CS 366
Intro to Cybersecurity

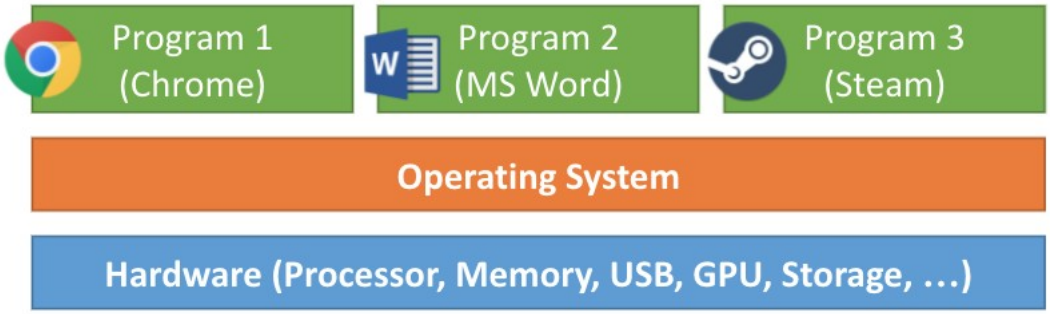
Dr. Stella Sun
EECS
University of Tennessee
Fall 2022

Today’s Class

- Operating system security
- Refresher
  - brief overview of an operating system
  - filesystem permissions (multiple users)
  - User ID (UID) & Effective User ID (EUID)
  - Set-UID programs

Operating System

- Layer of software to provide access to hardware
- Provides
  - Abstraction of complex hardware
  - Protected access to shared resources
  - Communication
  - Security



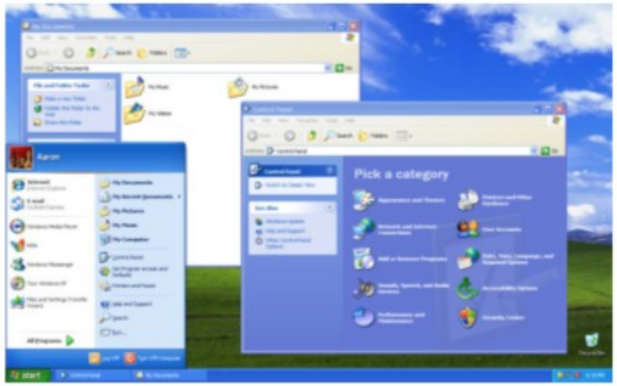
Operating System (Microsoft Examples)

Windows 3.1 (1992)

- A program is crashing → The entire OS may crash
- Many parts of the memory are shared between programs

Windows 95 (1995) – 98, ME

- Each program has their own address space
- A program is crashing → The entire OS may crash

