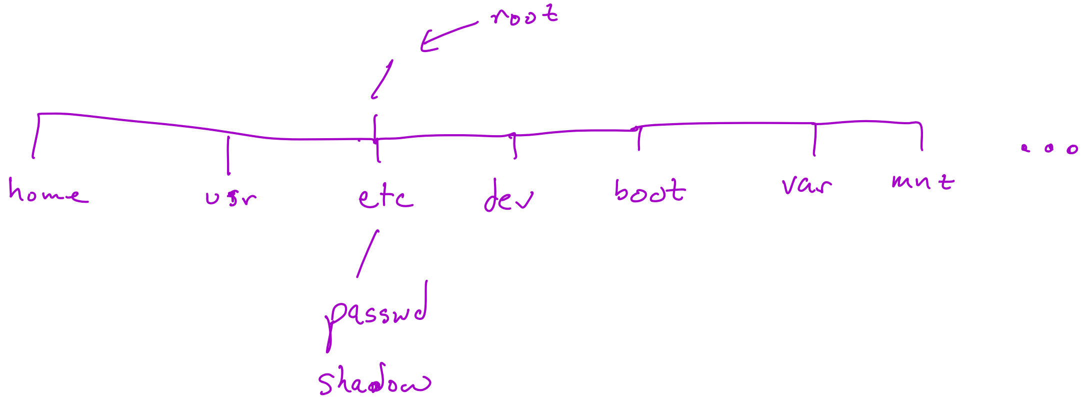
Linux Security Basics

Outline

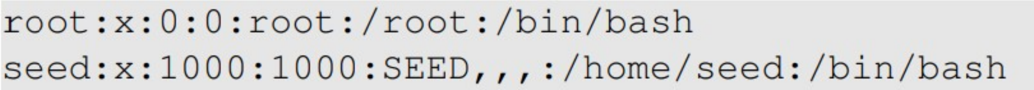
- Users and groups
- Permissions and access control
- Running commands with privilege
- Authentication



USER AND GROUP

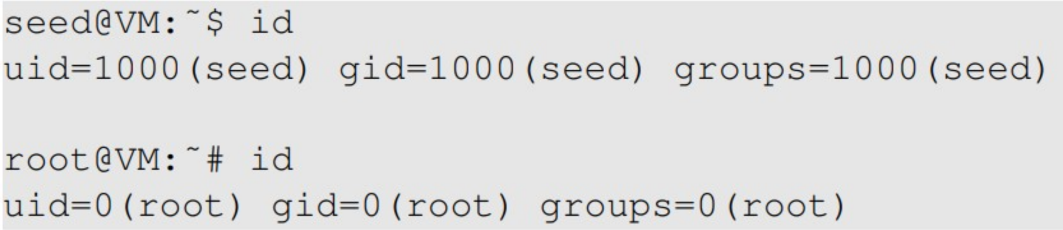
Users

- In Linux, each user is assigned a unique user ID
- User ID is stored in /etc/password



```
root:x:0:0:root:/root:/bin/bash
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
```

- Find user ID



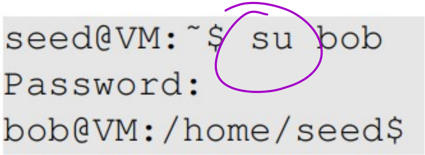
```
seed@VM:~$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed)

root@VM:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Add Users & Switch to Other Users

- Add users
  - Directly add to /etc/password
  - Use “adduser” command

- Switch to another user

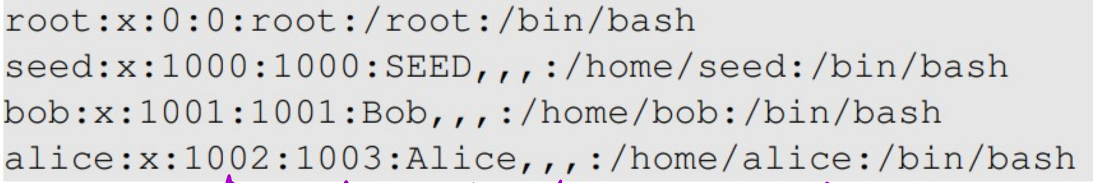


