Secure System Design: Threats and Countermeasures

CS-392

Spring 2021

Secure System Design: Threats and Countermeasure

- Instructor:
  - Samrat Mondal
  - samrat@iitp.ac.in
- TA :
  - Sanghamitra (1821cs14@iitp.ac.in)
  - Shashank (shashank_1921cs04@iitp.ac.in)
  - Akanksha (akanksha_2021cs39@iitp.ac.in)
- Course Materials:
  - Will be available in https://www.iitp.ac.in/~samrat/CS392_SSD/

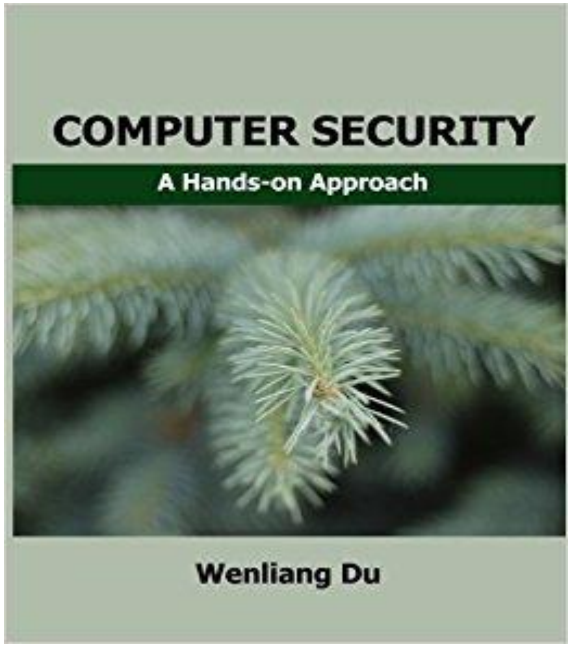
Class Timings:
Tuesday: 9 am to 9.55 am
Thursday: 9 am to 9.55 am
Friday: 9 am to 9.55 am

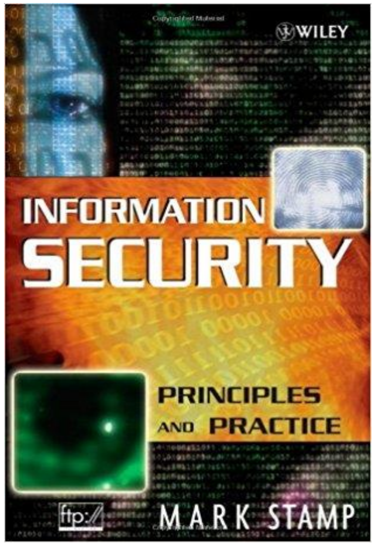
Tentative Plans

- Pre-midsem
- Overview of Unix Security basics
- buffer overflow
- format string,
- race condition;
- Shell functions, Shellshock vulnerability, Shellshock attack on Set-UID program,
- Return to libc attack;
- Dirty Cow Attack;

- Post-midsem
- Password file compromise Attack, Countermeasures;
- Code Analysis using Software Reverse Engineering;
- Interaction with the database in Web Application, SQL-Injection Attack, Countermeasures;
- ClickJacking attack;
- Cross-Site Requests and Its Problems, Cross-Site Request Forgery Attack, Scripting Attack;
- Access Control in Android Smartphone, Attack on Android Smart phone;

Books


COMPUTER SECURITY
A Hands-on Approach
Wenliang Du


WILEY
INFORMATION
SECURITY
PRINCIPLES
AND PRACTICE
ftp://
MARK STAMP

Evaluation Policy

- Assignments, Quizzes, Polls : 50%
- MidTerm Test/Viva: 25%
- EndTerm Test/Viva: 25%

Students who will be caught cheating, their assignment weightage will be reduced proportionally

Objectives of this Course

- To get familiar with the important security concerns that a software developer or manager or a stakeholder must be aware of
- To understand the various classical flaws in systems that can lead to security problems.
- Also, some possible countermeasures will be covered

Set up guidelines

- For programming assignments and practice, you can use virtual box and install 32 bit Pre-built ubuntu image from the following link
  - https://drive.google.com/file/d/12l8OO3PXHjUsf9vfjkAf7-I6bsixvMUa/view

- Check the following manual for running SEED VM on Virtual Box
  - https://seedsecuritylabs.org/Labs_16.04/Documents/SEEDVM_VirtualBoxManual.pdf

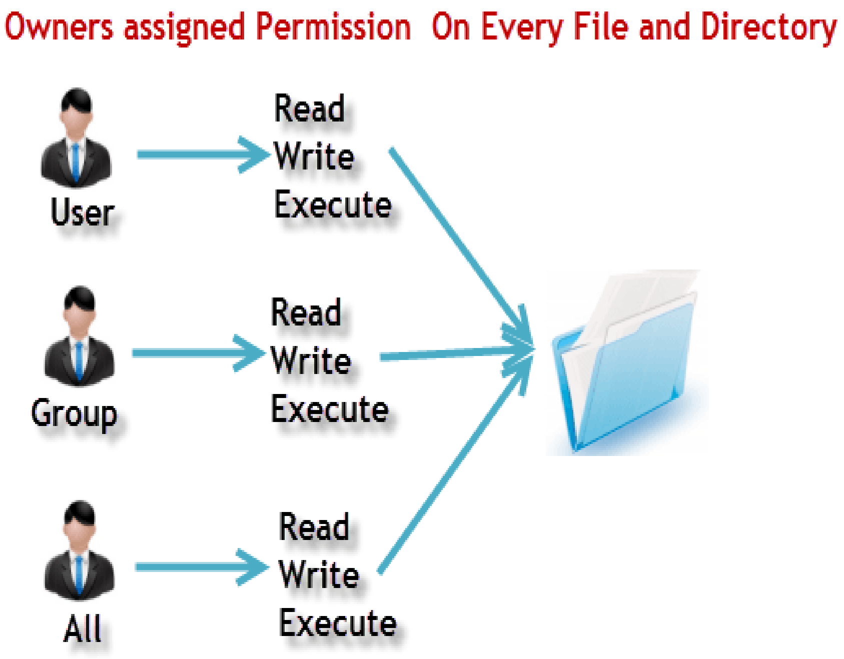
Let’s Begin

Ownership of Linux Files

- Every file and directory on Unix/Linux system is assigned 3 types of owner
  - User: A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.
  - Group: All users belonging to a group will have the same access permissions to the file.
  - Other: Any other user who has access to a file.