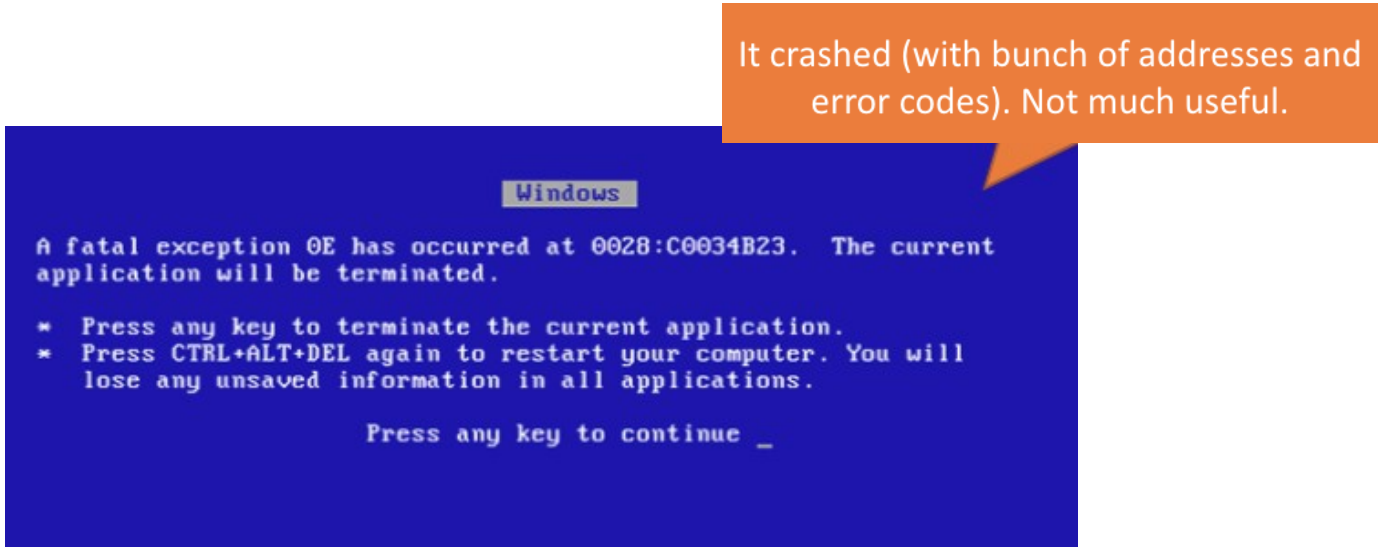


Windows XP (2001) (NT), 7, 8, 10

- Each program has their own address space
- A program can't crash the entire OS

Blue Screen of Death (BSOD)

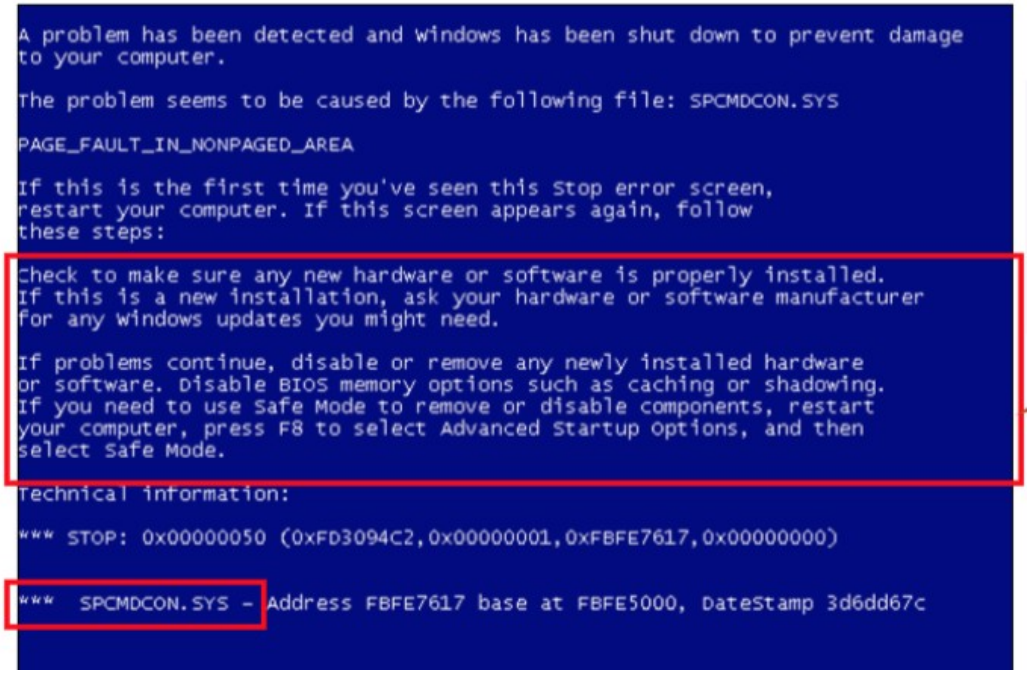
- Windows 3.1 didn’t really have one – just black screen
- Windows 9x BSOD



Blue Screen of Death (BSOD)

- Windows XP

  - You rarely see those since XP unless device drivers are unstable



It crashed (with bunch of addresses and error codes) and this is some information (while that is largely unhelpful)


