

## Unix (and Linux) Security

- ▶ Identification and Authentication
- ▶ Access Control
- ▶ Other security related things:
  - Devices, mounting filesystems
  - Search path
  - Race conditions
- ▶ NOTE: filenames may differ between OS/distributions

1

## Users

- ▶ Principals have unique UID
  - System cares about ID, not name
  - Several users can have different names but same ID. Then they are treated as the same.
- ▶ Superuser (root) has UID = 0
  - There is only one superuser
- ▶ Stored in /etc/passwd
- ▶ Processes are subjects.

2

## UIDs for Processes

- ▶ Real user ID – The ID of the logged in principal
  - Can only be changed by root (effective user ID = 0) → this is how login works
- ▶ Effective user ID – The ID used for access control
  - Can be changed by root (effective user ID = 0) to anything
    - Used by processes with effective user ID = 0 when they temporarily access files as a less privileged user
  - Can be changed by anyone (any effective user ID) to real user ID
    - This process has to be able to get back to effective user ID = 0
- ▶ Same rules apply to group ID

3

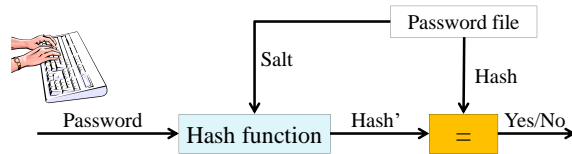
## Groups

- ▶ Can not associate multiple user IDs with one file
  - We have to put users in groups if we want several users to have access to the file
- ▶ Every user belongs to a primary group.
- ▶ Older Unix: Can only be in one group at a time
- ▶ Newer Unix and Linux: Can be in several groups at the same time
  - New files are associated with current group ID of user
  - Process group ID is the current group ID of user running the process
- ▶ Change group (newgrp)
- ▶ Primary group given in /etc/passwd
- ▶ Other groups in /etc/group
  - A group can not belong to a group

```
users:x:100:
Students:x:1000:alice,bob
```

4

## Authentication



- ▶ Salt is always used
- ▶ Hash function and salt will depend on OS
- ▶ We look at three variants

EITA25 - Computer Security

5

5

## Traditional crypt (Password Hashing)

- ▶ Design dates back to 1976
- ▶ Based on DES
- ▶ Password up to 8 characters, salt 12 bits
  - Take least significant 7 bits → 56 bit key
  - Encrypt zero string 25 times with DES
  - If bit  $i = 1$  in salt, swap bits  $i$  and  $i + 24$  in E-box output
  - Output  $12 + 64 = 76$  bits. Encode to 13 characters.
- ▶ **Problems:** Short passwords, short salts, constant cost (and fast function)

EITA25 - Computer Security

7

7

## Other Alternatives – MD5 crypt

- ▶ MD5 crypt
  - Developed for FreeBSD to avoid export restrictions and allow longer passwords (up to  $2^{64}$  bits)
  - Algorithm uses 1000 iterations → slow
  - Salt 12-48 bits
  - Output: '\$salt\$' 128 bit hash output
- ▶ **Problem:** Constant cost

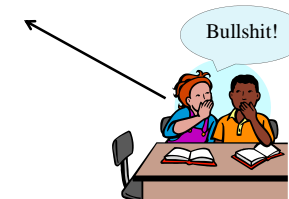
EITA25 - Computer Security

9

9

## Other Alternatives – bcrypt

- ▶ Based on block cipher blowfish
- ▶ Password up to 72 characters, 128 bit random salt
- ▶ Internal loop with variable cost
- ▶ Output '\$cost\$salt\$' + 192 bit hash output
- ▶ Default in OpenBSD
- ▶ **All problems solved**



EITA25 - Computer Security

11

11

## Current defaults on different systems

- ▶ Arch Linux     SHA-512-crypt with 5000 rounds
- ▶ Ubuntu 18.04   SHA-512-crypt with 5000 rounds
- ▶ FreeBSD 12     SHA-512-crypt with 5000 rounds
- ▶ OpenBSD 6.4    bcrypt with cost 10
- ▶ macOS 10.8+    PBKDF2-SHA512 with ~40000 rounds

EITA25 - Computer Security

12

12

## Comparison

	DES crypt	MD5 crypt	bcrypt	sha512crypt
Password length	max 8 chars	virtually any	max 72 chars	virtually any
Salt length	12 bits	12-48 bits	128 bits	typically 48 bits
Variable cost	No	No	Yes	Yes
Hashes/sec	9 800 000	120 000	170	2 500

- ▶ Hashes/sec based on 3.4 GHz processor with 4 cores, approximate values given
- ▶ The given performance for bcrypt with cost 10, and for sha512crypt with 5000 rounds