```
seed@VM:~$ su bob
Password:
bob@VM:/home/seed$
```

Group

/etc/group

- Represent a group of users
- Assigning permissions based on group
- A user can belong to multiple groups
- A user’s primary group is in /etc/password

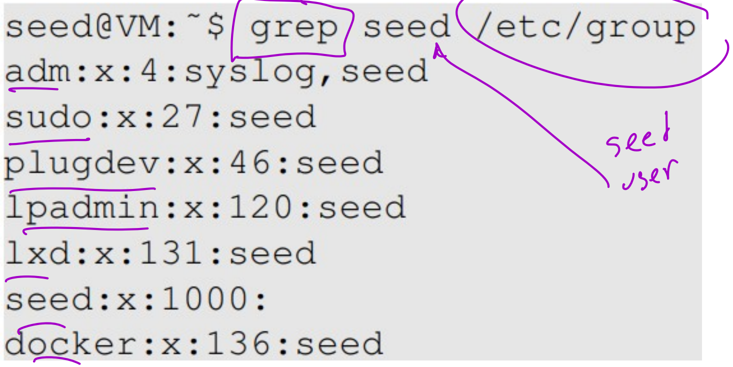


```
root:x:0:0:root:/root:/bin/bash
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
bob:x:1001:1001:Bob,,,:/home/bob:/bin/bash
alice:x:1002:1003:Alice,,,:/home/alice:/bin/bash
```

? uid groupid

comment

Which Group Does a User Belong To?

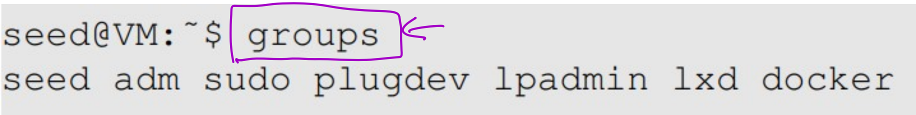


```
seed@VM:~$ grep seed /etc/group
adm:x:4:syslog,seed
sudo:x:27:seed
plugdev:x:46:seed
lpadmin:x:120:seed
lxd:x:131:seed
seed:x:1000:
docker:x:136:seed
```

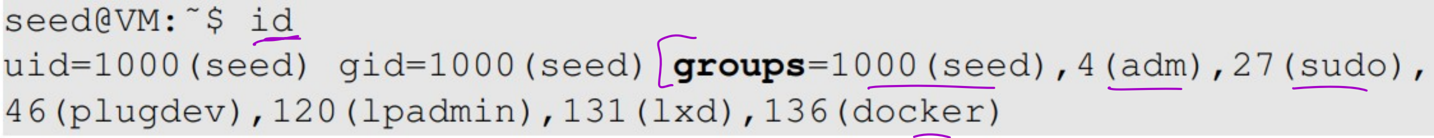
seed
user

grep
egrep

find



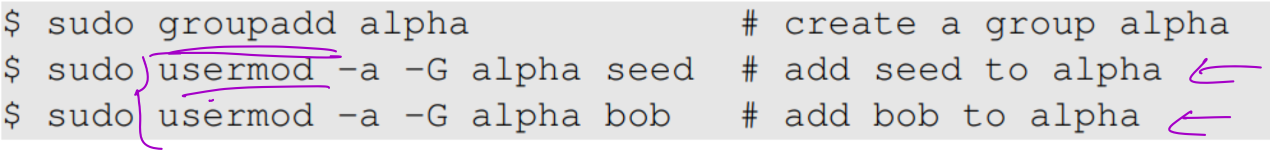
```
seed@VM:~$ groups
seed adm sudo plugdev lpadmin lxd docker
```



```
seed@VM:~$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),27(sudo),
46(plugdev),120(lpadmin),131(lxd),136(docker)
```

Group Management

How to add users