Gepäckrückgabe
Baggage claim
Ein STOP: 0x00000050 (0xf8458800,0x00000000,0x00000000,0x00000002)
PAGE_FAULT_IN_NONPAGED_AREA
CPUID:AuthenticAMD 5.1.4 irql:1f SYSVER 0xf0000565
Dll Base DateStmp - Name
80010000 337546bf - ntoskrnl.exe
80001000 3344d3a53 - atapi.sys
80024000 336016a2 - Disk.sys
8003c000 3326d637 - Ntfs.sys
f4e6c000 00000000 - Null.SYS
f4e72000 00000000 - Beep.SYS
f4e7c000 31ec6c94 - kbdclass.sys
f4838000 344effa4 - cirrus69.sys
f4840000 3224804d0 - Mpfs.SYS
f4000000 336157ac - win32k.sys
ff2b1000 335bd30e - Fastfat.SYS
ff365000 3372990ef - tcpip.sys
ff3fc000 31ec6e15 - elnk3.sys
f4b80000 33248371 - netbios.sys
f4f40000 00000000 - Parallel.SYS
f4a60000 332480ab - Serial.SYS
ff205000 3360f103 - srv.sys
Dll Base DateStmp - Name
80010000 33247f88 - hal.dll
80007000 33248043 - SCSIPORT.SYS
80240000 336015af - CLASS2.SYS
f4b10000 00000000 - Floppy.SYS
f4c61000 31e4b68b - KSecDD.SYS
f4b40000 335bc82a - i8042prt.sys
f4b50000 33248011 - VIDEOPRT.SYS
f4b80000 332480d4 - Msfs.SYS
ff024000 3326da41 - NDIS.SYS
f4970000 33655903 - cirrus.dll
ff1d4000 31ec6e6c - TDI.SYS
ff340000 3362a53a - netbt.sys
f4930000 33443add - afd.sys
ff410000 00000000 - Parport.SYS
f4a62000 00000000 - ParUdm.SYS
ff23f000 3337f77c - rdr.sys
ff1f4000 332483b5 - nup.sys
Address dword dump Build [1381] - Name
ff29fe14 8013fcca 8013fcca 00000000 e0450100 00000000 000f0001 - ntoskrnl.exe
ff29fe28 8010ab58 8010ab58 ff29fe44 ff29fe58 a0050301 8010a9dd - ntoskrnl.exe
ff29fe34 a0050301 a0050301 8010a9dd ff29fe58 ff647a10 ff647a08 - win32k.sys
ff29fe38 8010a9dd 8010a9dd ff29fe58 ff647a10 ff647a08 00000000 - ntoskrnl.exe
ff29fe58 a00503cb a00503cb 00000000 00000023 00000023 00000000 - win32k.sys
ff29fe60 a00503cb a00503cb ff29fec4 00000000 a0025d6b 00000000 - win32k.sys
ff29fe6c a0025d6b a0025d6b 00000000 00010246 ff29ff04 0029fc74 - win32k.sys
ff29fe70 a00503cb a00503cb ff29fe90 00000024 ff29fe90 00000000 - win32k.sys
ff29fe8d a01f9680 a01f9680 a0123550 ffffffff ff29fee4 a0050819c - win32k.sys
ff29fe8c a0123550 a0123550 ffffffff ff29fee4 a0050819c 0000004c - win32k.sys
ff29fec8 a0050819c a0050819c 0000004c 0029fc74 00011f04 ff29fe8c - win32k.sys
ff29fedc a0102815 a0102815 00000246 ff29ff04 a00503de 0000004c - win32k.sys
ff29fee8 a00503de a00503de 0000004c 00000000 0029fc74 8013cc94 - win32k.sys
ff29fef8 8013cc94 8013cc94 0000004c 00000000 00000023 00000000 - ntoskrnl.exe
Starten Sie erneut, und verwenden Sie Wiederherstellungsoptionen in der Systemsteuerung, oder setzen Sie die Startoption /CRASHDEBUG. Wenn diese Meldung nochmal erscheint, wenden Sie sich an Ihren Systemadministrator oder Techniker.

What Is An Operating System

- Software that provides a more convenient/featureful machine interface

  - Resource sharing, protection, isolation: CPU, memory

  - Clean/easy abstractions: file reads/writes vs. accessing disk sectors

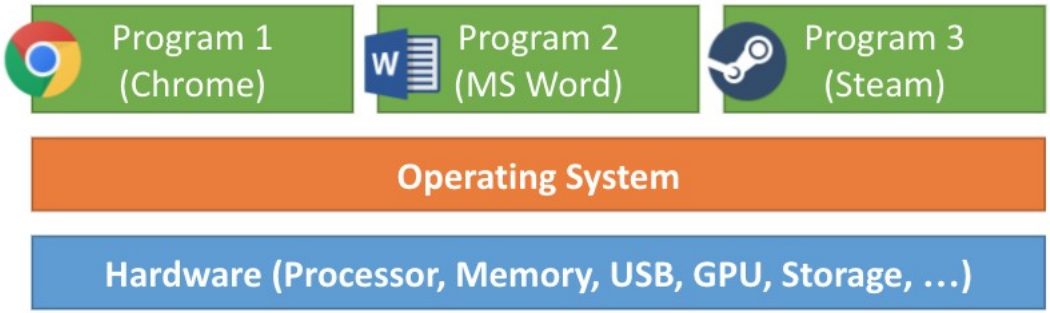
  - Provide common services: storage, authorization, networking, ...

