“Process user” control -- su, sudo, and SUID

David Morgan

Processes and users

- running processes are associated with user(s)
  - real user/UID -- user ID of the process’ parent process
  - effective user/UID -- determines resource access
  - real=”by whom” effective=“as whom” process runs
- process’s real and effective UIDs are same, usually
- login shell’s UIDs are (both) the one per the login
- check with getuid() and geteuid()
Process gets from caller/parent, gives to called/child

```
#include <unistd.h>
#include <sys/types.h>

uid_t getuid(void);
uid_t geteuid(void);
```

description
getuid() returns the real user ID of the current process.
geteuid() returns the effective user ID of the current process.

Diagnostic UID revealer program

```
[root@CHANG ~]# cat ids.c
/
/*
 * ids.c - Print UIDs and GIDs
 */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(void)
{
    printf("Real user ID: %d\n", getuid());
    printf("Effective user ID: %d\n", geteuid());
    printf("Real group ID: %d\n", getgid());
    printf("Effective group ID: %d\n", getegid());
    exit(EXIT_SUCCESS);
}
```

another revealer:       ps -eo pid,ruid,euid,command
ps -eo pid,ruid,euid,command

Real UID

Effective UID

<table>
<thead>
<tr>
<th>PID</th>
<th>RUID</th>
<th>EUID</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>11387</td>
<td>0</td>
<td>0</td>
<td>sshd: root@pts/0</td>
</tr>
<tr>
<td>11393</td>
<td>500</td>
<td>500</td>
<td>-bash</td>
</tr>
<tr>
<td>11964</td>
<td>500</td>
<td>500</td>
<td>xinit -- :1</td>
</tr>
<tr>
<td>11965</td>
<td>500</td>
<td>0</td>
<td>xterm -geometry 41x1 -n login</td>
</tr>
<tr>
<td>12000</td>
<td>500</td>
<td>500 /bin/sh /usr/lib64/firefox-1.5.0.7/firefox -U</td>
<td></td>
</tr>
<tr>
<td>12014</td>
<td>500</td>
<td>500 /bin/sh /usr/lib64/firefox-1.5.0.7/run-mozilla</td>
<td></td>
</tr>
<tr>
<td>12019</td>
<td>500</td>
<td>500 /usr/lib64/firefox-1.5.0.7/firefox-bin -U -locale en-US</td>
<td></td>
</tr>
<tr>
<td>12242</td>
<td>500</td>
<td>500 /usr/libexec/gconfd-2 11</td>
<td></td>
</tr>
<tr>
<td>12237</td>
<td>501</td>
<td>503</td>
<td>-bash</td>
</tr>
<tr>
<td>12263</td>
<td>501</td>
<td>0</td>
<td>passwd</td>
</tr>
<tr>
<td>12283</td>
<td>0</td>
<td>0</td>
<td>[pdfuskh]</td>
</tr>
<tr>
<td>12387</td>
<td>0</td>
<td>0</td>
<td>[pdfuskh]</td>
</tr>
<tr>
<td>12290</td>
<td>0</td>
<td>0</td>
<td>ps -eo pid,ruid,euid,command</td>
</tr>
<tr>
<td>12293</td>
<td>0</td>
<td>0</td>
<td>tail -l</td>
</tr>
</tbody>
</table>

3 active users, 0 500 503
2 processes have effective UID different from real UID

Controlling a process’s UIDs

- su
- sudo
- SUID
su syntax

```
su < -c command > <user>
```

- defaults
  - omit user: root
  - omit command: bash
- password prompt: for other user’s password, not yours

Run “ids”

Q: where does it pick up its UIDs from?
A: inherits them

- run it from 2 diff logins
  - reflects the logins’ UIDs
- su to some other user, then run it
  - reflects the other user’s UIDs
- run fork9a mini-shell via su as another user, then run ids from fork9a
  - fork9a reflects the UIDs of the launching shell
  - ids in turn reflects those of fork9a
sudo – secure solution

- lets certain user(s) run certain program(s) as another user
- user runs program indirectly under sudo’s control: `sudo <targetprogram>`
- sudo configuration defines who can run what as whom

sudo syntax

`sudo <–u user> command`

- defaults
  - omit user: root
  - omit command: *not optional*
- password prompt: for *your* password, not other user’s (you don’t know who that is)
sudo config file: /etc/sudoers

- privilege specifications
- other specifications
  - User aliases – named groups of “by” users
  - Runas aliases – named groups of “as” users
  - Cmd aliases – named groups of commands

What?

- What is a “‘by’ user” ?? an “‘as’ user” ??
- default: command runs as whoever launched it
- sudo purpose: let a command launched by one user run as another
  - “by” user is the one who launches the command
  - “as” user is the one the command runs as, as if he had actually launched it (even though he didn’t)
sudoers privilege specifications

<who by>   <where>=(<who as>)   <what>

SUID – exception to the rule

- SUID – a permission characteristic of files
- changes the effective UID of file’s process when run
  - from UID of user who runs the program
  - to UID of user who “owns” the file
SUID – exception to the rule

- WHAT – “When a SUID file is run, the process involved takes on an effective UID that is the same as the owner of the file.”
- WHY – “Sometimes, unprivileged users must be able to accomplish tasks that require privileges.
- EXAMPLES – passwd, mail

passwd uses SUID

passwd program’s executable is SUID therefore runs as whichever user is its owner

which in the case of password database /etc/shadow, since it can be read by its owner who is root, enables the passwd program to read /etc/shadow…

…which lets users change their own passwords
Applying SUID to a file

```
[root@clay ~]# ls -l testfile
-rwxr-xr-x 1 root root 4 Jan 12 11:09 testfile
[root@clay ~]# chmod u+s testfile
[root@clay ~]# ls -l testfile
-rwxr-xr-x 1 root root 4 Jan 12 11:09 testfile
[root@clay ~]# whoami
ids
```

Effect of SUID

ids program reports as whom it is running

- run in this root shell, ids runs as root (gets it from the shell)
- but if its executable is SUID and david owns it, ids runs as david (gets it from the executable)
SUID shell scripts

- BAD
- DON’T
- Security flaw – launches SUID shell to run script

- most modern unix’s now ignore SUID on a script