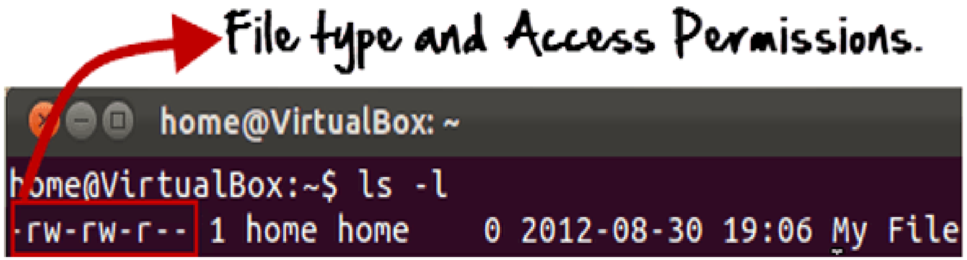
Permissions

- Every file and directory in UNIX/Linux system has following 3 permissions defined for all the 3 owners.
  - Read
  - Write
  - Execute



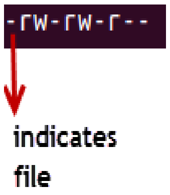
ls command to check permission

```
ls - l
```

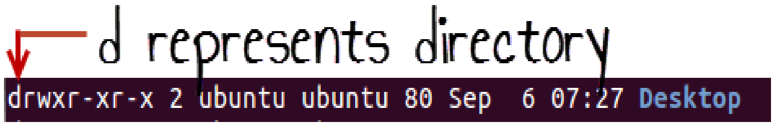


File type and Access Permissions.

```
home@VirtualBox: ~
home@VirtualBox:~$ ls -l
-rw-rw-r-- 1 home home     0 2012-08-30 19:06 My File
```



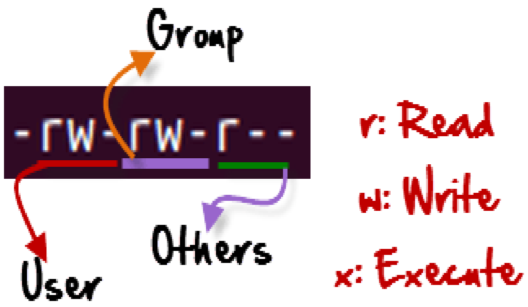
```
-rw-rw-r--
```



d represents directory

```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep  6 07:27 Desktop
```

r = read permission
w = write permission
x = execute permissior
- = no permission

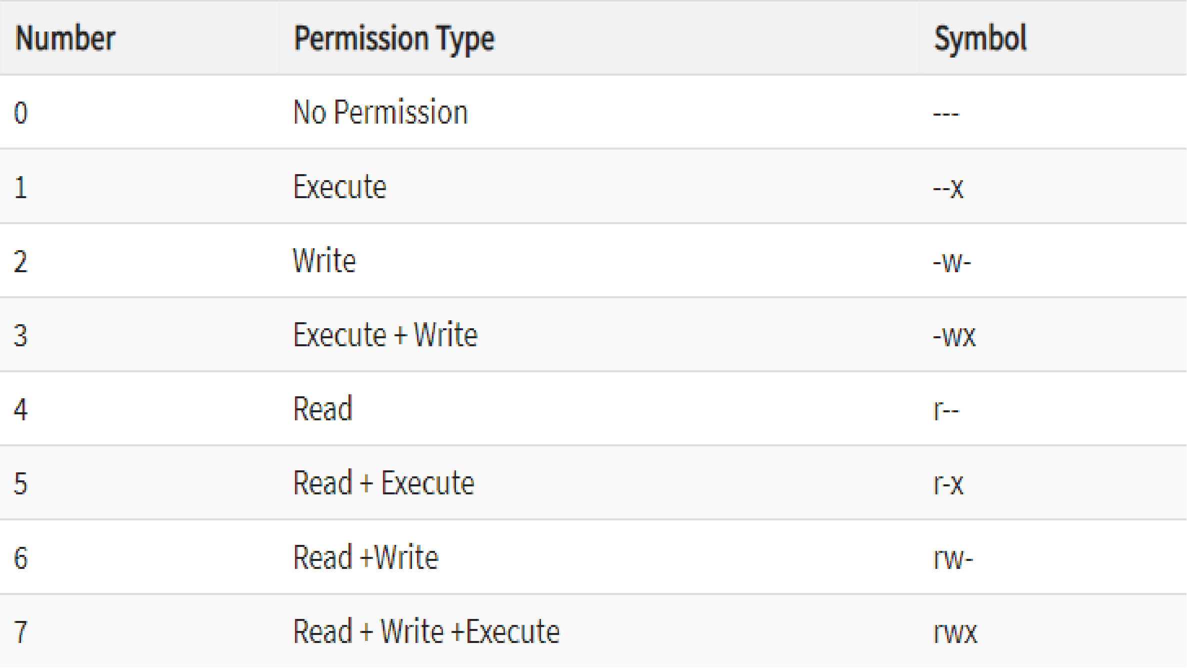


chmod command

- The '**chmod'** command stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world.
- Syntax: *chmod permission filename*
- Two ways-
  - Absolute mode
  - Symbolic mode

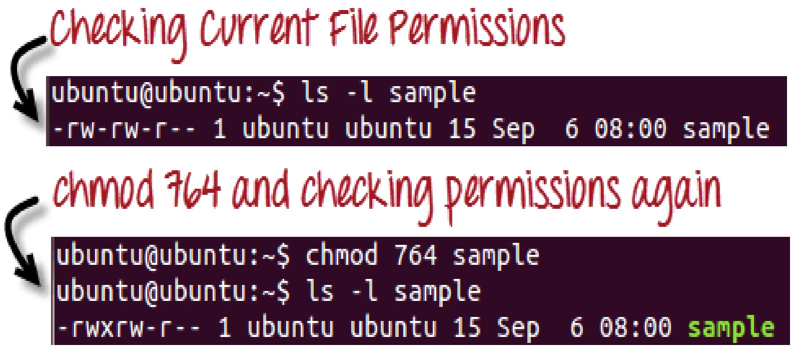
Absolute Mode

- In this mode, file **permissions are not represented as characters but a three-digit octal number**.



Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

chmod in Absolute mode

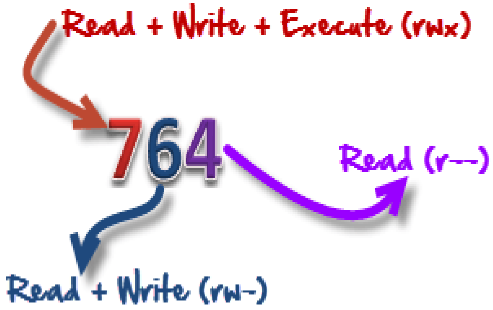


Checking Current File Permissions

```
ubuntu@ubuntu:~$ ls -l sample
-rw-rw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

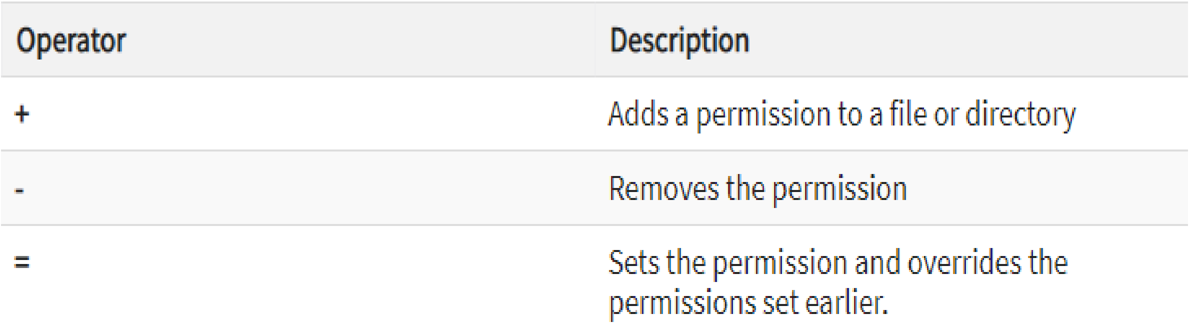
chmod 764 and checking permissions again

```
ubuntu@ubuntu:~$ chmod 764 sample
ubuntu@ubuntu:~$ ls -l sample
-rwxrw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

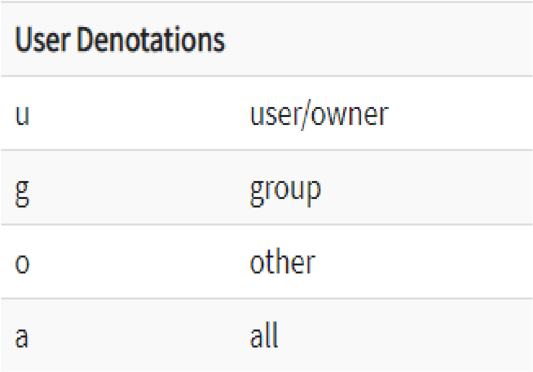


Symbolic mode

- Useful to modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

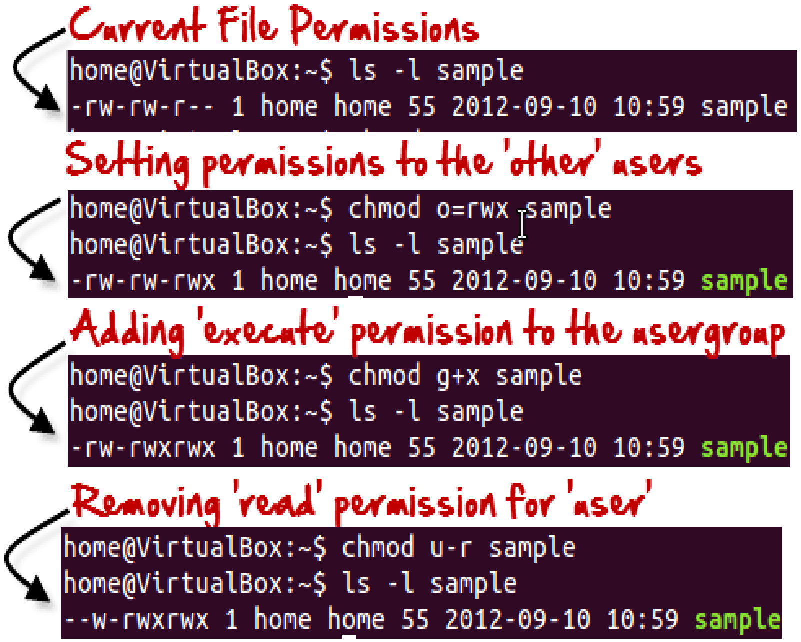


Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.



User Denotations	
u	user/owner
g	group
o	other
a	all

chmod in symbolic mode



Current File Permissions

```
home@VirtualBox:~$ ls -l sample
-rw-rw-r-- 1 home home 55 2012-09-10 10:59 sample
```

Setting permissions to the 'other' users

```
home@VirtualBox:~$ chmod o=rwx sample
home@VirtualBox:~$ ls -l sample
-rw-rw-rwx 1 home home 55 2012-09-10 10:59 sample
```

Adding 'execute' permission to the usergroup

```
home@VirtualBox:~$ chmod g+x sample
home@VirtualBox:~$ ls -l sample
-rw-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

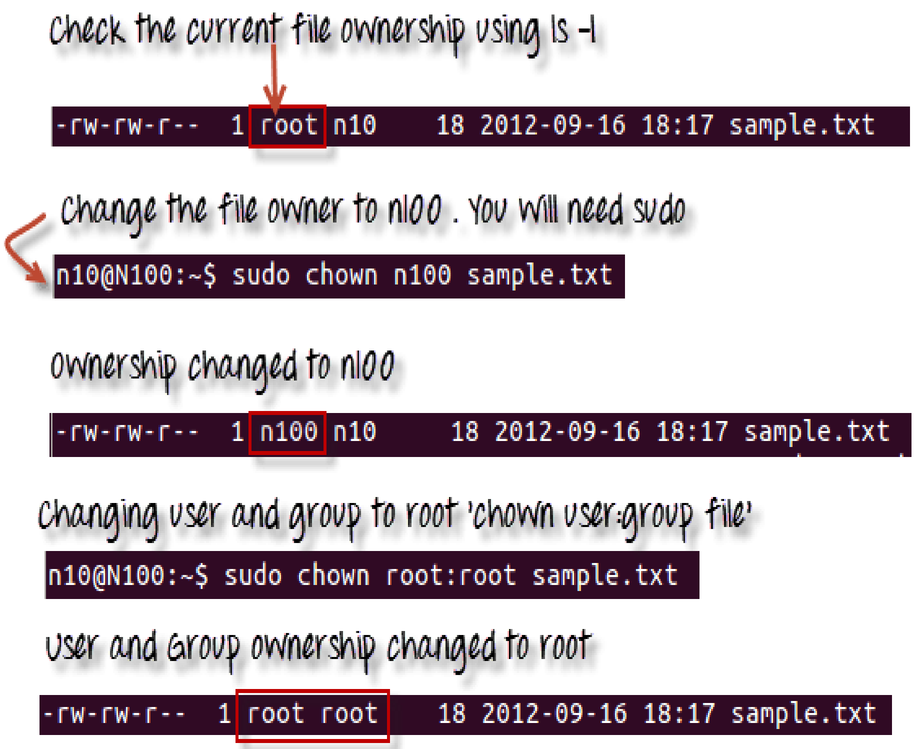
Removing 'read' permission for 'user'