The Security Job of An Operating System

- Controls access to resources
- Ensures integrity of resources
- Security of the OS underpins the security of all software running on the OS

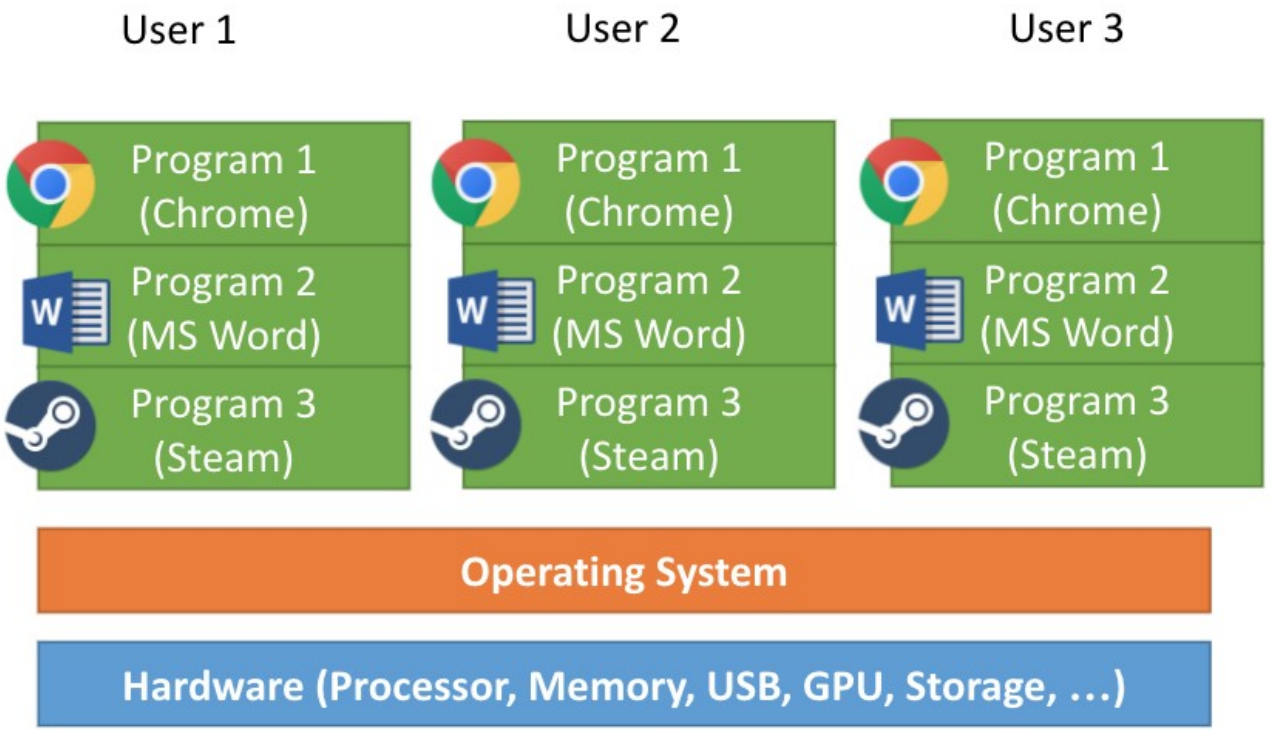
Resources the OS Manages

- Hardware
  - memory, CPU, storage, network access
- Allows three general behaviors
  - Read
  - Write
  - Execute



Protected Storage

Multiple Users in An OS



File Permissions

- Files in modern OSes have a set of permission bits
- Control a user's ability to read, write and execute a file

- we call each of these behaviors a capability

- How you resolve what a user has permission to do depends on the OS

- user-based access control: capabilities are bound to a particular user

- role-based access control: capabilities are bound to a collection of users that have particular needs based on their expected task