```
$ sudo groupadd alpha                # create a group alpha
$ sudo usermod -a -G alpha seed      # add seed to alpha
$ sudo usermod -a -G alpha bob       # add bob to alpha
```

adduser

groupadd

usermod

- apt update ←
- apt upgrade ←

history

Package managers

apt → dpkg

dpkg -l

apt list | grep installed | more

PERMISSIONS AND ACCESS CONTROL

Traditional Permission Model

- Types of access on files
  - **read (r)**: user can view the contents of the file
  - **write (w)**: user can change the contents of the file
  - **execute (x)**: user can execute or run the file if it is a program or script
- Types of access on directories
  - **read (r)**: user can list the contents of the directory (e.g., using ls)
  - **write (w)**: user can create files and sub-directories inside the directory
  - **execute (x)**: user can enter that directory (e.g., using cd)

$2^2 \quad 2^1 \quad 2^0$

$\boxed{4 \mid 2 \mid 1}$ =

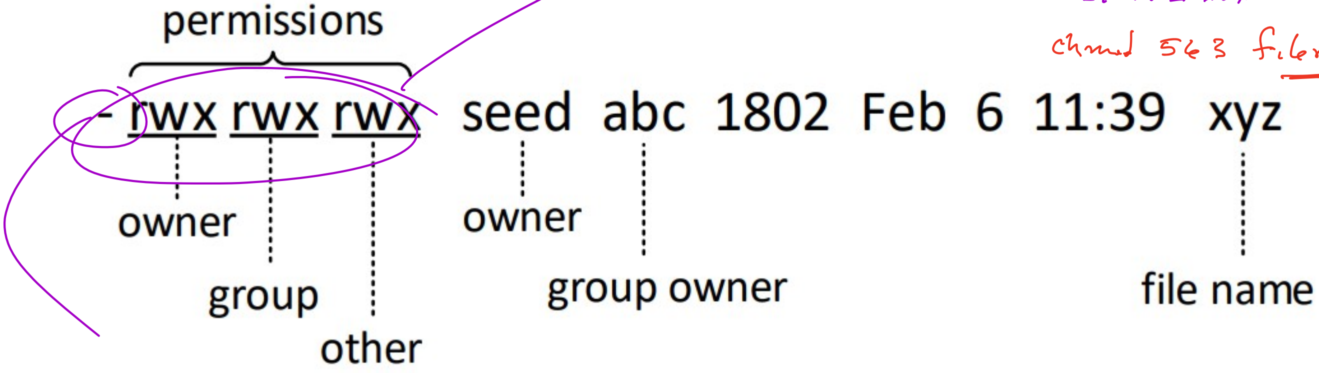
$1 \quad 0 \quad 1 \rightarrow 5 \rightarrow 1*2^2+0\times 2^1+1*2^0$

$1 \quad 1 \quad 0 \rightarrow 6$

File Permissions

ls -l

chmod



permissions

- rwx rwx rwx seed abc 1802 Feb 6 11:39 xyz

owner

group

other

owner

group owner

file name

rwxrwxrwx

octal value

5 6 3

$101\ 110\ 011_2 \rightarrow 563$

r_x rw_ _wx

chmod 563 filename

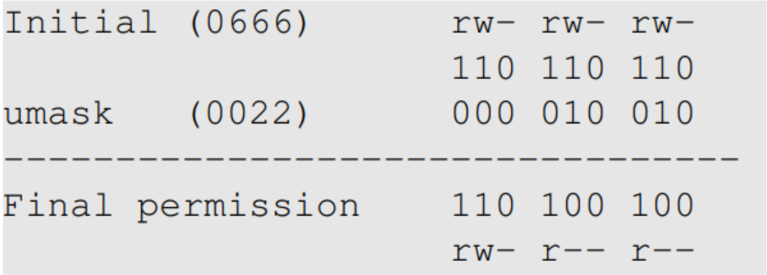
-

d

l - a link to another file/directory

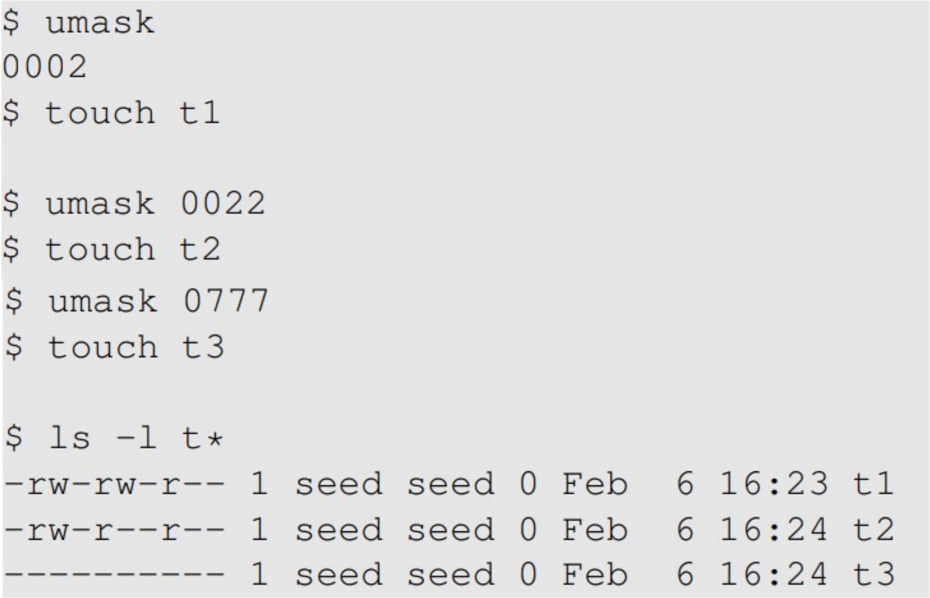
Default File Permissions

- umask value: decides the default permissions for new files
- Example