EITA25 - Computer Security

13

13

## Final words on our password discussion

- ▶ "All problems solved" is kind of bullshit
- ▶ Some devices can be really fast to a low cost
  - With enough money they are really really really fast
  - Several instances can be implemented in parallel
- ▶ Can no longer compare
  - CPU – "needed" when verifying password
  - GPU, FPGA, ASIC – used by attackers
- ▶ Make this more fair by making hashing more difficult (costly) for GPUs, FPGAs and ASICs
- ▶ **Example:** Argon2 – variable cost in both time and memory



FPGA/ASIC



GPU

EITA25 - Computer Security

14

14

## The File /etc/passwd

- ▶ Store user (principal) information

Format:

```
Username:password:UID:GID:ID string:home directory:login shell
```

- ▶ File is world readable
- ▶ Example:

```
alice:x:1004:100::/home/alice:/bin/bash
bob:x:1005:100::/home/bob:/bin/bash
```

EITA25 - Computer Security

15

15

## The File /etc/shadow

- ▶ Save passwords in a non-world readable file
  - Username
  - (hashed) password
  - Date of last change (days since Jan 1, 1970)
  - Minimum days between password changes (0 means anytime)
  - Maximum days of validity
  - Days in advance to warn user about change
  - Days account is active after password expired
  - Date of account disabling (days since Jan 1, 1970)
  - Last entry is reserved

```
alice:$6$Gar7uDv0$Ihuwd...wKGlNnWavx:17912:0:99999:7:::
bob:$6$q1/LoHbE$7Md2k...hAtXiw4hW.:17912:0:99999:7:::
```

EITA25 - Computer Security

16

16

## Access Control

- ▶ **Discretionary access control** – owner of file can change permissions
- ▶ Three categories: User (owner), Group, Other (world)
- ▶ Three access rights: Read, Write, Execute

```
alice@eita25:/data/1$ ls -l
total 8
drwxr-xr-x 1 alice Students 26 Jan 16 10:05 directory
-rw-r--r-- 1 alice Students 31 Jan 16 10:04 file1
-rw-r--r-- 1 alice Students 10 Jan 16 10:04 file2
```

### Other info from ls -l

Link counter, owner, group, size, date of last change, name

EITA25 - Computer Security

17

17

## Order of Checking

1. Owner
2. Group
3. Other

### Consequence:

if owner = r and other = rw then owner has no write permission

```
alice@eita25:/data/2$ ls -l
total 0
-r--rw-rw- 1 alice Students 0 Jan 16 10:06 file
alice@eita25:/data/2$ echo hello > file
bash: file: Permission denied
```

```
bob@eita25:/data/2$ ls -l
total 0
-r--rw-rw- 1 alice Students 0 Jan 16 10:06 file
bob@eita25:/data/2$ echo hello > file
bob@eita25:/data/2$
```

EITA25 - Computer Security

18

18

## Permissions For Directories

- ▶ Read = list the directory
- ▶ Write = Delete, rename and insert files in directory
- ▶ Execute = access directory and access files in directory

```
alice@eita25:/data/3$ ls -la
total 0
dr-xr-xr-x 1 alice Students 8 Jan 16 10:09 .
drwxr-xr-x 1 root root 36 Jan 16 10:33 ..
-rw-rw-rw- 1 alice Students 0 Jan 16 10:09 file
alice@eita25:/data/3$ rm -f file
rm: cannot remove 'file': Permission denied
```

```
alice@eita25:/data/4$ ls -la
total 0
drwxr-xr-x 1 alice Students 8 Jan 16 10:10 .
drwxr-xr-x 1 root root 36 Jan 16 10:33 ..
-rw-r--r-- 1 root root 0 Jan 16 10:10 file
alice@eita25:/data/4$ rm -f file
alice@eita25:/data/4$
```

EITA25 - Computer Security

19

19

## Change Permissions – chmod

- ▶ Used to change permissions on files
- ▶ Mnemonics can be used: **u**ser, **g**roup, **o**ther, **a**ll, **r**ead **w**rite **e**xecute.
- ▶ Examples:
  - chmod u+rw file
  - chmod u=r file
  - chmod a+rx file
  - chmod u-w,g+r,o+r file
  - chmod a-rwx,u+r file1 file2

EITA25 - Computer Security

20

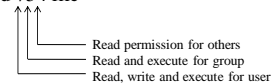
20

## Change Permissions – chmod

- ▶ Alternatively, numbers can be used.
- ▶ See each group of permissions as one number.
  - Read = 4
  - Write = 2
  - Execute = 1

Sum gives permission

- ▶ Example:
  - chmod 754 file



```
alice@eita25:/data/5$ chmod 754 file; ls -l file
-rwxr-xr-- 1 alice Students 0 Jan 16 10:12 file
```

