet.org.es.

The techniques and tools described in this two-part article are presented and practiced in the SANS "[Deploying GenIII HoneyNets](#)" course, which I have developed.

About the author

[Raul Siles](#) is a senior security consultant with Hewlett-Packard. His current research interests include honeynet technologies, kernel rootkits and wireless security. He is one of the few individuals who have earned the GIAC Security Expert (GSE) designation. More information can be found on his website, www.raulsiles.com.

Copyright © 2006, SecurityFocus.

[Privacy Statement](#)

Copyright 2006, SecurityFocus