```
Initial (0666)       rw- rw- rw-
                     110 110 110
umask   (0022)       000 010 010
---------------------------------
Final permission     110 100 100
                     rw- r-- r--
```

Examples (umask)



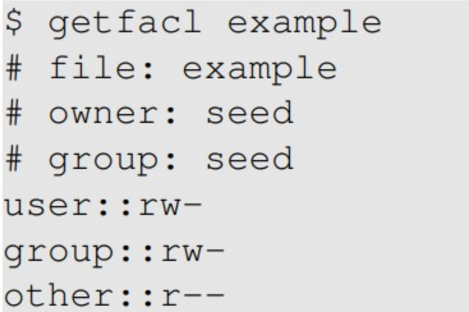
```
$ umask
0002
$ touch t1

$ umask 0022
$ touch t2
$ umask 0777
$ touch t3

$ ls -l t*
-rw-rw-r-- 1 seed seed 0 Feb  6 16:23 t1
-rw-r--r-- 1 seed seed 0 Feb  6 16:24 t2
---------- 1 seed seed 0 Feb  6 16:24 t3
```

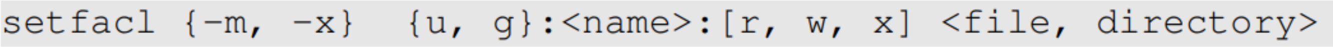
Access Control List

- Fine grained ACL
- Assign permissions to individual users/groups
- Coexist with the traditional permission model
- Example

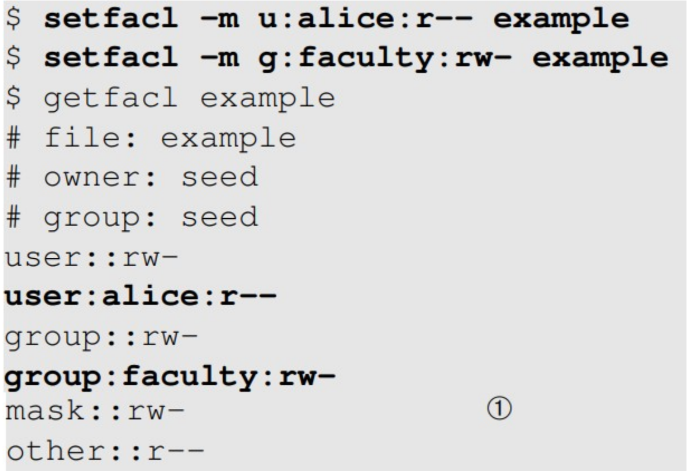