Unix File Permissions

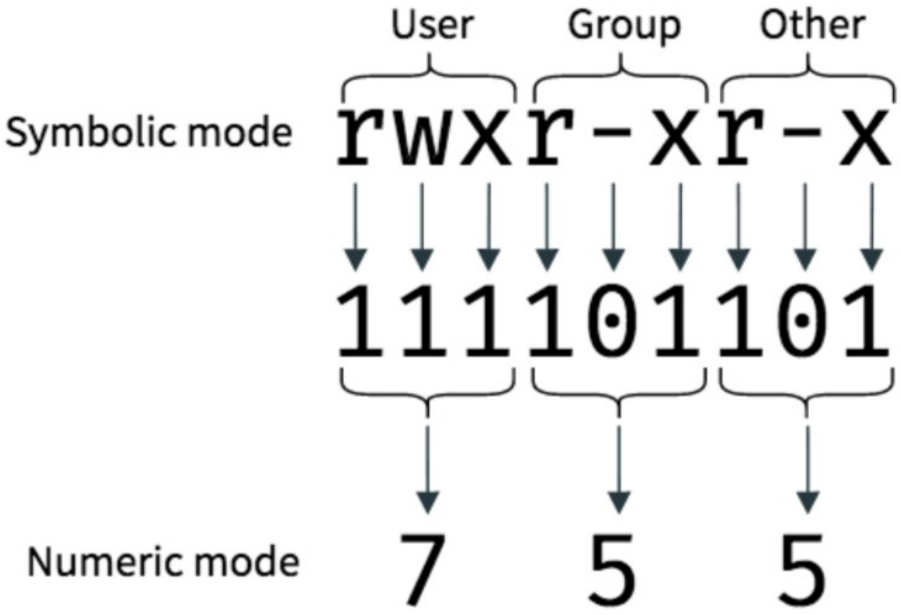
- Can be viewed with “ls -l”
- Can be changed with “chmod”
- Permission mask represented by an integer
- Can be set for three different subjects (ugo model)

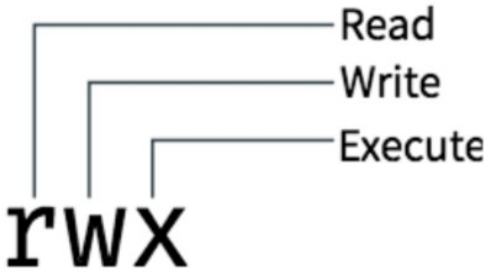
- User: file user owner
- Group: file group owner
- Other: everyone else

- Your permission level is determined by checking in that order

- First one that matches your user is the one applied

Permission Bits – A Picture





Permission Bits Example: Directories

- **r** – ability to read the contents of the directory (e.g., ls)
- **w** – ability to write into the directory (e.g., rename/create/delete files and directories within the directory)
- **x** – ability to enter that directory (e.g., cd)

Permissions Example

- foo.bar: owned by bob:gradstudents
  - with permission 754
- hello.world: owned by alice:gradstudents
  - with permission 705

- Users:
  - alice:gradstudents
  - bob:gradstudents
  - charlie:gradstudents
  - dave:undergradstudents

UID

User ID (UID)

- A number assigned by Unix to each user on the system

  - used to identify the user to the system and determine which system resources the user can access