```
home@VirtualBox:~$ chmod u-r sample
home@VirtualBox:~$ ls -l sample
--w-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Changing ownership

- For changing the ownership of a file/directory, you can use the following command:
  - Syntax: chown user
- To change the user as well as group for a file or directory use the command
  - Syntax: chown user: group filename

chown command



Check the current file ownership using ls -l

```
-rw-rw-r--   1 root n10     18 2012-09-16 18:17 sample.txt
```

Change the file owner to n100 . You will need sudo

```
n10@N100:~$ sudo chown n100 sample.txt
```

Ownership changed to n100

```
-rw-rw-r--   1 n100 n10      18 2012-09-16 18:17 sample.txt
```

Changing user and group to root 'chown user:group file'

```
n10@N100:~$ sudo chown root:root sample.txt
```

User and Group ownership changed to root

```
-rw-rw-r--   1 root root    18 2012-09-16 18:17 sample.txt
```

Linux Password file

- Traditional Linux systems keep user account information, including one-way encrypted passwords, in a text file called “/etc/passwd”
- As this file is used by many tools (such as ``ls'') to display file ownerships, etc. by matching user id #'s with the user's names, the file needs to be world-readable.

/etc/passwd file

- ``/etc/passwd'' file contains account information, and looks like this:

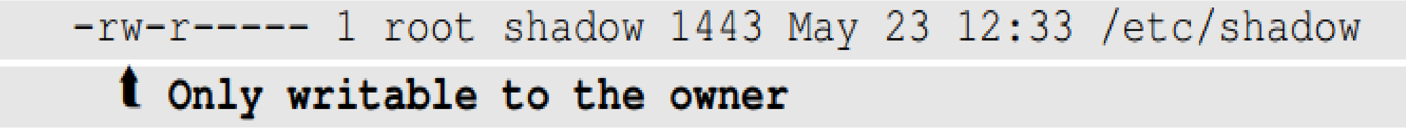
```
smithj:x:561:561:Joe Smith:/home/smithj:/bin/bash
```

Each field in a passwd entry is separated with ":" colon characters, and are as follows:

- Username, up to 8 characters. Case-sensitive, usually all lowercase
- An "x" in the password field. Passwords are stored in the ``/etc/shadow'' file.
- Numeric user id. This is assigned by the ``adduser'' script. Unix uses this field, plus the following group field, to identify which files belong to the user.
- Numeric group id. Red Hat uses group id's in a fairly unique manner for enhanced file security. Usually the group id will match the user id.
- Full name of user.
- User's home directory. Usually /home/username (eg. /home/smithj). All user's personal files, web pages, mail forwarding, etc. will be stored here.
- User's "shell account". Often set to ``/bin/bash'' to provide access to the bash

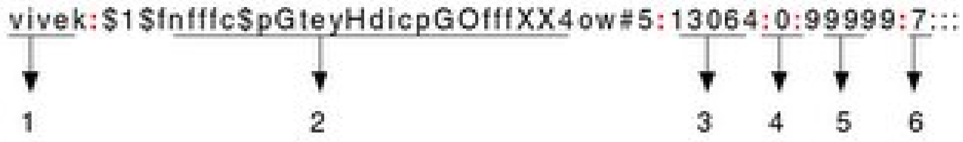
Need for Privileged Programs

- Password Dilemma
  - Permissions of /etc/shadow File:



```
   -rw-r----- 1 root shadow 1443 May 23 12:33 /etc/shadow
     ↑ Only writable to the owner
```

/etc/shadow file



```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::
```

1: Username: login name

2: Password: It is in encrypted form. Algorithms such as MD5, Blowfish, SHA-256, SHA-512 are used to store the password

3: Last Password changed: Days since 1st Jan 1970

4: Minimum: The minimum number of days required between password change

5: Maximum: The maximum number of days the password is valid. After that the user is forced to change his/her password

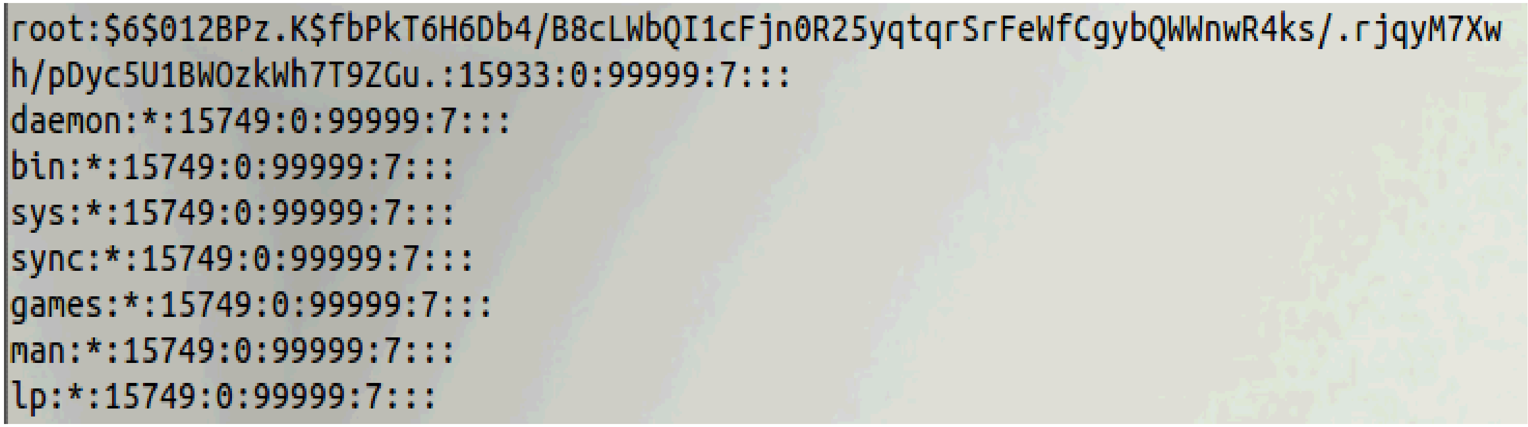
6: Warn: The number of days before the password is to expire that the user is warned that his/her password must be changed

