

Teaching a old utility new tricks.

Updating The Automated Security Enhancement Tool (ASET) to use cryptographically strong checksum and to check files that are commonly found in rootkits. The files commonly found in rootkits are checked against the Solaris Fingerprint Database.

The following utilities will be needed for these enhancements to ASET.

md5 (only required for Solaris versions prior to Solaris 10)

<http://sunsolve.sun.com/md5/md5.tar.Z>

(This file contains md5-sparc and md5-x86)

sfpC.pl (sfpC-v0.5) Solaris Fingerprint Database Companion

<http://www.timwort.org/aset/sfpscripts.zip>

(links to the Solaris Fingerprint Database fail in previous versions of sfpC.pl)

sidekick.sh a script used to collect the file fingerprints. Renamed sfpS.sh for this paper.

<http://www.timwort.org/aset/sfpscripts.zip>

The sfpC.pl uses perl, perl will need to be updated with HTTP::Request, LWP::UserAgent, and HTML::Parser modules. The easiest way to add the modules is to use CPAN. (See <http://www.cpan.org> for module installation information.)

```
# perl -MCPAN -e shell
cpan> install <module>
```

You will need to install all module dependencies as well.

The sfpS.sh (sidekick) utility will need to have three variables configured.

SIDEKICK_MD5 needs to be set to the location of md5, the default is /usr/sbin/md5.

For version of Solaris prior to Solaris 10 copy md5-sparc or md5-x86 to /usr/sbin/md5.

For Solaris 10 you can use the Solaris 10 digest utility. This will require adding a new variable called MD5_OPTIONS, this will be used to pass options to the digest command.

```
MD5_OPTIONS="-va md5"
SIDEKICK_MD5=/usr/bin/digest
```

Search for SIDEKICK_MD5 and add the MD5_OPTIONS to the command that uses the SIDEKICK_MD5 variable.

...

```
for binary in ${FILELIST}
do
    if [ -f ${binary} ]; then
        $SIDEKICK_MD5 $MD5_OPTIONS $binary >> ${OUTPUTFILE}
```

```
fi
done
...
```

The `SIDEKICK_PERL` variable will be set to the location of your `perl` program. For this paper the location of `perl` is `/usr/local/bin/perl`

```
SIDEKICK_PERL=/usr/local/bin/perl
```

The final variable that needs to be configured points to the location of the `sfpC.pl` script, for this paper the location is `/usr/local/scripts`. The variable is `SIDEKICK_SFPC`.

```
SIDEKICK_SFPC=/usr/local/scripts/sfpC.pl
```

Now you can test `sfpS.sh` (`sidekick`) and `sfpC.pl`.

```
# ./sfpS.sh -r
Searching for files commonly found in rootkits.
The output has been saved to /tmp/rootkitfiles-md5.20070829134422.
Using sfpC to process MD5 signatures from file, /tmp/rootkitfiles-
md5.20070829134422.
```

```
...
```

```
74d8bfe0b36c4a9ad59a158348dc4146 - - 1 match(es)
```

```
canonical-path: /usr/bin/nawk
package: SUNWesu
version: 11.10.0,REV=2005.01.21.15.53
architecture: sparc
```

```
source: Solaris 10/SPARC
```

```
patch: 118815-04
```

ASET lives in the `/usr/aset` directory, the `aset` script uses various scripts and utilities found in `/usr/aset` or in directories under `/usr/aset` such as `/usr/aset/tasks`.

The `asetenv` files is used to configure variables that are used by `aset` and the tasks `aset` runs.

The first change will allow ASET to use `md5` rather than the `sum` command when generating checksum values for files.

One of the tasks ran by the `aset` script is the `cklist` (checklist) task, this task creates a list of files, defined in the `asetenv` file. The task creates checksum values for each file, on subsequent runs of `aset` the checksum for each file is checked against a checksum generated by the current task run. The changes, if any, are reported in the `cklist.rpt`.

