CyberSoft White Papers

Use of Avatar™ on a Honey Pot System

By James A. Roach, Jr.
CyberSoft, Inc.
11 May 2001

Needing to set up security for a high visibility web site, I decided to implement control by using CyberSoft's Avatar™. The target machine is a Sun Micro Systems Sparc(TM) workstation running the Solaris(TM) 8 operating system.

You will need to read the manual page for Avatar. You can find this manual page on the CyberSoft, Inc. web site, www.cybersoft.com, under the Customer Support link. There you can learn the command syntax.

A user with system administration permissions must run avatar. For my Solaris system root is the system administrator user id. The root user id will be used to run all testing and, when implemented, production execution.

You will learn that there is a configuration file required. The specifications for this project are the existence, permissions, ownership and MD5 cryptographic signature for every local file on the system are to be recorded but not the contents since the entire idea of a honey pot system is to record what a "hacker" would modify on the system. The cryptographic signature for the /proc and /dev files is illogical and will also not be recorded. All named pipes are to be skipped.

At first you may consider using a modified version of the sample configuration file, as suggested by the manual page. The results of this configuration file exceeded my needs. Not only is every file on the system recorded in the database, but also every file on every mounted link to other systems. I do not need to duplicate the file systems from the other computers, as they have their own Avatar databases.

First Avatar Configuration file

Repos /

For testing you will want to write a shell script to run Avatar with the create command. This shell script redirects the stderr output to a file in /tmp. My example shell script also places the baseline directory in the /tmp directory. On the test computer the file system is very full and does not have enough space for the baseline databases. The baseline directory must not exist before Avatar is run with the create command. Therefore the rm command is in this shell script.

Avatar Create Shell Script

```
#!sh
AVATAR_CFG_DIR=/tmp
export AVATAR_CFG_DIR
rm -rf /tmp/oops /tmp/baseline
/usr/bin/avatar create / /tmp/baseline ${AVATAR_CFG_DIR}/avatar.cfg 2>
/tmp/oops
```

Note: Indented line is a continuation of previous line

I then wrote a shell script to select only the local files using the find command. To receive the required results, I chose to use three find commands. The first find command returns all of the regular files on the local system, skipping all named pipes, /proc, and /dev files. Each file name is appended to the configuration keys. In this case the configuration keys are e, for existence, p, for permissions, o, for ownership, and s, for cryptographic signature.

The second find command returns all of the /proc files. Their filenames and paths are appended to all of the configuration keys listed above except for s, the cryptographic signature. This is not required as it changes every time a program uses that program identification number or pid.

The last find command gives all of the /dev files. Again, their filenames and paths are appended to the same configuration keys as used for the /proc files.

While this method records every file, proc and dev file on the system, it proved to be unsatisfactory. It creates a static snap shot of the file system in the configuration file. It quickly become apparent that this left many large holes in the security of the system. Avatar would not be able to dynamically add any new files created on the system to the configuration file. The configuration file was almost two mega-bytes in size, using up valuable system resources. The contents of the configuration file are duplicated in the database generated by Avatar.

The shell script takes one to two hours to run, depending on system utilization. This is actually OK since running Avatar Create is a rare occurrence. Most Avatar command executions will be Avatar Check or Avatar Correct. In the case of our Honey Pot Server only Avatar Check will be run without the option to run Avatar Correct.

Sample avatar_cfg.sh File

```
AVATAR_CFG_DIR=/tmp
export AVATAR_CFG_DIR
find / \( \( -type f -a -local -mount -a ! \( -name core \) -a !
\( -fstype proc \) -a -perm -u+r,-g+r,-o+r \) \) -print | \
sed "s/\(.*\)/epos \1/" > ${AVATAR_CFG_DIR}/avatar.cfg
find /proc \( \( -type f -a -local -mount -a ! \( -name core \)
 -a -perm -u+r,-g+r,-o+r \) \) -print | \
sed "s/\(.*\)/epo \1/" >> ${AVATAR_CFG_DIR}/avatar.cfg
find /dev \( \( -type f -a -local -mount -a ! \( -name core \)
-a -perm -u+r,-g+r,-o+r \) \) -print | \
sed "s/\(.*\)/epo \1/" >> ${AVATAR_CFG_DIR}/avatar.cfg
echo `wc -l ${AVATAR_CFG_DIR}/avatar.cfg |
sed "s/ *\([0-9][0-9]*\).*/\1/"` \<

"files in configuration file" ${AVATAR_CFG_DIR}/avatar.cfg
```

Initially, the Avatar Configuration shell script did not handle the /dev or /proc files. These were added after the first find correctly built an avatar.cfg file. The echo statement at the end of the shell script provides positive feedback to the operator.

The need to use the recursive ,R, feature is necessary to create a manageable configuration file. My first attempt at a configuration file is modified to utilize this feature. I do not want to recursively scan the mounted directories on other machines. Therefore, an entry for each mounted directory is added, the second through forth lines. These entries add to the baseline database the existence, permissions, and ownership of the mounted directories. The MD5 signature is not required as it can change from sources on the other computers.

The MD5 signature is not useful when considering the files in the /dev, /devices, and /proc directories. The fifth through seventh lines in the configuration file implement this concept. These entries need to be recursive, the R, to assure that all files and directories are baselined.

The contents of the /tmp directory, by definition, are not required. But the existence, ownership, and permissions of the /tmp directory should be recorded in the baseline. The last line shows this requirement.