```
$ getfacl example
# file: example
# owner: seed
# group: seed
user::rw-
group::rw-
other::r--
```

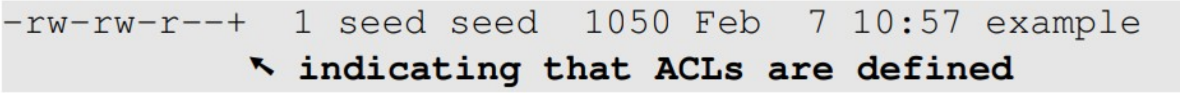
ACL Commands



```
setfacl {-m, -x}  {u, g}:<name>:[r, w, x] <file, directory>
```



```
$ setfacl -m u:alice:r-- example
$ setfacl -m g:faculty:rw- example
$ getfacl example
# file: example
# owner: seed
# group: seed
user::rw-
user:alice:r--
group::rw-
group:faculty:rw-
mask::rw-                        ①
other::r--
```



```
-rw-rw-r--+  1 seed seed  1050 Feb  7 10:57 example
           ↖ indicating that ACLs are defined
```

RUNNING COMMAND WITH PRIVILEGE

Why

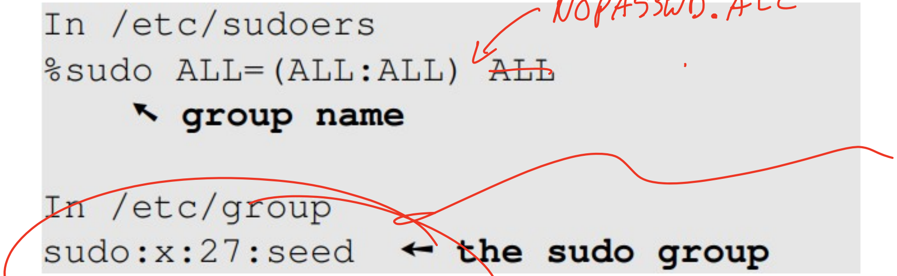
- Three command mechanisms
  - sudo
  - Set-uid programs (covered in a separate chapter)
  - POSIX capabilities

Using sudo

cd /

rm -rf *

- sudo: Super-user Do
- Run commands as a superuser
- A user must be authorized (/etc/sudoers)
- Here is how the seed user is allowed to run sudo



NOPASSWD:ALL

```
In /etc/sudoers
%sudo ALL=(ALL:ALL) ~~ALL~~
     ↖ group name

In /etc/group
sudo:x:27:seed  ← the sudo group
```

grep seed group

Getting Root Shell

```
- env
- env | grep PATH
```

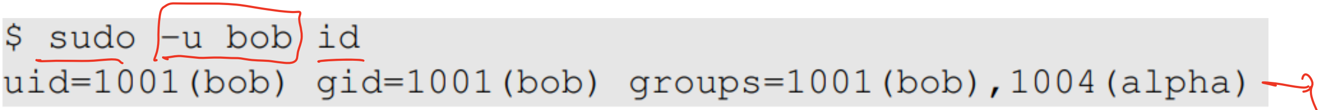
```
set
set | grep PATH
set | grep PS1
               PS2
```

PS1 sets up the primary prompt

- In Ubuntu 20.04, the root user account is locked
- Cannot log into the root account
- There are many ways to get a root shell
  - sudo –s
  - sudo bash #
  - sudo su
- It is not recommended to run commands using a root shell. Instead, use sudo to run individual commands.

Running Command Using Another User

- Run command using another user (instead of root, default)

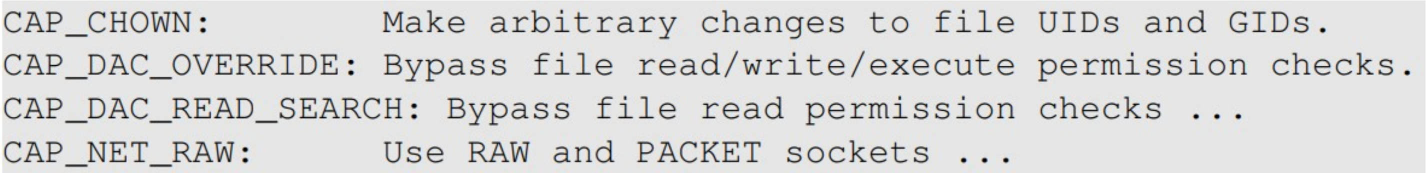