7: Inactive: The number of days after password expires the account is disabled

8: Expire: An absolute date specifying when the login may no longer be used

Big Question

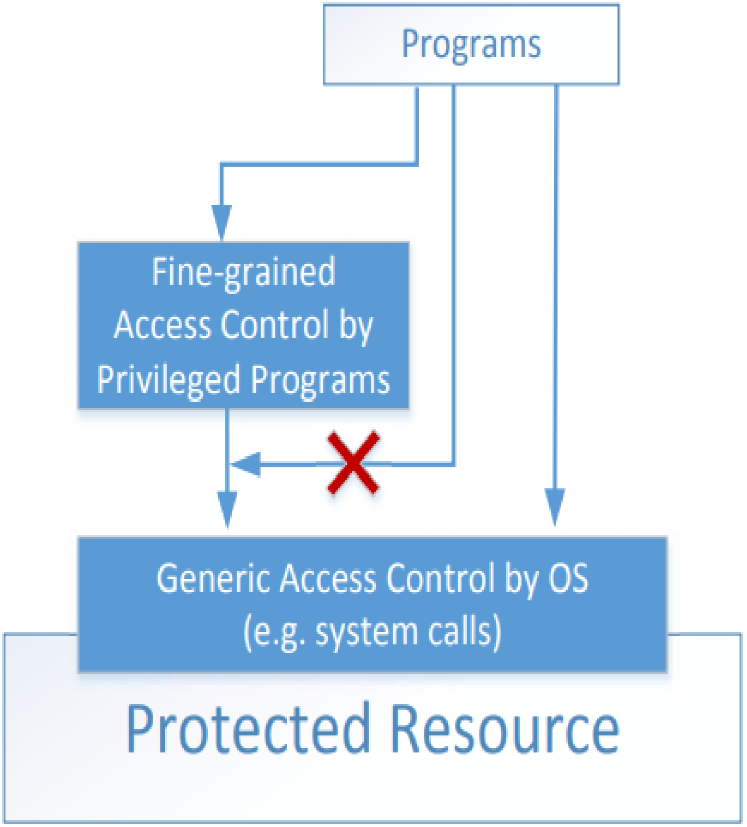
- How would normal users change their password?



```
root:$6$012BPz.K$fbPkT6H6Db4/B8cLWbQI1cFjn0R25yqtqrSrFeWfCgybQWWnwR4ks/.rjqyM7Xw
h/pDyc5U1BWOzkWh7T9ZGu.:15933:0:99999:7:::
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
sync:*:15749:0:99999:7:::
games:*:15749:0:99999:7:::
man:*:15749:0:99999:7:::
lp:*:15749:0:99999:7:::
```

Two-Tier Approach

- Implementing fine-grained access control in operating systems make OS over complicated.
- OS relies on extension to enforce fine-grained access control
- Privileged programs are such extensions


Programs
Fine-grained Access Control by Privileged Programs
Generic Access Control by OS (e.g. system calls)
Protected Resource

Types of Privileged Programs

- Daemons
  - Computer program that runs in the background
  - Needs to run as root or other privileged users

- Set-UID Programs
  - Invented by Dennis Ritchie
  - Widely used in UNIX systems
  - Program marked with a special bit

Superman Story

- Power Suit
  - Superpeople: Directly give them the power
  - Issues: bad superpeople

- Power Suit 2.0
  - Computer chip
  - Specific task
  - No way to deviate from pre-programmed task

- Set-UID mechanism: A Power Suit mechanism implemented in Linux OS

Set-UID Concept

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

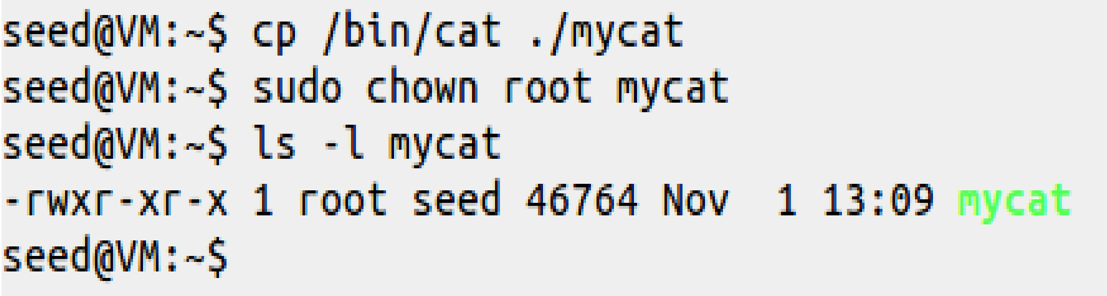
```
  $ ls -l /usr/bin/passwd
  -rwsr-xr-x 1 root root 41284 Sep 12  2012
/usr/bin/passwd
```

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
- When a normal program is executed, RUID = EUID, they both equal to the ID of the user who runs the program
- When a Set-UID is executed, RUID ≠ EUID. RUID still equal to the user’s ID, but EUID equals to the program **owner**’s ID.
  - If the program is owned by root, the program runs with the root privilege.