1. First the `asetenv` files will be modified to point to the `md5` or `digest` utilities. The `asetenv` files contains two lines that will be modified and one line will be added. Open the `asetenv` file for editing, add the following line at the end of the long list of variable definitions.

```
MD5=/usr/bin/digest
```

The resulting file will look something like:

```
...
CRONTAB=/bin/crontab
TOUCH=/bin/touch
MD5=/usr/bin/digest
```

(For Solaris releases prior to Solaris 10 use `MD5=/usr/sbin/md5`)

2. Next find the `sysutils` variable and the `progs` variable and add the `MD5` variable to each line.

```
sysutils="... CRONTAB TOUCH MD5" <== MD5 added
progs="... $IS_READABLE $HOMEDIR $EXPR $MD5" <== $MD5 added
```

The `aset` script uses the `/usr/aset/tasks/cklist` script, this script calls the `/usr/aset/util/addcksum` script to generate the initial list of files and the corresponding checksum and stores them in the `/usr/aset/masters` directory, subsequent runs of `aset` causes a new checksum list to be generated called a snapshot, the list in the masters directory is then compared to the snapshot and the differences are reported.

3. The `/usr/aset/util/addcksum` script needs to be updated to use the `MD5` utility defined in the `/usr/aset/asetenv` file. You will replace the current `cksum` line:

```
cksum=`$SUM $filename | $SED "s/^\([0-9]* [0-9]*\) .*/\1/"`
```

with the following line (Solaris 10):

```
cksum=`$MD5 $MD5_OPTIONS $filename | $AWK '{ print $4 }'`
```

or (prior to Solaris 10):

```
cksum=`$MD5 $filename | $AWK '{ print $4 }'`
```

If you are using the `digest` command with Solaris 10 you also need to add the following line near the top of the `/usr/aset/util/addcksum` script:

```
MD5_OPTIONS="-va md5"
```

4. To test the changes first remove any existing master cklist files.

```
# rm /usr/aset/masters/cklist.*
```

Now run the aset utility.

```
# cd /usr/aset
```

```
# ./aset low
```

```
===== ASET Execution Log =====
```

```
ASET running at security level low
```

```
Machine = chaos; Current time = 0829_15:36
```

```
aset: Using /usr/aset as working directory
```

```
Executing task list ...
```

```
    firewall
```

```
    env
```

```
    sysconf
```

```
    usrgrp
```

```
    tune
```

```
    cklist
```

```
    eeprom
```

All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:

```
    /usr/aset/util/taskstat      [aset_dir]
```

where aset_dir is ASET's operating directory, currently=/usr/aset.

When the tasks complete, the reports can be found in:

```
    /usr/aset/reports/latest/*.rpt
```

You can view them by:

```
    more /usr/aset/reports/latest/*.rpt
```

When the `cklist` task completes examine the the `/usr/aset/masters/cklist.low` file and verify the contents contains the md5 signatures.

```
-rwx----- 1 root bin 1928 Jan 21 16:27 2005 /usr/aset/tasks/cklist
1353b1e0e2ad8059efc59b4bc3248b5a
-rwx----- 1 root bin 1204 Jan 21 16:27 2005
/usr/aset/tasks/create_cklist 840b f0d2b35ec5184f15f3af133fc8ca
-rwx----- 1 root bin 1166 Jan 21 16:27 2005 /usr/aset/tasks/eeprom
67d3943f115e3099f3878741a61600bd
-rwx----- 1 root bin 4091 Jan 21 16:27 2005 /usr/aset/tasks/env
1077919c1bf32e01159543f566a14867
-rwx----- 1 root bin 3297 Jan 21 16:27 2005 /usr/aset/tasks/firewall
01d1042cfcb51674a16819691ecb3559
...
```

Only regular files will have signatures.

Below is the a example `/usr/aset/reports/latest/cklist.rpt` that shows a change to one of the checked files.

```
*** Begin Checklist Task ***
```

```
... Checklist snapshot is being created. Wait ...
... Checklist snapshot created.
```

Here are the differences in the checklist.

```
< lines are from the master;
> lines are from the current snapshot
```

```
249c249
```

```
< -rw-r--r-- 1 root sys 362 Jul 2 13:50 2007 /etc/vfstab
4446128d2b33c470c1281613ef243e27
```

```
---
```

```
> -rw-r--r-- 1 root sys 393 Aug 29 15:17 2007 /etc/vfstab
6a125fbb4bcbb1ae57149651f9146f35
```

```
*** End Checklist Task ***
```

One drawback is that ASET stores the original checklist on the local file system so should the system be compromised it is trivial for the attacker to change the master checklist file. This means the administrator must store the master checklist on read only media or copy a known good `cklist.<level>` file to the `/usr/aset/masters` directory each time ASET is used.

The next example is one way this issue might be addressed but applies only to Solaris 10.

Rather than use the `digest` command you will use the `mac` (Message Authentication Code) command. The `mac` command also generates a checksum or signature however a secret key is used in the signature generation process. To check the file signature the file must remain unchanged and the same key must be used.