EITA25 - Computer Security

21

21

## Setuid and Setgid (programs)

- ▶ Controlled invocation
- ▶ Effective ID of process is ID of program owner (usually root)
  - Here is the situation when RUID ≠ EUID
- ▶ Used to temporarily change access rights
- ▶ *x* is replaced by *s*

```
alice@eita25$ ls -l
total 40
-rwxr-sr-x 1 root root 16568 Jan 16 10:17 prog_setgid
-rwsr-xr-x 1 root root 16568 Jan 16 10:17 prog_setuid
alice@eita25$ ./prog_setuid &
[1] 249
alice@eita25$ ./prog_setgid &
[2] 250
alice@eita25$ ps -C prog_setgid,prog_setuid -o pid,ruser,euser,rgroup,egroup,args
PID RUSER  EUSER  RGROUP  EGROUP  COMMAND
249 alice   root    users   users   ./prog_setuid
250 alice   alice  users   root    ./prog_setgid
```

EITA25 - Computer Security

22

22

## Setuid and Setgid (Directories)

- ▶ Setuid on directory usually ignored
- ▶ Setgid on directory causes new files to get the same group as directory

```
alice@eita25:/data/7$ ls -l
total 0
drwxr-s--- 1 alice root 0 Jan 16 10:19 directory
alice@eita25:/data/7$ cd directory; touch file; ls -l
total 0
-rw-r--r-- 1 alice root 0 Jan 16 12:55 file
```

Without setgid, file would get the group which is current group ID for user (set by newgrp and defaults to primary group).

Allows users to share files more easily

EITA25 - Computer Security

23

23

## Important SUID Programs

- ▶ `/usr/bin/passwd`    change password
- ▶ `/bin/su`            change UID program

```
alice@eita25:/data$ ls -l /usr/bin/passwd /bin/su
-rwsr-xr-x 1 root root 44664 Jan 25 2018 /bin/su
-rwsr-xr-x 1 root root 59640 Jan 25 2018 /usr/bin/passwd
```

*Setuid and setgid:*  
`chmod u+s file` or `chmod 4XXX file`  
`chmod g+s file` or `chmod 2XXX file`

EITA25 - Computer Security

24

24

## The dangers of the setuid bit

- ▶ Extremely important to write correct code, since it will run as root
- ▶ Anything you code *will* be used against you
- ▶ Example of vulnerability: CVE-2018-14665 from October 2018
  - Affected most major Linux and BSD distributions
  - Xorg-server, binary `Xorg` has setuid bit set, so regular users can launch X with garbage (the output from X)
  - `Xorg -fp "root::16431:0:99999:7::" -logfile /etc/shadow` overwrites `/etc/shadow` with garbage (the output from X)
  - `Xorg -fp "root::16431:0:99999:7::" -logfile /etc/shadow`
  - The command above also adds an extra line to shadow, which sets a blank root password
  - Any user can now get root privileges simply by issuing `su`

EITA25 - Computer Security

25

25

## Sticky Bit

- ▶ Historically used to keep program code in memory when exiting program (still the case in, e.g. HP-UX)
- ▶ Now used to only let owner delete file
  - directory owner and superuser can also delete it

```
bob@eita25:/data/$ ls -la
total 0
drwxrwxr-t 1 alice Students 8 Jan 16 10:26 .
drwxr-xr-x 1 root root 36 Jan 16 10:33 ..
-rw-rw-r-- 1 alice Students 0 Jan 16 10:26 file
bob@eita25:/data/$ rm file
rm: cannot remove 'file': Operation not permitted
```

```
bob@eita25:/data/$ ls -la
total 0
drwxrwxr-x 1 alice Students 8 Jan 16 10:26 .
drwxr-xr-x 1 root root 36 Jan 16 10:33 ..
-rw-rw-r-- 1 alice Students 0 Jan 16 10:26 file
bob@eita25:/data/$ rm file
bob@eita25:/data/$
```

- ▶ Typical example: the directory `/tmp` has sticky bit set

EITA25 - Computer Security

26

26

## Default Access Rights (umask)

- ▶ Control default permissions, stored in `/etc/profile`
- ▶ Override in `~/.profile` or in prompt
- ▶ `umask` tells which permissions to **exclude** by default
- ▶ Access = full access AND NOT(umask)
  - Full access for programs and directories: `0777`
  - Full access for files: `0666`

```
alice@eita25:/data/a$ umask 0027; mkdir directory; touch file; ls -l
total 0
drwxr-x--- 1 alice users 0 Jan 16 12:59 directory
-rw-r----- 1 alice users 0 Jan 16 12:59 file
```