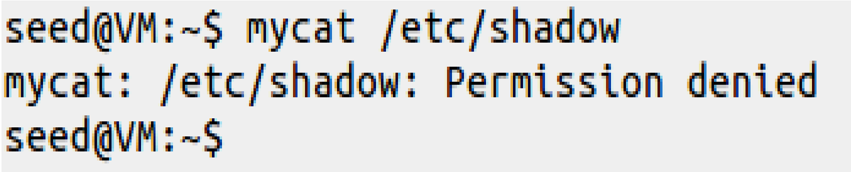
Turn a Program into Set-UID

- Change the owner of a file to root :



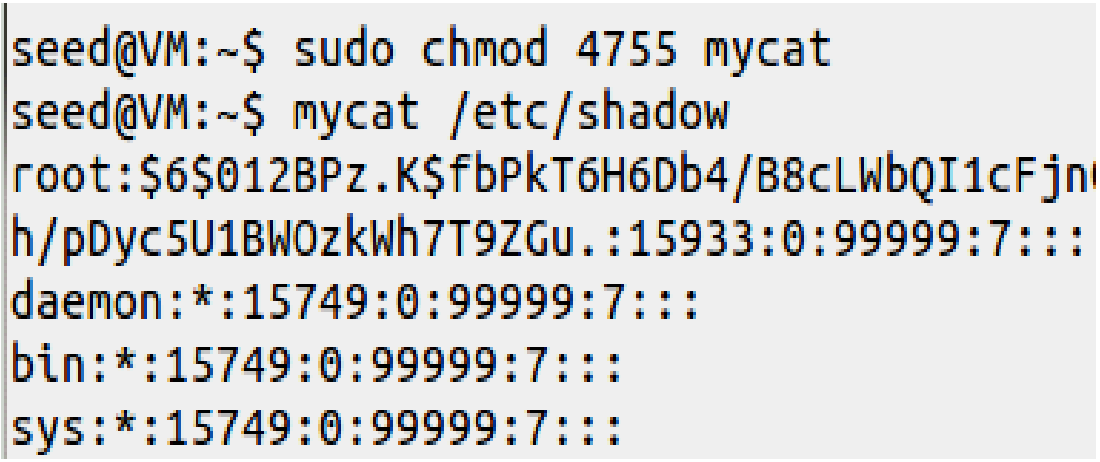
```
seed@VM:~$ cp /bin/cat ./mycat
seed@VM:~$ sudo chown root mycat
seed@VM:~$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Nov  1 13:09 mycat
seed@VM:~$
```

- Before Enabling Set-UID bit:



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

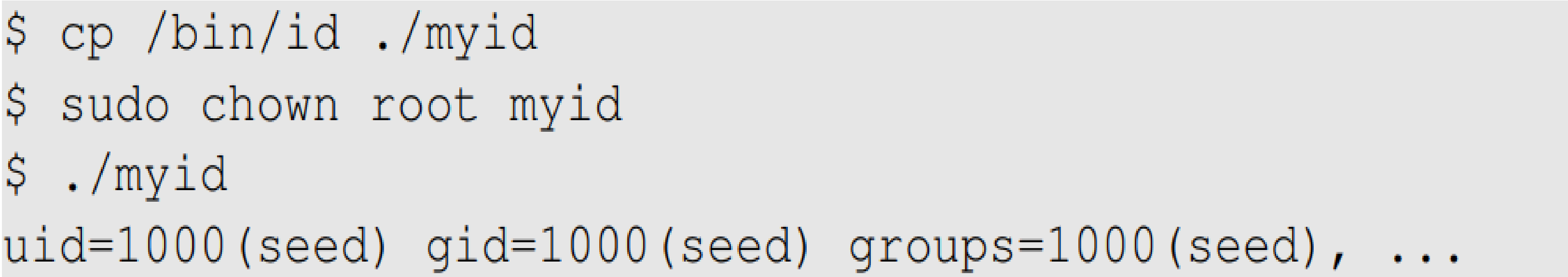
- After Enabling the Set-UID bit :



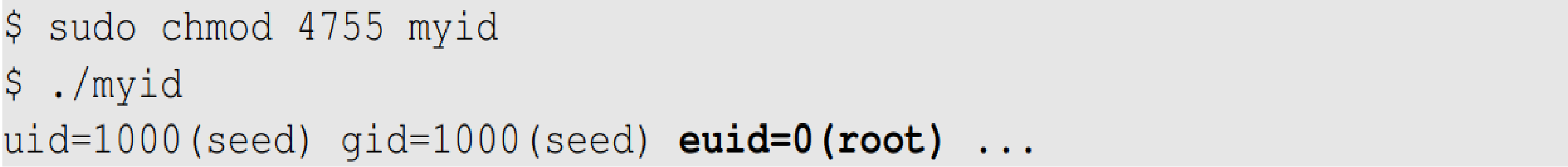
```
seed@VM:~$ sudo chmod 4755 mycat
seed@VM:~$ mycat /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8cLWbQI1cFjn
h/pDyc5U1BWOzkWh7T9ZGu.:15933:0:99999:7:::
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
```

How it 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

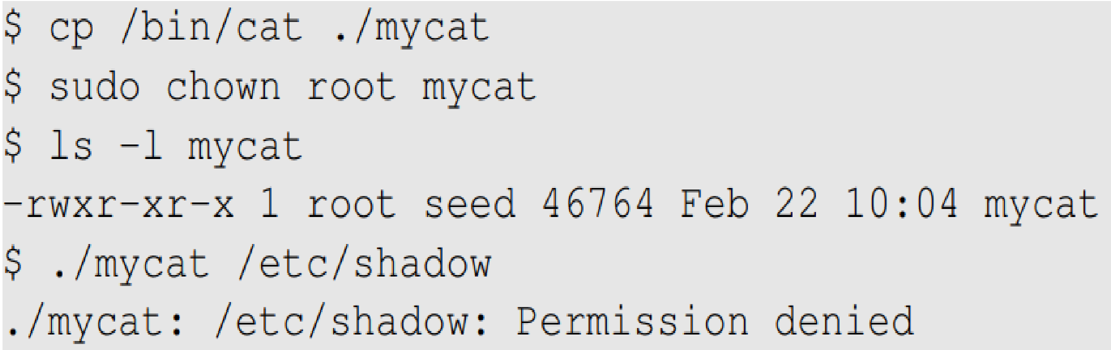


```
$ cp /bin/id ./myid
$ sudo chown root myid
$ ./myid
uid=1000(seed) gid=1000(seed) groups=1000(seed), ...
```



