

- 1. Last time
 - 2. Protection and security in Unix
 - Intro
 - Setuid
 - TOCTTOU
 - Other thoughts
-

2. Protection and security in Unix

A. Intro

UIDs, GIDs (user id, group id)

A process has one user id and one or more group ids

Files and directories are access-controlled

- Saw this in lab 2 (ls)
- System stores w/ each file who owns it
- Where is the info stored?

ls -l
file

rw-rw----

Special user: uid 0, called root, treated by kernel as the administrator.

(h) | || permissions: can

UID 0 (root) has all permissions

read any file, do anything.

certain things only root can do:

- set clock
- halt machine
- mount filesystems
- change process's user or group id

B. setuid

ex: how do users change their password?

\$ passwd

/etc/passwd
/etc/shadow

A prog. can be "setuid"

\$ ls

real uid: mwatfish
effective uid: mwatfish

\$ passwd

real uid: mwatfish
effective uid: root

\$ ls -l /sbin/passwd

passwd root root --- rws

^{binary} "su": change to new userid if correct passwd is typed.

Example attacks

(a) attacker in an attacker binary eff uid: <eviluser>
real uid: <eviluser>

```
close(2);  
exec("/usr/bin/passwd");
```

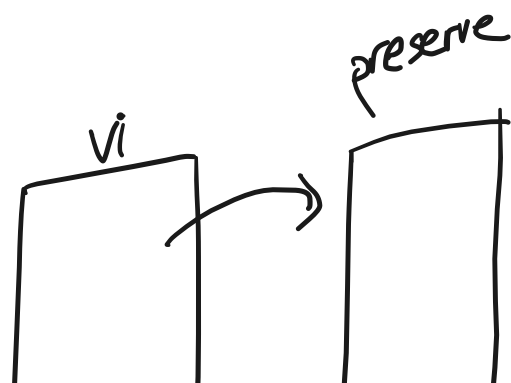
passwd:

```
main()  
{  
  2 fd = open("/etc/passwd", ...)  
  :  
  :  
  2 fprintf(stderr, "Err msg\n");  
}
```

real uid: <eviluser>
eff uid: 0 <root>

WR

(b) old days: "preserve"
setuid root





attacker redefines IFS = '/',
and runs vi in that environment.

attacker:
create "bin", which does:

```
cp /bin/sh ./attack  
chown root attack  
chmod 4755 attack
```

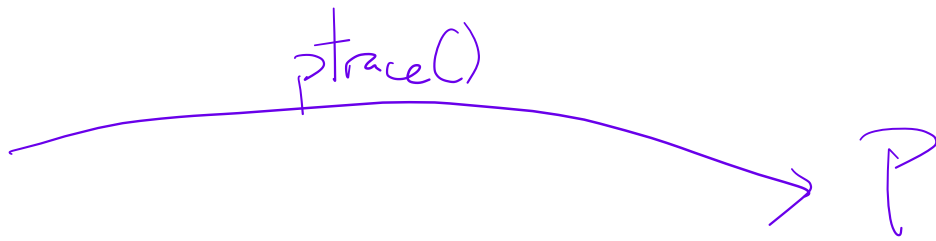
define IFS back

preserve:
system("/bin/mail");
parsed as "bin mail"

ptrace()

Attacker

setuid prog: P
root



TOCTTOU

setuid program: P ^{etc/shadow} <logfile>

⋮

fd = open(logfile, ^{user input} O_CREAT | O_WRONLY | O_TRUNC, 0666)

access(): check whether the real id, not effective id, is allowed to access the file.

if (access (logfile, W_OK) < 0)
return ERROR;

fd = open (logfile, ...) /* as above */

attacker runs "P /tmp/X"

P

attacker
creat ("/tmp/X");

time of check

checks access ("/tmp/X") → OK

unlink ("/tmp/X");

symlink ("/etc/passwd", "/tmp/X");

attack fails

~~access ("/tmp/X", W_OK)~~

fd =
open ("/tmp/X", O_TRUNC |
O_WRONLY);

time of use

fd represents /etc/passwd