ASET is not designed to be interactive so some method must be devised to supply a key to the `mac` command. The method used here has the location of the key supplied via an environment variable, the value of the variable can be supplied when the `aset` command is executed or the key location can be coded in the `/usr/aset/addcksum` script.

Again you must edit the `/usr/aset/asetenv` file. The changes are nearly the same as earlier except MD5 is replaced with a variable called MAC.

1. Open the `asetenv` file for editing, add the following line at the end of the long list of variable definitions.

```
MAC=/usr/bin/mac
```

The resulting file will look something like:

```
...
CRONTAB=/bin/crontab
TOUCH=/bin/touch
MAC=/usr/bin/mac
```

2. Next find the `sysutils` variable and the `progs` variable and add the MAC variable to each line.

```
sysutils="... CRONTAB TOUCH MAC" <== MAC added
progs="... $IS_READABLE $HOMEDIR $EXPR $MAC" <== $MAC added
```

3. Edit the `/usr/aset/util/addcksum` script and add the following line:

```
MAC_OPTIONS="-va sha256_hmac -k $KEYFILE"
```

(If you want to define the `$KEYFILE` variable in the script add a line before the `MAC_OPTIONS=` line. For example `KEYFILE=/var/run/keyfile`)

4. Test the configuration.

Create a key:

```
# dd if=/dev/urandom of=/var/run/keyfile bs=512 count=1
```

Set the `KEYFILE` variable for your environment.

```
# KEYFILE=/var/run/keyfile;export KEYFILE
# rm /usr/aset/masters/cklist.*
# cd /usr/aset
# ./aset low
```

Examine the `/usr/aset/masters/cklist.low` file to verify signatures have been created.

```
# more cklist.low
-rwx----- 1 root bin 1928 Jan 21 16:27 2005 /usr/aset/tasks/cklist
8c2a84196f0e295de20ee4f612325eed046a8b45d5d0830cb45c2f2525007ef9
-rwx----- 1 root bin 1204 Jan 21 16:27 2005
/usr/aset/tasks/create_cklist
7121c79c69fb3b3422869f4a664a2c8ff9acc5024ea4f16184ac71fe016c057c
-rwx----- 1 root bin 1166 Jan 21 16:27 2005 /usr/aset/tasks/EEPROM
536d3ba97f76104d4810db9391f528c9624bfe8264f823aace3093b560f5fad1
-rwx----- 1 root bin 4091 Jan 21 16:27 2005 /usr/aset/tasks/env
277dff1ec8660f4ce7ac4b81e4d409409807ae9060704258ed07d0f6661aa235
...
```

The issue with the previous configuration is that it does not work well when automating ASET using the `cron` facility. The key file should not be stored on the system, if it is stored on the system it should be encrypted.

The Solaris Fingerprint Database is a repository of md5 signatures for files that shipped with Solaris OS releases. Users can submit lists of signatures to be checked against the signature stored in the repository. The Solaris Fingerprint Database will return information about each file that matches a submitted signature or return the string "0 Matches" if the signature submitted is not found in the database.

The Solaris Fingerprint Database can be accessed at <http://sunsolve.sun.com/fileFingerprints.do> and signatures can be submitted via a web browser. The `sfpC.pl` script allows the user to submit a set of signatures using the command line. The `sidekick` script (`sfpS.sh`) is used to collect signatures and pass the signatures to the `sfpC.pl` script.

1. Edit the `sfpS.sh` script to define the location of the output file that will contain the signatures.

Find the following entry in the `sfpS.sh` script:

```
while getopts rR:uUgGsaSh v; do
```

```
case ${v} in
```

```
R)
```

```
ROOTDIR="${OPTARG}"
```

```
echo "Setting root directory to ${ROOTDIR}."
```

```
;;
```

```

r)
    OUTPUTFILE="rootkitfiles-md5"
    FUNCTION="RootKit"
    echo "Searching for files commonly found in rootkits."
    ;;

```

Change the “r” case OUTPUTFILE variable to the location to be used.

```
while getopts rR:uUgGsaSh v; do
```

```
    case ${v} in
```

```

R)
    ROOTDIR="${OPTARG}"
    echo "Setting root directory to ${ROOTDIR}."
    ;;

```

```

r)
    OUTPUTFILE="/var/run/rootkitfiles-md5"
    FUNCTION="RootKit"
    echo "Searching for files commonly found in rootkits."
    ;;