Final Configuration File

```
Repos /
epos /export/home/cvs
epos /export/home/engineering
epos /export/home/pubshare
Repo /dev
Repo /devices
Repo /proc
epo /tmp
```

Avatar notifies the user of what it is doing as it runs. Notification is displayed to the screen from standard error, stderr. A sample of this output is displayed below. The first two lines show output from the LICENSE checking routine. Avatar checks the license file named LICENSE in the VSTK_HOME environment variable directory. The default location for this file is located in

/usr/lib/vstk. Examples of each of the configuration file entries are given for cross-referencing with the example of the final configuration file above.

Sample Output From Avatar

```
##==> FOO/BAR Site License:
##==> VSTKP SITE LICENSE KEY
baselining Repos /
baselining Repos /.TTauthority
baselining Repos /.Xauthority
baselining Repos /.dt
baselining Repos /.dt/Desktop
baselining epos /export/home/cvs
baselining epos /export/home/engineering
baselining epos /export/home/pubshare
baselining Repo /dev
baselining Repo /dev/.devfs_eventmgr
baselining Repo /dev/cfg
baselining Repo /devices
baselining Repo /devices/SUNW,ffb@1e,0:ffb0
baselining Repo /devices/pseudo
baselining Repo /devices/pseudo/arp@0:arp
baselining Repo /proc
baselining Repo /proc/0
baselining Repo /proc/0/as
baselining Repo /proc/0/auxv
baselining epo /tmp
```

To provide a high level of security, running Avatar with the check command is to be run by cron periodically. This assures that any unauthorized intrusion can be quickly remedied. The output of the Avatar check command should be E-mailed to an appropriate person for further action.

Sample Avatar Check Command with Output

```
# avatar check / /tmp/baseline
##==> FOO/BAR Site License:##==> VSTKP SITE LICENSE KEY
baseline MD5 = a07e98310d52f1edf9049c7a8594bf96 :
/export/home/roach/.bash_history
file MD5 = c68d06a8bd0001309b6bbab5f2ce4dda :
/export/home/roach/.bash_history
baseline MD5 = 5ff3b5bd6ad0b37a836a4ac89eaa4928 :
/export/home/roach/typescript
file MD5 = d41d8cd98f00b204e9800998ecf8427e :
/export/home/roach/typescript
```

The system administrator may, at his/her discretion, run Avatar with the update, iupdate, or correct command. The Avatar Update command will update the baseline database to the current file and directory specifications automatically. No interaction from the user is required.

Avatar Update Command with Output

```
# avatar update / /tmp/baseline##==> FOO/BAR Site License:##==>
VSTKP SITE LICENSE KEYupdating owner from uid = 1039,
gid = 7 to uid = 0, gid = 7:
/devices/pseudo/pts@0:7updating signature from
```

```
deebbbdbd8663f2b27d88c1e4bd500c4 to
0e6242e8a8f85036121de5d4a66bab0b: /etc/mnttabupdating owner
from uid = 1039, gid = 108 to uid = 0, gid = 1:
/export/home/roach/typescriptupdating permissions from mode = 0100644
to mode = 0100666:
/export/home/roach/typescriptupdating signature from
5ff3b5bd6ad0b37a836a4ac89eaa4928 to
9b6a35bcb46d27621683f0b447948c42: /export/home/roach/typescript
```

Note: Indented lines are continuations of previous line

>Using the Avatar Iupdate, the user is prompted before each entry requiring updating in the baseline database. Only a 'y' response will have the corresponding entry updated.

Avatar Update Command with Output

```
avatar iupdate / /tmp/baseline
##==> FOO/BAR Site License:
##==> VSTKP SITE LICENSE KEY
file signature has changed from 16453c32fbbfd6a1107b7e736c287e73 to
1389cd8c4534bcbc6a863d9f4c6b888e: /export/home/roach/.bash_history
update? [yn] n
file permissions have changed from mode = 0100666 to mode = 0100644:
/export/home/roach/typescript
update? [yn] y
updating permissions from mode = 0100666 to mode = 0100644:
/export/home/roach/typescript
file signature has changed from 9b6a35bcb46d27621683f0b447948c42 to
e1eb511e2509e280a96e42932b92be4e: /export/home/roach/typescript
update? [yn] n
```

Note: Indented lines are continuations of previous line

Using the Avatar Correct command, Avatar attempts to restore the effected file back to the baseline settings. If the contents have changed and the contents are not in the baseline the correction can not be restored.

Avatar Correct Command with Output

```
# avatar correct / /tmp/baseline
##==> FOO/BAR Site License:
##==> VSTKP SITE LICENSE KEY
baseline uid = 1039, gid = 7 : /devices/pseudo/pts@0:8
file uid = 0, gid = 7 : /devices/pseudo/pts@0:8
owner corrected: /devices/pseudo/pts@0:8
baseline mode = 020666 : /devices/pseudo/pts@0:8
file mode = 020620 : /devices/pseudo/pts@0:8
permissions corrected: /devices/pseudo/pts@0:8
baseline MD5 = c5bea925d801e2e850ea8258600e9428 : /etc/mnttab
file MD5 = a6ceffbd21c8836af453f585c8d92ea6 : /etc/mnttab
couldn't get saved file contents: /etc/mnttab
couldn't restore file: /etc/mnttab
baseline MD5 = 14863b7bea894b9fd54abb97f2c0272f :
/export/home/roach/.bash_history
file MD5 = 17dad806778cc7374c388b2e0066b02b :
/export/home/roach/.bash_history
```

```
couldn't get saved file contents: /export/home/roach/.bash_history
couldn't restore file: /export/home/roach/.bash_history
```

Note: Indented lines are continuations of previous line

508 Compliance TestPros TESTED

This site certified 508 Compliant