- Special user: *root (superuser)*

  - The root user on Unix has access to everything

  - Windows has similar concepts with *Admin* and *System* accounts

  - UID = 0

Where Are UIDs Stored

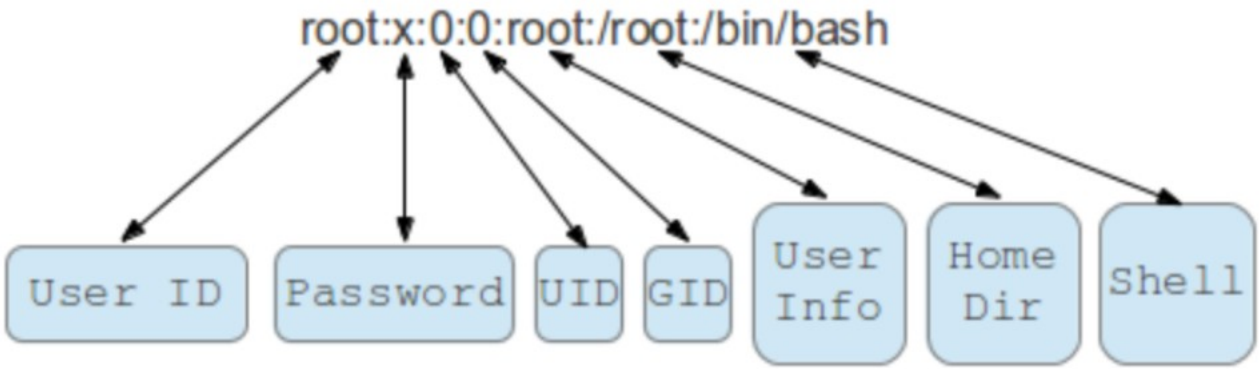
- Stored in /etc/passwd

  - Traditionally Unix keeps passwords (hashes) in this world-readable file

  - Nowadays passwords are stored in /etc/shadow readable only by root

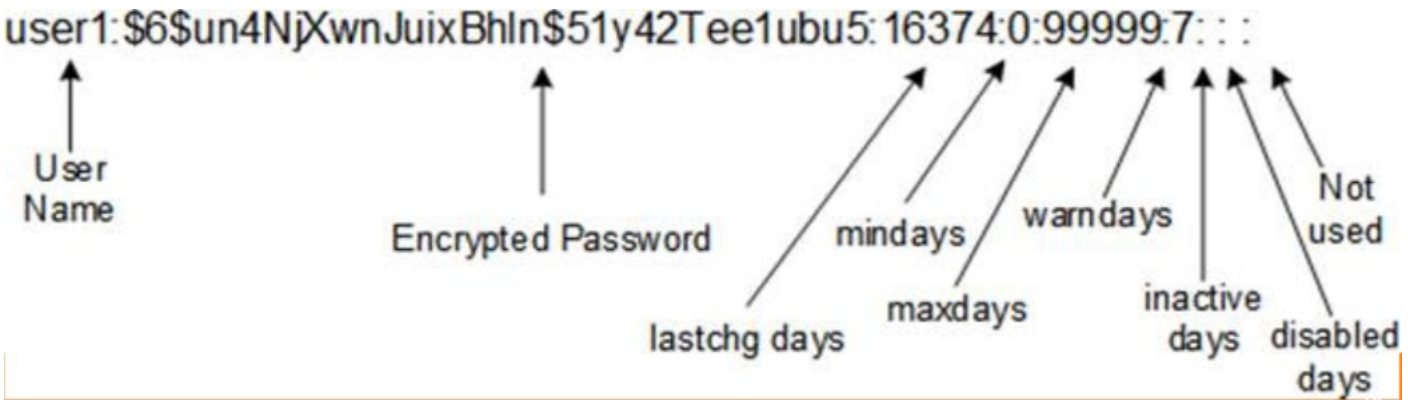
/etc/passwd

```
bob@bobs-computer:~$ cat /etc/passwd
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
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```



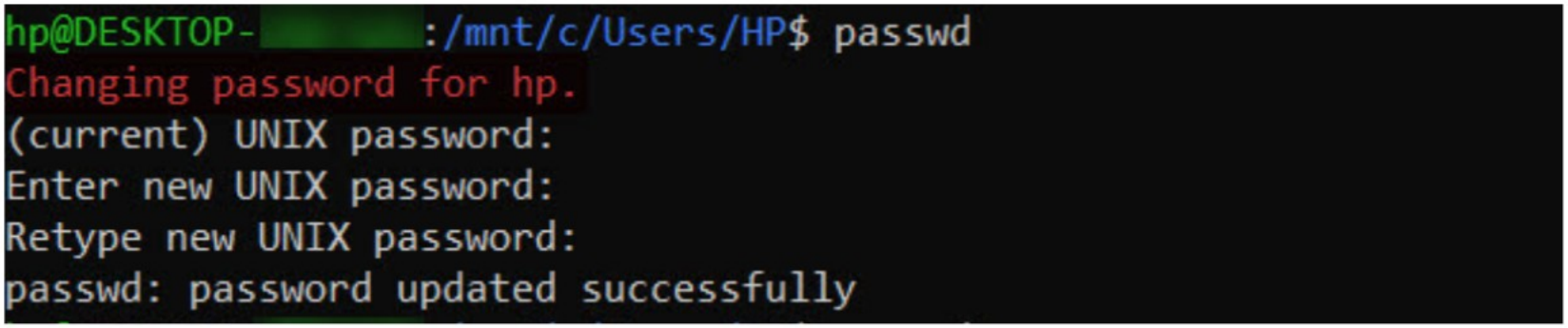
/etc/shadow