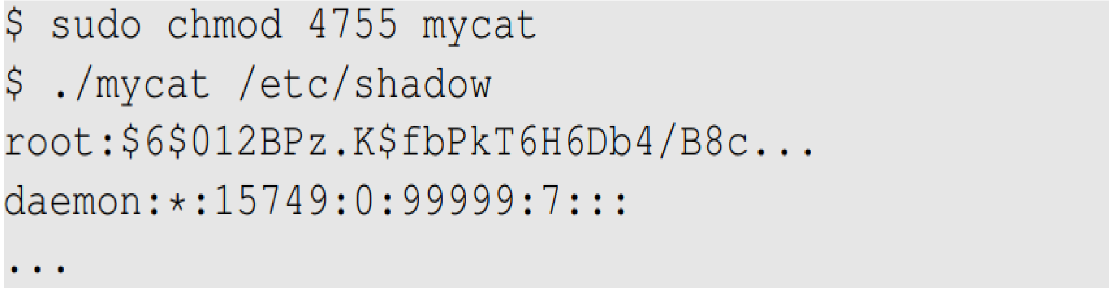
```
$ sudo chmod 4755 myid
$ ./myid
uid=1000(seed) gid=1000(seed) euid=0(root) ...
```

Example of Set UID



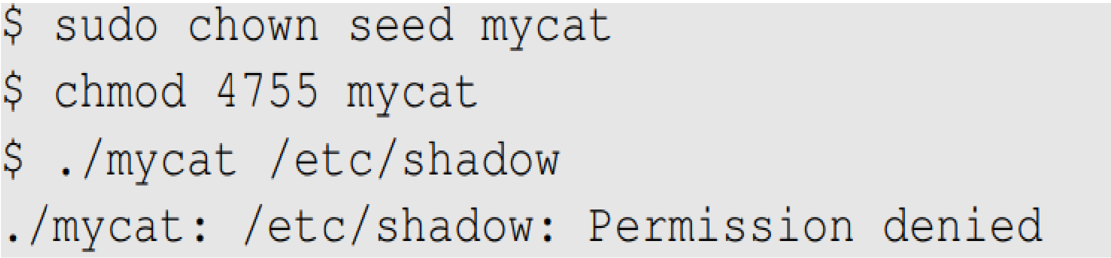
```
$ cp /bin/cat ./mycat
$ sudo chown root mycat
$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Feb 22 10:04 mycat
$ ./mycat /etc/shadow
./mycat: /etc/shadow: Permission denied
```

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

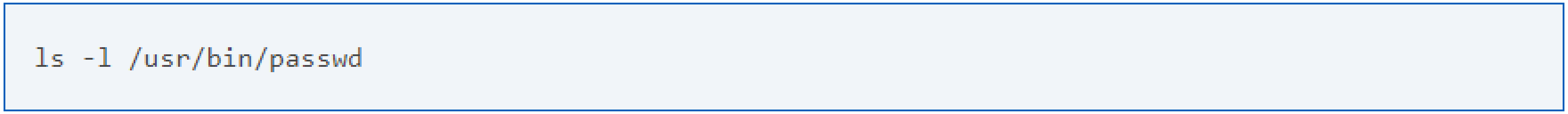


```
$ sudo chown seed mycat
$ chmod 4755 mycat
$ ./mycat /etc/shadow
./mycat: /etc/shadow: Permission denied
```

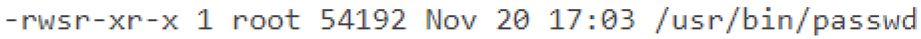
It is still a privileged program, but not the root privilege

Set UID

- When an executable file's setuid permission is set, users may execute that program with a level of access that matches the user who owns the file.
- When viewing a file's permissions with the **ls - l** command, the setuid permission is displayed as an "**s**" in the "user execute" bit position.



```
ls -l /usr/bin/passwd
```



```
-rwsr-xr-x 1 root 54192 Nov 20 17:03 /usr/bin/passwd
```

- To set the set-uid bit

```
chmod u+s myfile
```

- Non-executable files can be marked as set-uid, but it has no effect;

```
ls -l myfile
```

```
-rw-r--r-- 1 user 0 Mar 6 10:45 myfile
```

```
chmod u+s myfile
```

```
ls -l myfile
```

```
-rwSr--r-- 1 user 0 Mar 6 10:45 myfile
```

Uppercase letter

If we change the permission to **u+x**, then the set-uid permission comes into effect.

How is Set-UID Secure?

- Allows normal users to escalate privileges
  - This is different from directly giving the privilege (sudo command)
  - Restricted behavior – similar to superman designed computer chips

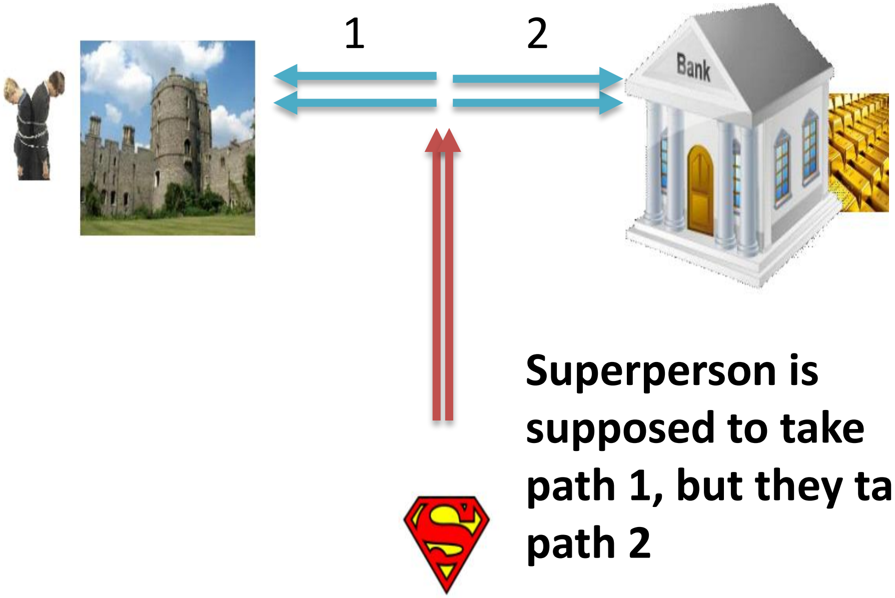
- Unsafe to turn all programs into Set-UID
  - Example: /bin/sh
  - Example: vi

Attack on Superman

- Cannot assume that user can only do whatever is coded
  - Coding flaws by developers

