CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 26, Thur Apr 11 2024

Security & related stuff

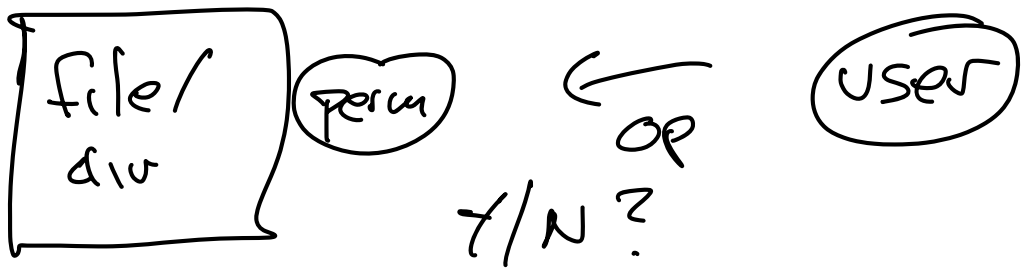
{actors} . {ops} {objects}

→ authentication → pub. key encryption → signatures

access control:

user . op . file/directory/... ⇒ yes/no

permissions



Access control matrix

unix file: R W X (execute)

directory: list ↑ modify ↑ traverse

ls /dir1/dir2
X R

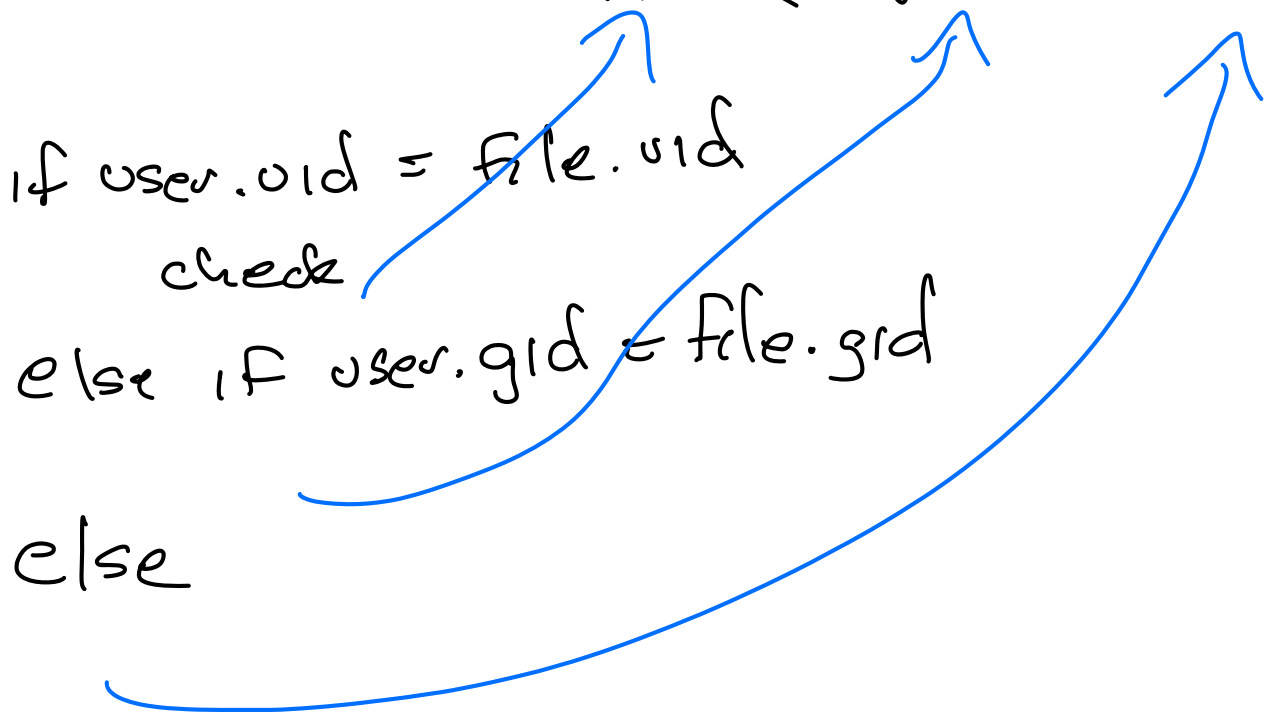
cat /dir1/dir2/file
X X R

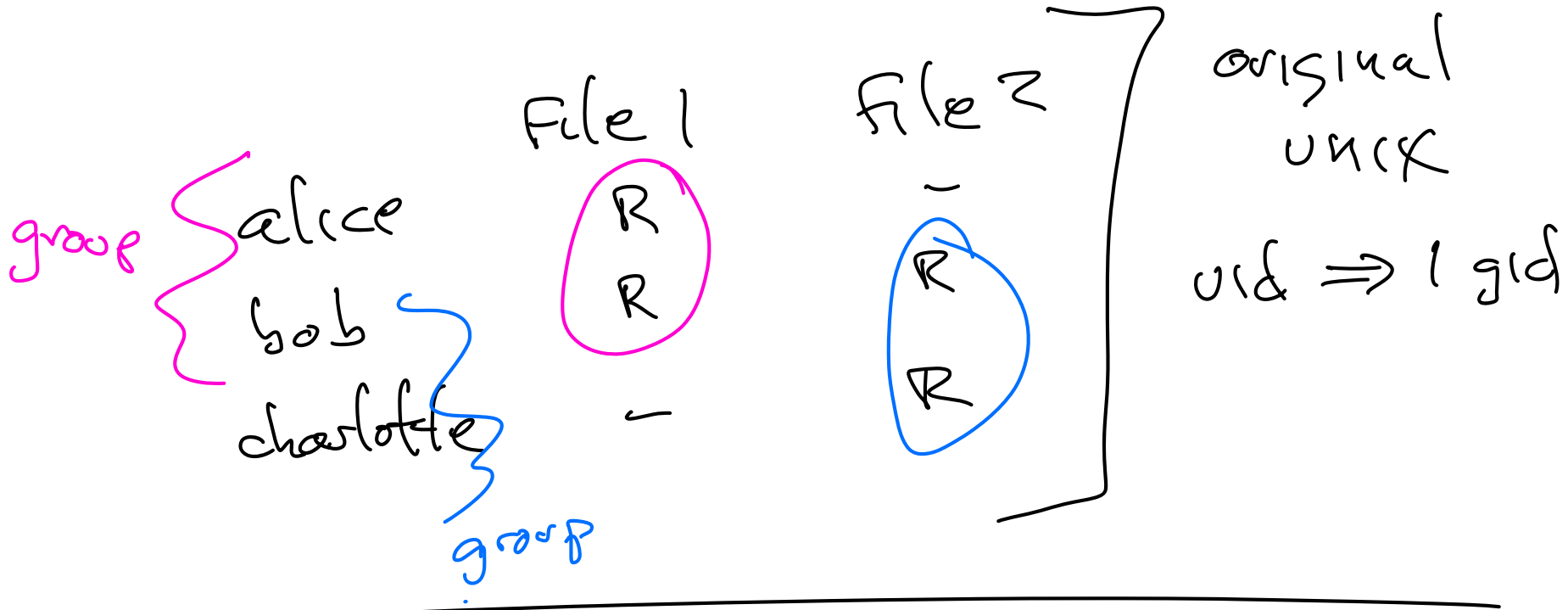
file owner: user id, group id (numeric, text in /etc/passwd, group)

user is: user id + group id

file permissions: user group other (world)

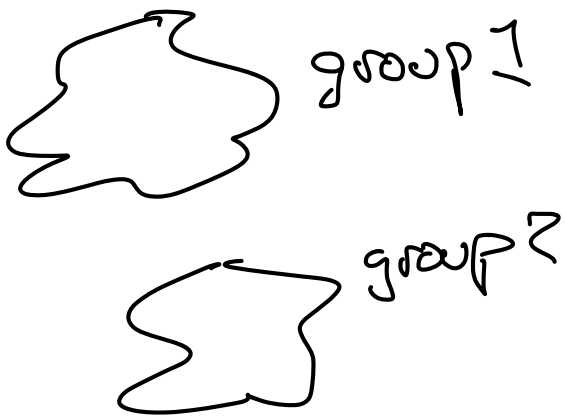
RWX RWX RWX





newer unix:

user → {groups}
≤ 32



id

	file1 (alice.G1)	file2 (bob.G2)
alice G1	R	—
bob G1, G2	R	R
charlotte G2	—	R
	... G=R--	G=R--