```
$ sudo -u bob id
uid=1001(bob) gid=1001(bob) groups=1001(bob),1004(alpha)
```

```
root # su bob
      $ cd
      ~
bob@vm:~ $ id
```

POSIX Capabilities

- Divide the root privilege into smaller privilege units
- Known as capabilities
- Use “man capabilities” to find all the capabilities
- Examples



```
CAP_CHOWN:          Make arbitrary changes to file UIDs and GIDs.
CAP_DAC_OVERRIDE: Bypass file read/write/execute permission checks.
CAP_DAC_READ_SEARCH: Bypass file read permission checks ...
CAP_NET_RAW:        Use RAW and PACKET sockets ...
```

ps -elf | grep bash

Setting File Capabilities (1)

- Before

Bash

```
$ cp /bin/bash ./mybash
$ ./mybash
$ cat < /etc/shadow
mybash: /etc/shadow: Permission denied     ← Failed
```

getpcaps $$

- Setting the capabilities

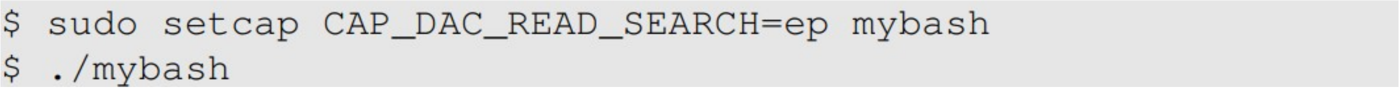
bash

```
$ sudo setcap CAP_DAC_READ_SEARCH=ep mybash
$ ./mybash
$ getpcaps $$     # List the capability of the current process
65331: = cap_dac_read_search+ep  ← The process has the capability
```

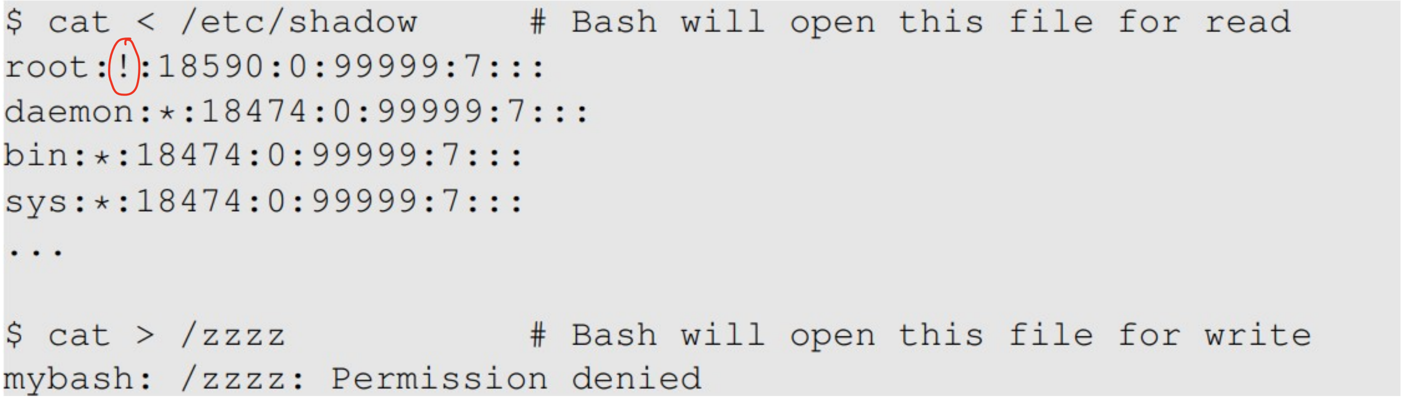
PID

Setting File Capabilities (2)

- After