```
vcva2013:~ # more /etc/shadow
bin:*:15887:0:60:7:::
daemon:*:15887:0:60:7:::
haldaemon:*:15887:0:60:7:::
ldap:*:15887:0:60:7:::
mail:*:15385::60::::
man:*:15887:0:60:7:::
messagebus:*:15887:0:60:7:::
nobody:*:15385::60::::
ntp:*:15887:0:60:7:::
polkituser:*:15887:0:60:7:::
postfix:*:15887:0:60:7:::
root:$6$KvZ/hiWS$PhOtIUN85TXBgQxc01B3JfrRtPhXxL8n/IwW0fe5XrHETb0Cz9l4bzGlk8gGw9
CXjjbq.PPcCPTFxAJ9M3tgo.:15895:0:1095:7:::
```



Set-UID

Need for Privileged Programs



Need for Privileged Programs

- Password dilemma

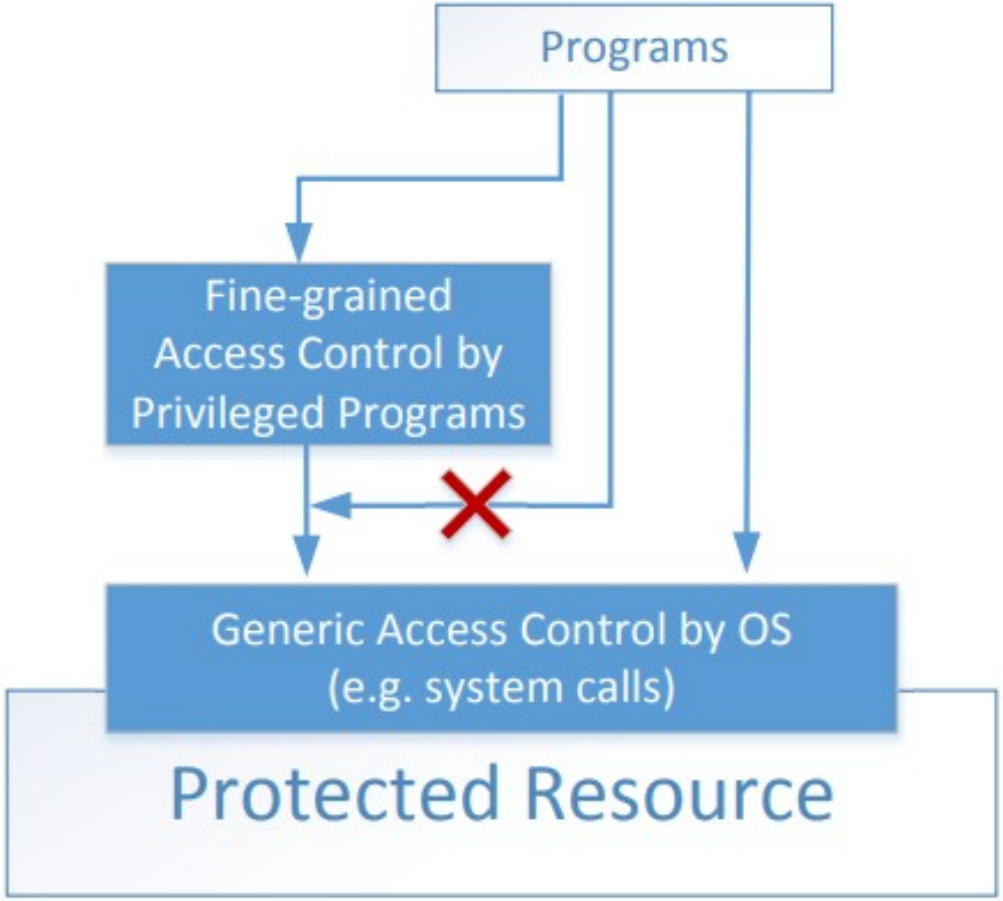
  - Permissions of /etc/shadow file

```
-rw-r----- 1 root shadow 1443 May 23 12:33 /etc/shadow
```

Only writable to the owner

  - How would normal users change their password?

Two-Tier Approach



Two-Tier Approach

- Current access control on file level: all or nothing
- Implementing fine-grained access control makes OS over complicated
- OS relies on extensions to enforce fine-grained access control
- Privileged programs are such extensions

Types of Privileged Programs

- Daemons

  - computer programs that run in the background

  - need to run as root or other privileged users

- Set-UID programs

  - widely used in Unix systems

  - programs marked with a special bit

Set-UID Concept

- Allow user to run a program with program owner's privilege
- Allow user to run programs with temporary elevated privileges
- Example: the *passwd* program

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 41284 Sep 12  2012 /usr/bin/passwd
```

Set-UID Concept

- Every process has two User IDs
- Real UID (RUID): identifies real owner of process
- Effective UID (EUID): identifies privilege of a process

