Sécurité des logiciels

Quelques notions de sécurité système

Samuel Thibault <samuel.thibault@u-bordeaux.fr>

CC-BY-NC-SA

Permissions

Unix users

uid (User identity)

- Integer that represents a user
- By convention, 0 is for root

gid (Group identity)

- Integer that represents a group of users

```
$ id
uid=16595(sathibau) gid=1111(enseignant)
groups=1111(enseignant),1113(runtime),
1203(employee),1027(researcher),1211(teacher), …
$ id aguermou
uid=14925(aguermou) gid=1111(enseignant)
groups=1111(enseignant),
1203(employee),1027(researcher),1211(teacher), …
```

Unix permissions

Permissions : **rwx**

- r : Read (4)
- w : Write (2)
- x : eXecute (1)

Permission triplet : **ugo**

- u : User
- g : Group
- o : Other

```
u+rwx,g+rx,o+rx    rwx r-x r-x    755
u+rw,g+r,o+r       rw- r-- r--    644
u+rw,g+r           rw- r-- ---    640

$ ls -ldn ~sathibau
drwxr-x--x 16595 1111 /net/cremi/sathibau
```

Changing permissions

```
chmod 755 ~/tmp
chmod g+rx,o+rx ~/tmp

chmod 700 ~/secret
chmod g-rwx,o-rwx ~/secret

Also, see ACL (Access Control List)
```

setuid
(Set User ID)

setuid

Programs have their own uid, inherited from uid of parent

```
sshd(sathibau)
    \- bash(sathibau)
          \- ls(sathibau)
```

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

```
sshd(root)
    \- sshd(sathibau)
           \- bash(sathibau)
                  \- ls(sathibau)
```

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

`sshd(root)`

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

```
sshd(root)
    \- sshd(root)
```

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

```
sshd(root)
    \- sshd(root)   setuid(16595)
```

`setuid()` changes the current uid of the calling process

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

```
sshd(root)
    \- sshd(sathibau)
```

`setuid()` changes the current uid of the calling process

setuid

Programs have their own uid, inherited from uid of parent, except when they change it

```
sshd(root)
    \- sshd(sathibau)
           \- bash(sathibau)
```

`setuid()` changes the current uid of the calling process

setuid

Programs have their own uid, inherited from uid of parent, except **when the program is setuid**

```
sshd(root)
    \- sshd(sathibau)
           \- bash(sathibau)
                  \- chsh(root)
```

```
$ ls -l /bin/chsh
-rwsr-xr-x root root /bin/chsh
```

Yes, this is a terrifying design.
They need to be perfectly sane.
We'll see various ways **not** to be perfectly sane.

setuid

- **real** uid (**`getuid()`**): the uid that started the program
- **effective** uid (**`geteuid()`**) : the uid currently set for the program
- **saved** uid (**`getresuid()`**) : an uid saved for later use

```
\- bash(sathibau)
        \- at(root)
```

setuid

- **real** uid (`getuid()`): the uid that started the program
- **effective** uid (`geteuid()`) : the uid currently set for the program
- **saved** uid (`getresuid()`) : an uid saved for later use

```
\- bash(sathibau)
        \- at(root) seteuid(getuid())
```

setuid

- **real** uid (**`getuid()`**): the uid that started the program
- **effective** uid (**`geteuid()`**) : the uid currently set for the program
- **saved** uid (**`getresuid()`**) : an uid saved for later use

```
\- bash(sathibau)
        \- at(sathibau)
```

setuid

- **real** uid (**`getuid()`**): the uid that started the program
- **effective** uid (**`geteuid()`**) : the uid currently set for the program
- **saved** uid (**`getresuid()`**) : an uid saved for later use

```
\- bash(sathibau)
        \- at(sathibau)
```

Can use the saved uid to make effective alternate between the two.

PATH,
LD_LIBRARY_PATH,
LD_PRELOAD

PATH

```
$ which ls
/bin/ls

$ echo $PATH
/usr/local/bin:/usr/bin:/bin
```

Can be used to trick setuid programs…

LD_LIBRARY_PATH

```
$ ldd /bin/ls
[…]
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6

$ mkdir ~/lib
$ cp /lib/x86_64-linux-gnu/libc.so.6 ~/lib/
$ export LD_LIBRARY_PATH=~/lib
$ ldd /bin/ls
[…]
libc.so.6 => /net/cremi/sathibau/lib/libc.so.6
```

Can be used to override system library

But **cannot** be used to trick setuid programs…

« safe » execution startup that ignores LD_LIBRARY_PATH

LD_PRELOAD

```
$ LD_PRELOAD=~/lib/libm.so ldd /usr/bin/ls
[…]
/net/cremi/sathibau/lib/libm.so
```

Can be used to override symbols

But **cannot** be used to trick setuid programs…

« safe » execution startup that ignores LD_PRELOAD

`*PATH=.` considered harmful

```
$ PATH=.:$PATH
$ LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

Is a terribly bad idea...

```
$ cd /tmp
$ ls
```

Would run whatever `ls` program that anybody would have left there...

Races

Races

- TOCTOU : Time Of Check to Time Of Use

```
char *file = argv[1];
stat(file, &st);
if (st.st_uid == getuid()) {
    // User file, can safely open
    int fd = open(file, O_RDONLY);
    read(…); printf(…);

$ ( while true ; do
ln -sf ~/myfile /tmp/hack
ln -sf /etc/shadow /tmp/hack
done ) &
$ while ! /bin/suid-victim /tmp/hack; do : ; done
```

Races

- TOCTOU : Time Of Check to Time Of Use
- **Atomicity** between check and use

```
char *file = argv[1];
int fd = open(file, O_RDONLY);
fstat(fd, &st);
if (st.st_uid == getuid()) {
    // User file, can safely read
    read(…); printf(…);
```

Races

- TOCTOU : Time Of Check to Time Of Use

```
char *path = argv[1];
char *dir = dirname(strdup(path));


stat(dir, &st);
if (st.st_uid == getuid()) {
    // User file, can safely open
    int fd = open(path, O_RDONLY);
    read(…); printf(…);
}
```

Races

- TOCTOU : Time Of Check to Time Of Use
- **Atomicity** between accessing directory and accessing its files

```
char *path = argv[1];
char *dir = dirname(strdup(path));
char *file = basename(path);
int dfd = open(dir, O_PATH) ;
fstat(dfd, &st);
if (st.st_uid == getuid()) {
    // User file, can safely open
    int fd = openat(dfd, file, O_RDONLY);
    read(…); printf(…);
}
```