```
$ sudo setcap CAP_DAC_READ_SEARCH=ep mybash
$ ./mybash
```



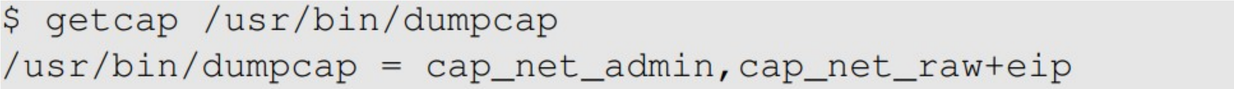
```
$ cat < /etc/shadow        # Bash will open this file for read
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
...

$ cat > /zzzz              # Bash will open this file for write
mybash: /zzzz: Permission denied
```

?

Case Study 1: Wireshark

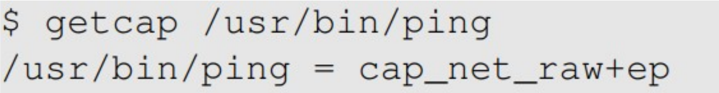
- Wireshark
  - Sniffing tool, needs privilege
  - The graphic part is not privileged
  - The sniffing part is done by dumpcap, privileged



```
$ getcap /usr/bin/dumpcap
/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip
```

Case Study 2: ping

- The ping program
  - Uses raw socket
  - Has the CAP_NET_RAW capability



```
$ getcap /usr/bin/ping
/usr/bin/ping = cap_net_raw+ep
```

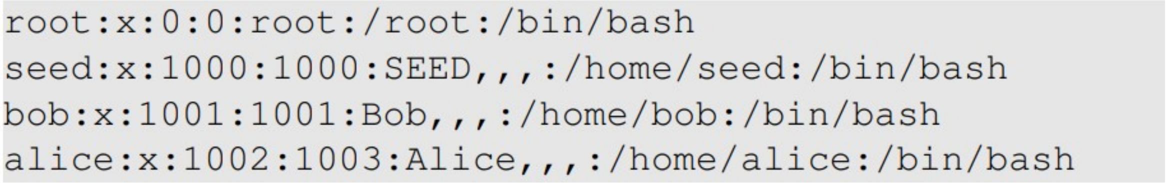
AUTHENTICATION

Authentication Methods

- A process to verify a user's identity
- Typical authentication methods
  - based on something the **user knows**: password
  - based on something the **user has**: ID card
  - based on something the **user is or does**: fingerprint
- Multi-factor authentication

The Password File

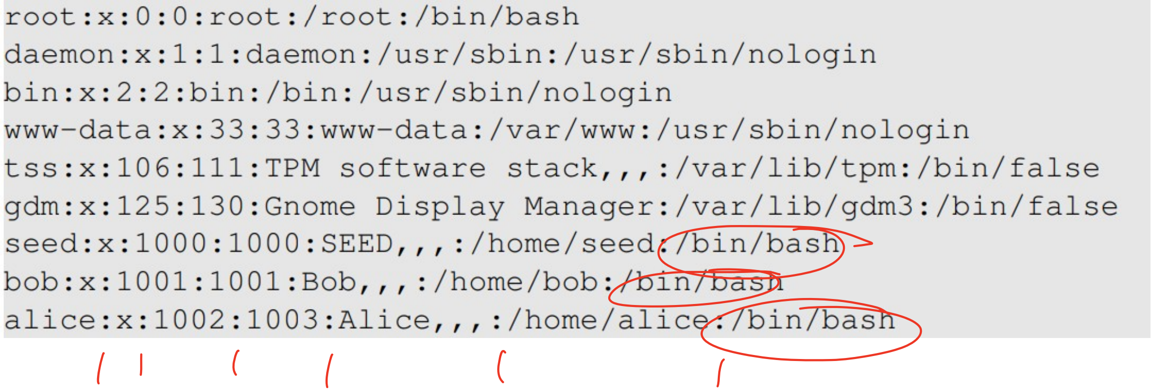
- Each entry contains a user account information
- Password is not stored here (used to be)



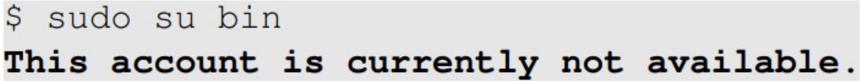
```
root:x:0:0:root:/root:/bin/bash
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
bob:x:1001:1001:Bob,,,:/home/bob:/bin/bash
alice:x:1002:1003:Alice,,,:/home/alice:/bin/bash
```

First Command After Login

- The last field of each entry



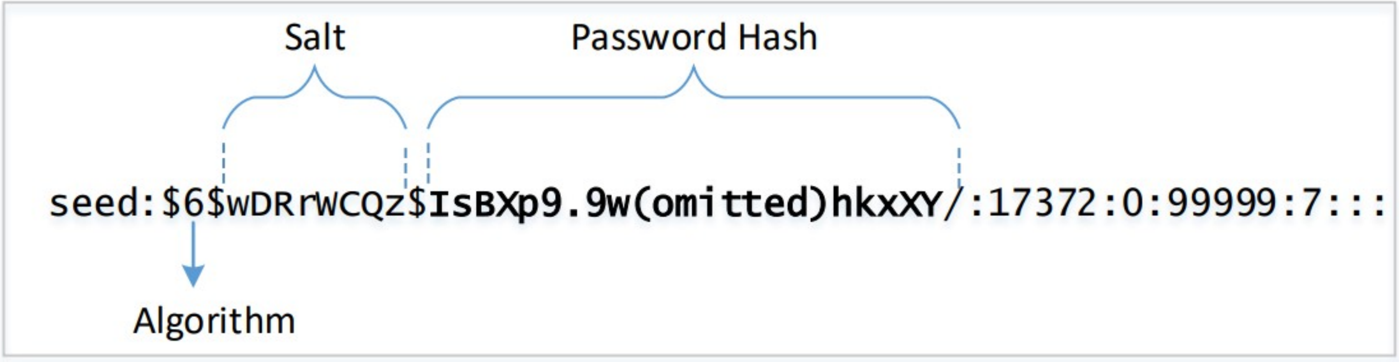
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
bob:x:1001:1001:Bob,,,:/home/bob:/bin/bash
alice:x:1002:1003:Alice,,,:/home/alice:/bin/bash
```



```
$ sudo su bin
This account is currently not available.
```