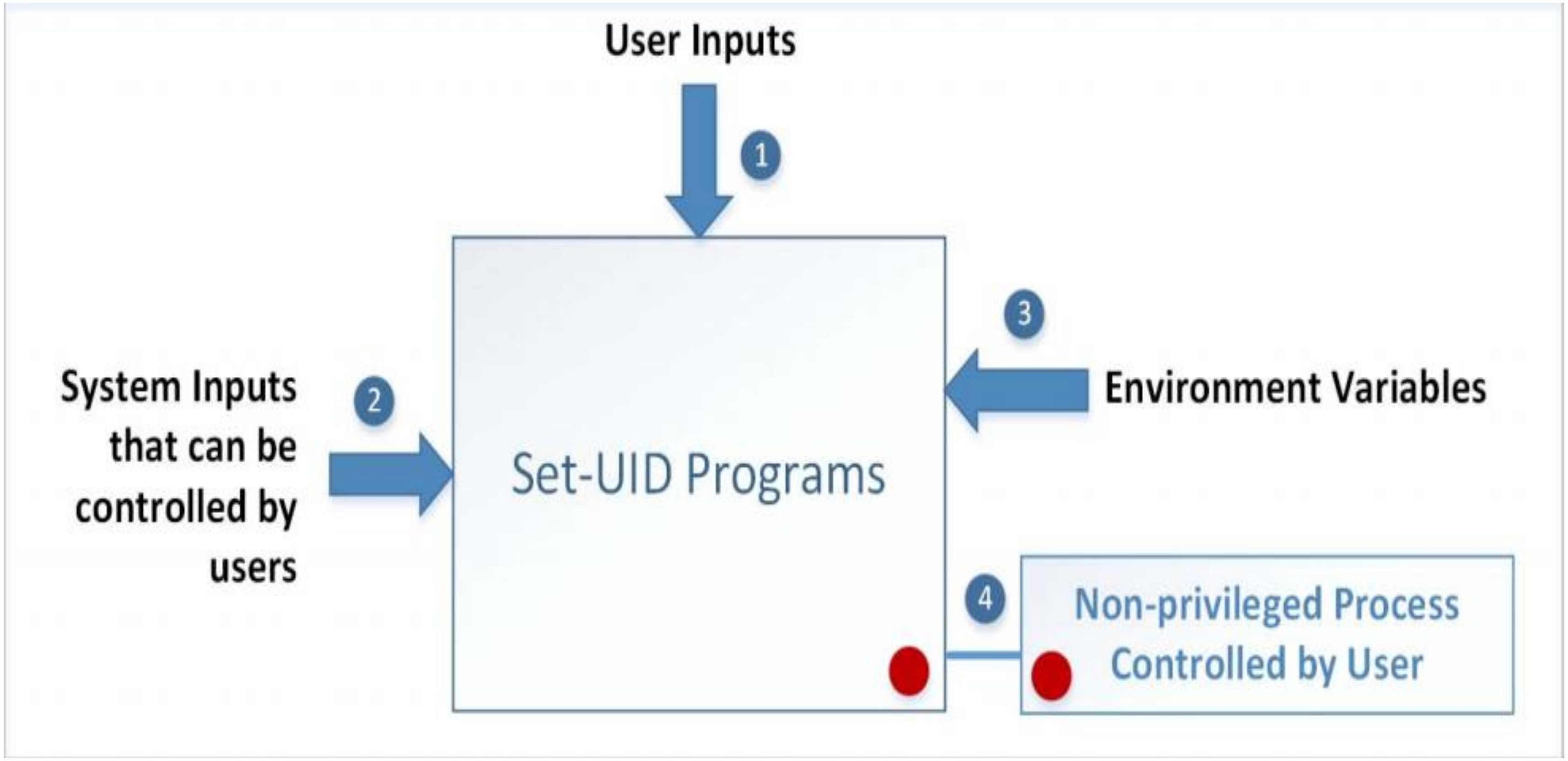
- Superperson Mallory
  - Fly north then turn left
  - How to exploit this code?

- Superperson Malorie
  - Fly North and turn West
  - How to exploit this code?



Superperson is supposed to take path 1, but they take path 2

Attack Surfaces of Set-UID Programs


User Inputs
1
System Inputs that can be controlled by users
2
Set-UID Programs
3
Environment Variables
4
Non-privileged Process Controlled by User

Attacks via User Inputs

User Inputs: Explicit Inputs

- Buffer Overflow
  - Overflowing a buffer to run malicious code
- Format String Vulnerability
  - Changing program behavior using user inputs as format strings

Attacks via User Inputs

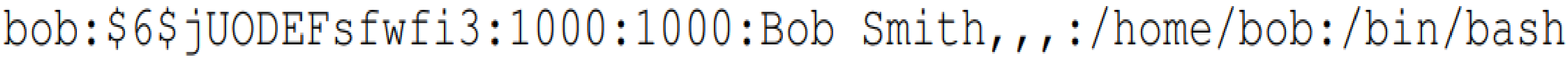
CHSH – Change Shell

- Set-UID program with ability to change default shell programs
- Shell programs are stored in /etc/passwd file

Issues

- Failing to sanitize user inputs
- Attackers could create a new root account

Attack



```
bob:$6$jUODEFsfwfi3:1000:1000:Bob Smith,,,:/home/bob:/bin/bash
```

Attacks via System Inputs

System Inputs

- Programs may get input from the underlying systems

  – Race Condition

    - Symbolic link to privileged file from a unprivileged file
    - Influence programs
    - Writing inside world writable folder

Attacks via Environment Variables

- Behavior can be influenced by inputs that are not visible inside a program.
- Environment Variables : These can be set by a user before running a program.
- Detailed discussions on environment variables will be done later.

Attacks via Environment Variables

- `PATH` Environment Variable
  - Used by shell programs to locate a command if the user does not provide the full path for the command
  - system(): call /bin/sh first
  - system(“ls”)
    - /bin/sh uses the PATH environment variable to locate “ls”
    - Attacker can manipulate the PATH variable and control how the “ls” command is found
- More examples on this type of attacks will be presented later

Capability Leaking

- In some cases, Privileged programs downgrade themselves during execution
- Example: The `su` program
  - This is a privileged Set-UID program
  - Allows one user to switch to another user (say user1 to user2)
  - Program starts with EUID as root and RUID as user1
  - After password verification, both EUID and RUID become user2’s (via privilege downgrading)
- Such programs may lead to capability leaking
  - Programs may not clean up privileged capabilities before downgrading