Permissions:

user match	group match	no match
RWX	RWX	RWX

file ownership
+
file permissions
+
user identity
} = allowed ops

Alternative ways of specifying access

access control list

id . action → accept
id . action → deny

or

file1 alice, R → ok
bob, R → ok
* deny

file2 bob, R → ok
charlotte, R → OK
* deny

file3 joe, * → deny
prof1 ** → ok
S = faculty → deny
* → ok

Role-based access control

Attacks

stack overflow

1988 internet worm

fd = accept()

dup2(fd, 0) dup2(fd, 1)

close fd

fork ~~~~ exec

inetd

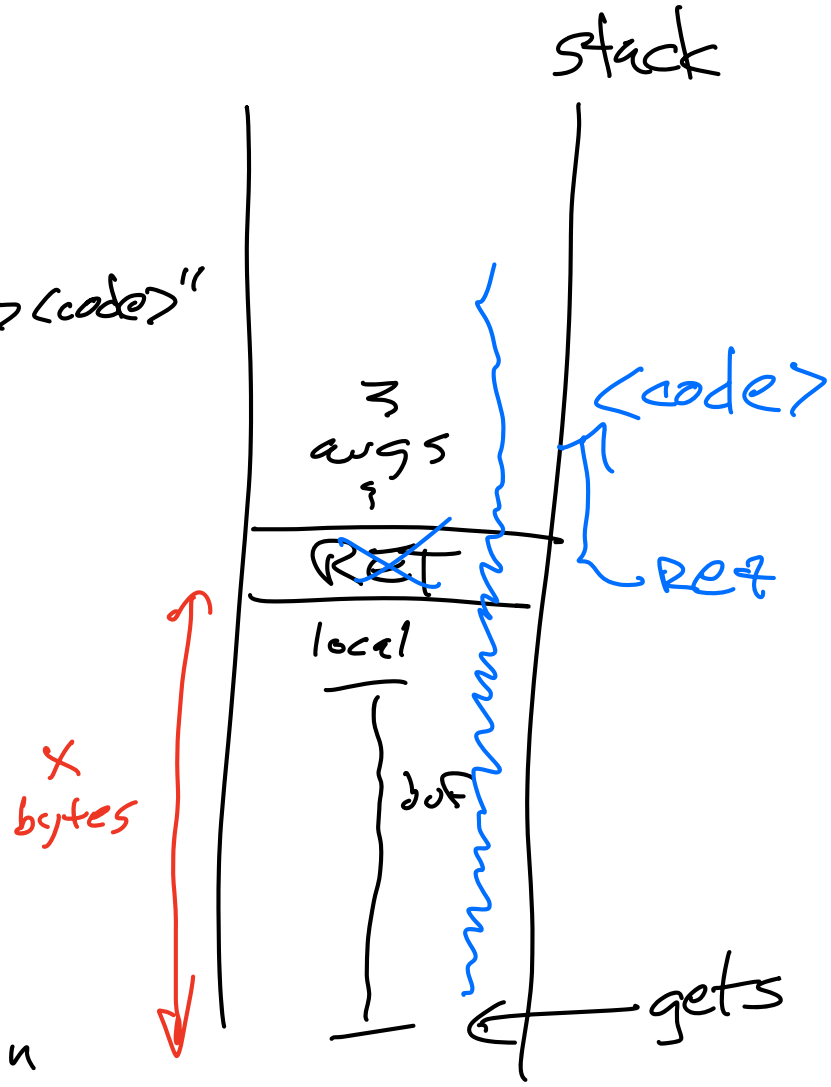
```
{
  char buf[512];
  gets(buf);
```

fgets(buf, sizeof(buf), stdin)

"512 bytes of junk ... <fake ret> <code>"

defenses

- memory safe languages
- execute permissions (on memory pages)
- address layout randomization



push ...
system("/bin/sh")
CALL x01234...

Network attack

"user" input ⟶ causes incorrect operation
(attacker)

"ROBERT'; DROP TABLE STUDENTS"

user input ⟶ (f) ⟶ SQL

system("cmd") system("ls")

system("echo xyz; echo abc")

Set UID

how does sudo or passwd work?

read current pw
check it
read new pw
edit /etc/passwd
(/shadow)

setuid flag

rwsx--x--x owner: root

normal
user can execute, not write

exec("/usr/bin/passwd")
→ sets user ID to owner (root)