EITA25 - Computer Security

27

27

## Change Owner and Group (chown and chgrp)

- ▶ chown is used to change the owner of a file (or directory)
- ▶ chgrp is used to change the group of a file (or directory)
  - chown can set group also
- ▶ **Possible problem:** A user creates a suid program and owner gets changed to root
- ▶ **Common solution:**
  - Only root can change owner and setuid and setgid bits are removed when owner is changed
  - Anyone can change group to a group they are member of, but setuid and setgid bits are removed when group is changed
- ▶ Other solutions possible
  - Let only root use chown, but preserve setuid and setgid bits
  - Let any user change owner on his/her own files, but remove setuid and setgid bits

EITA25 - Computer Security

28

28

## The inode

- ▶ Stores file information
- ▶ Directory contains filename and inode number

```
alice@eita25:/data/b$ ls -l
25600573 file1 25600574 file2 25600575 file3 25600573 file4
```

Note that file1 and file4  
points to the same  
inode

- ▶ inode contains e.g.:
  - Access rights
  - Owner (UID)
  - Group (GID)
  - Time of latest access, modification and change
  - Size of file
  - Pointers to block of data

EITA25 - Computer Security

29

29

## inode Information (stat)

- ▶ Some information about an inode can be found using stat

```
alice@eita25:/data/b$ stat file1
File: file1
Size: 30          Blocks: 8          IO Block: 4096   regular file
Device: 19h/25d  Inode: 25600573    Links: 2
Access: (0644/-rw-r--r--)  Uid: ( 1004/  alice)   Gid: ( 1000/Students)
Access: 2019-01-16 10:34:10.995269889 +0000 ← Last access
Modify: 2019-01-16 10:34:49.137268710 +0000 ← Last modification of file
Change: 2019-01-16 10:35:38.482208788 +0000 ← Last modification of inode
Birth: -
Access rights given to this file
```

EITA25 - Computer Security

30

30

## Copy files

- ▶ Files can be copied in two ways
- ▶ **cp src dest**
  - Creates a new inode and new physical file owned by user running cp
- ▶ **ln target linkname**
  - Creates filename and pointer to target's inode. No new file is created.
  - When one filename is deleted the other is still there and the file is not deleted
  - rm subtracts the number of links in the inode by 1. If it becomes zero the corresponding data block is freed
- ▶ **ln -s target linkname**
  - Creates a symbolic link, not a real link
  - When opening symbolic link for reading or writing link is automatically dereferenced
  - If target is deleted, the symbolic link remains, pointing to nothing

EITA25 - Computer Security

31

31

## Race conditions

- ▶ Assume process "proc" with effective user ID = 0 writes to files in /tmp directory
  - Process creates e.g., /tmp/file and writes temporary data to this file
- ▶ What if malicious user creates /tmp/file as symbolic link to /etc/passwd?
  - The file /etc/passwd will be overwritten since "proc" has write access to this file
  - System is damaged
- ▶ **Race condition:** Who creates the file first

32

## Solutions To This Race Condition

- ▶ Create files with unpredictable filenames in /tmp
  - Still, attacker can try thousands of filenames and will succeed with probability  $> 0$
- ▶ Use O\_CREAT and O\_EXCL flag when opening file
  - Then open fails if file already exists
  - Will not follow symbolic links during creation either

Function mkstemp()  
will do this

33

## Protection of devices

- ▶ Devices are treated as files
- ▶ **Example:** If you can read/write physical memory all access control is overruled!
- ▶ **/dev/mem** is the physical memory
- ▶ **/dev/sda** is the first disk drive (in Linux)

34

## Mounting File Systems

- ▶ Different physical devices put under a single root "/"
- ▶ The mounted file system may contain unwelcome programs
  - nosuid – turn off SUID and SGID bits
  - noexec – no binaries can be executed
  - nodev – no devices can be accessed
  - ro – read-only
- ▶ UIDs and GIDs are local identifiers that need not be interpreted the same on different Unix systems
  - Need *global unique* identifiers on networks

35



## Searchpath

- ▶ When executing programs, system needs to know where to look for it → PATH tells system where to look
- ▶ PATH=: \$HOME/bin:/usr/bin:/bin:
  - Programs can be located in current directory + 3 bin directories
  - Trojan horse
- ▶ Can be a bad idea to put your current directory in the search path (especially for programs executed by root)
- ▶ Order matters, current directory last reduces risk
- ▶ Alternatively, call program by full name

## Before the labs

- ▶ Labs start next week (yay!)
- ▶ Sign up if you haven't already
  - Lots of free spots Friday afternoon :D
- ▶ There are preparatory assignments for all labs
  
- ▶ Start preparations early for lab 2
- ▶ Also, start preparations early for lab 2
- ▶ Note that there are two question hours especially for lab 2
  - Check course home page under Labs for exact hours
  
- ▶ You are not alone! You can ask me questions