```

2. Create the ASET task script.

The following script named `rkitchk` will be placed in the `/usr/aset/tasks` directory, the script will be owned by user `root` and group `bin` with permissions set to 700.

```

#!/bin/sh
#
# Root Kit Check using Sidekick and sfpC.pl
# and compare the hash's with the Sun Fingerprint Database
#
echo "*** Begin Root Kit Files Check ***"
echo ""

$SFPS -r | grep " 0 match"

echo "Any entry above this line indicates a 0 match and"

```

```
echo "the file(s) listed failed the cryptographic checksum"
echo "comparison. YOU MAY HAVE BEEN ROOTED!"
```

```
$RM /var/run/rootkit*
```

```
echo ""
echo "*** End Root Kit Files Check ***"
```

3. Edit the `/usr/aset/asetenv` file to add the new task and enable the `sfpS.sh` script.

Edit the `TASKS` variable to add the new task:

```
TASKS="firewall env sysconf usrgrp tune cklist eeprom rkitchk"
```

Add a variable `SFPS` to point to the `sfpS.sh` script.

...

```
TOUCH=/bin/touch
```

```
MAC=/usr/bin/mac
```

```
SFPS=/usr/local/scripts/sfpS.sh
```

Edit the `sysutils` and `progs` variables to include the `SFPS` variable.

```
sysutils="... CRONTAB TOUCH MAC SFPS"
```

```
progs="... $EXPR $MAC $SFPS"
```

4. Test the configuration.

Use `snoop` to verify the connection to `sunsolve.sun.com` and the Fingerprint Database.

You should see the initial connection from your system and the results being returned from `Sunsolve`.

```
<your system> -> sunsolve.Sun.COM HTTP POST /fileFingerprints.do
HTTP/1.1
```

```
<your system> -> sunsolve.Sun.COM HTTP
md5list=%0Amd5+(%2Fbin%2Fdate)+%3D+00a7737d352eca7e4e7e7a7434bdc9d5%0
A%0Amd5+(%2F
```

```
<your system> -> sunsolve.Sun.COM HTTP
137851c1407fd9ea20785bc0%0A%0Amd5+(%2Fusr%2Fucb%2Fsparcv9%2Fps)+%3D+5
e1fa7c274041
```

```
<your system> -> sunsolve.Sun.COM HTTP
pg4%2Fbin%2Fgrep)+%3D+200ec5590463d5707f2b73d078c2efc1%0A%0Amd5+(%2Fu
sr%2Fsbin%2F
```

...

```
sunsolve.Sun.COM -> <your system> HTTP (body)
```

```
sunsolve.Sun.COM -> <your system> HTTP ucb/du</TT>
```

```

sunsolve.Sun.COM -> <your system>      HTTP (body)
sunsolve.Sun.COM -> <your system>      HTTP canonical-path:
<TT>/usr/ucb/du</TT>

sunsolve.Sun.COM -> <your system>      HTTP (body)
sunsolve.Sun.COM -> <your system>      HTTP 998.09.01.04.16</TT>

sunsolve.Sun.COM -> <your system>      HTTP          <LI> canonical-
path: <TT>/usr/ucb/du</TT>

```

You can also monitor the `/var/run/directory` for the creation of the signature file.

After verifying the configuration the `sfpC.pl` and `sfpS.sh` scripts should be protected. For this example the scripts live in the `/usr/local/scripts` directory. The two scripts are owned by the root user and have the permissions set to mode 700.

Now both scripts are added to the `tune.*` files in `/usr/aset/masters` directory. On each ASET run the permission will automatically adjust if they are found to be less secure than the setting listed in the `tune.*` files.

```

# echo "/usr/local/scripts/sfpC.pl 0700 root root file" >>
/usr/aset/masters/tune.low
# echo "/usr/local/scripts/sfpC.pl 0700 root root file" >>
/usr/aset/masters/tune.med
# echo "/usr/local/scripts/sfpC.pl 0700 root root file" >>
/usr/aset/masters/tune.high
# echo "/usr/local/scripts/sfpS.sh 0700 root root file" >>
/usr/aset/masters/tune.low
# echo "/usr/local/scripts/sfpS.sh 0700 root root file" >>
/usr/aset/masters/tune.med
# echo "/usr/local/scripts/sfpS.sh 0700 root root file" >>
/usr/aset/masters/tune.high

```

ASET is not a complete security solution. ASET is easy to configure and customize for the administrator so it is also easy to configure and customize for an attacker, however, it does provide another layer of security checks making the process of hiding an attacker's footprint more difficult.