The Shadow File

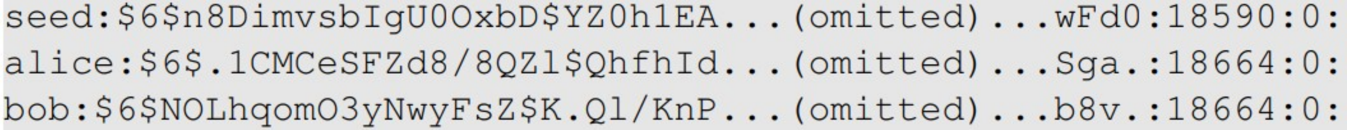
- Store password, why not use /etc/password anymore?
- Structure for each entry



```
seed:$6$wDRrWCQz$IsBXp9.9w(omitted)hkxXY/:17372:0:99999:7:::
```

The Purpose of Salt

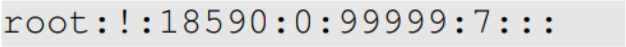
- Defeat brute-force attacks
  - dictionary attack, rainbow table attack
- These 3 accounts have the same password



```
seed:$6$n8DimvsbIgU0OxbD$YZ0h1EA...(omitted)...wFd0:18590:0:
alice:$6$.1CMCeSFZd8/8QZl$QhfhId...(omitted)...Sga.:18664:0:
bob:$6$NOLhqomO3yNwyFsZ$K.Ql/KnP...(omitted)...b8v.:18664:0:
```

Locking Account

- Putting an invalid value in the password field
- The root account is locked



```
root:!:18590:0:99999:7:::
```