  - access control is based on EUID

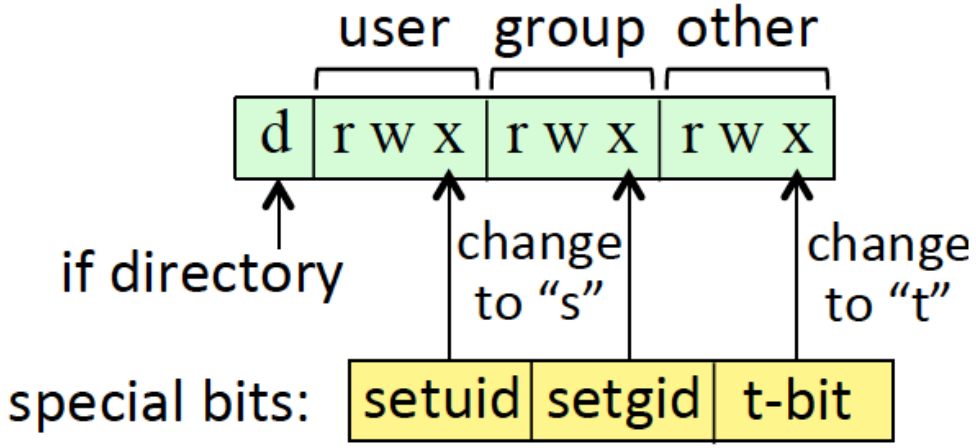
Set-UID Concept

- When a normal program is executed, RUID = EUID, they are both equal to the ID of the user who runs the program
- When a Set-UID program is executed, RUID != EUID. RUID is still equal to the user's ID, but EUID is equal to the program **owner**'s ID

  - If the program is owned by root, it runs with the root privilege

How Set-UID Works

- A Set-UID program is just like any other program, except that it has a special marking, which a single bit called Set-UID bit



How Set-UID Works

- A Set-UID program is just like any other program, except that it has a special marking, which a single bit called Set-UID bit

Binary (12 bits)	Octal	Symbolic	Meaning
000 100 000 000	0400	-r-- --- ---	user (owner) has R
000 010 000 000	0200	--w- --- ---	user (owner) has W
000 001 000 000	0100	---x --- ---	user (owner) has X
000 000 110 000	0060	--- rw- ---	group has R, W
000 000 101 000	0050	--- r-x ---	group has R, X
000 000 011 000	0030	d --- -wx ---	group has W, X; file is a directory file
000 000 000 111	0007	--- --- rwx	other has R, W, X
000 110 100 100	0644	-rw- r-- r--	user has R, W; group and other have R

How Set-UID Works

- A Set-UID program is just like any other program, except that it has a special marking, which a single bit called Set-UID bit

```
$ cp /bin/id ./myid
$ sudo chown root myid
$ ./myid
uid=1000(seed) gid=1000(seed) groups=1000(seed), ...
```

```
$ sudo chmod 4755 myid
$ ./myid
uid=1000(seed) gid=1000(seed) euid=0(root) ...
```

Turn A Program into Set-UID

- Change the owner of a file to root :

```
seed@VM:~$ cp /bin/cat ./mycat
seed@VM:~$ sudo chown root mycat
seed@VM:~$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Nov  1 13:09 mycat
seed@VM:~$
```

- Before Enabling Set-UID bit:

```
seed@VM:~$ mycat /etc/shadow
mycat: /etc/shadow: Permission denied
seed@VM:~$
```

- After Enabling the Set-UID bit :

```
seed@VM:~$ sudo chmod 4755 mycat
seed@VM:~$ mycat /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8cLWbQI1cFjn(
h/pDyc5U1BWOzkWh7T9ZGu.:15933:0:99999:7:::
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
```

Example of Set-UID

```
$ cp /bin/cat ./mycat
$ sudo chown root mycat
$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Feb 22 10:04 mycat
$ ./mycat /etc/shadow
./mycat: /etc/shadow: Permission denied
```

- Not a privileged program

```
$ sudo chmod 4755 mycat
$ ./mycat /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8c...
daemon:*:15749:0:99999:7:::
...
```

- Become a privileged program

```
$ sudo chown seed mycat
$ chmod 4755 mycat
$ ./mycat /etc/shadow
./mycat: /etc/shadow: Permission denied
```

- It is still a privileged program, but not the root privilege

How Secure Is Set-UID?

- Allows normal users to escalate privilege

  - This is different than directly giving the privilege (sudo)

  - Behaviors restricted by the Set-UID program

- Unsafe to turn all programs into Set-UID

  - Example: /bin/sh

  - Exmaple: vi (e.g., :!ls, :!/bin/sh)