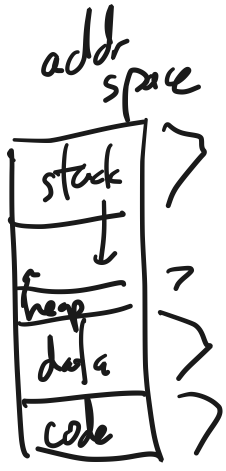
1. Last time
2. Stack smashing, cont'd
3. Protection and security in Unix
   - Intro
   - setuid
   - TOCTTOU (= "time of check to time of use")
   - Other thoughts

---

2. arms race

defenses: W ^ X

response: ROP



defenses: hide the binary

response: BROP

defenses: use ASLR

response: BROP :

defenses: use canary values

response: BROP

BROP:

⇒ Bittau et al.
IEEE S&P
"Hacking Blind"

---

3. Protection and security in Unix

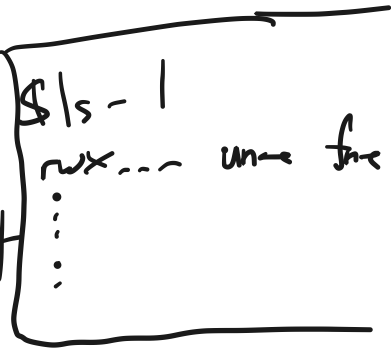
A. Intro

mwalfish ≡ 357(?)
$ echo ..
$ cat

UIDs, GIDs (user ID, group IDs)

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?



Special user: uid 0, called root, treated by kernel as the administrator
- UID 0 (root) has all permissions: can read any file, do anything

Certain things only root can do:
- mounting file systems
- set clocking
- halting machine
- change process's user or group id

B. setuid

ex: how do you users change their password?

$ passwd

/etc/passwd
/etc/shadow

A prog. can be "setuid"

$ ls

real uid: mwalfish
effective uid: mwalfish

$ passwd real uid: mwalfish
effective uid: root

Ex: passwd
su : changes to new user ID if correct passwd is typed

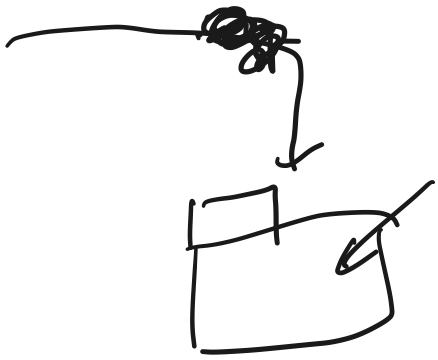
Example attacks

- close fd 2
- exec passwd()

```
close(2);
exec("/sbin/passwd");
```

```
passwd()
{ fd = open("/etc/passwd", ...)
  if (fd == -1)
    fprintf(stderr, "Err msg");
}
```

- Old days: preserve
setuid root



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

vi:

↳ fork(); exec("/usr/lib/preserve")

IFS: internal field separator: " "

↳ '/'

invoke "bin": cp /bin/sh ./attack
chown root attack
chmod 4755 attack

C. TOCTTOU

$ prog X

Ex: prog setuid-root
it needs to log to some file specified by the invoker of the prog.

prog:

argv[1]

fd = open (logfile, O_CREAT | O_WRONLY | O_TRUNC, 0666);

real UID: attacker
→ effective UID: root

$ prog /etc/passwd

goal: open succeeds only if real UID has permission

```
if (access (logfile, W_OK) < 0) // check real user id
    return ERROR;
fd = open (logfile, O_WRONLY | O_TRUNC);
write (fd, buffer, sizeof buffer);
```

Bad

fd = open (logfile, O_CREAT | ----) // check effective user id

time ↓

prog

attacker

creat ("/tmp/X");

check access ("/tmp/X") → OK

unlink ("/tmp/X");

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

logfile

open ("/tmp/X", O_CREAT ---)

↳ b/c root does have privilege over /etc/passwd, this "succeeds" and trashes the password file.