System Security Basics

Access Control and Permission

- Access Control
  - Operating System (OS) needs to know whether a **user (or group)** is allowed to access the **resource** (e.g., file).
  - In Linux, this is done by looking at a policy called Access Control List.
- User and Group
  - User ID (UID)
  - Group ID (GID)
- superuser **Root**
  - UID is 0,
- What would you try to do if you were attacking the system?

Other Basics

- Linux Permissions
    - What read, write, execute(i.e., rwxrwxrwx) means.
        - Permission on file
        - Permission on folder
            - r: list the content of the folder
            - w: create files or folders in a folder
            - x: enter the folder
    - Default Permissions: umask command
        - user file-creation mode
        - Question) umask 0077 is considered to more secure over 0002. Please explain why.

Other Basics

- Linux Permissions (Continue..)

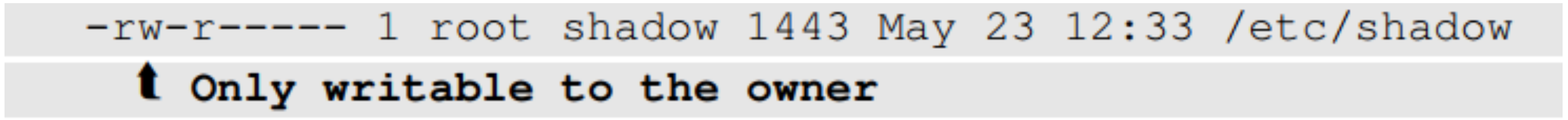
  - Other commands you should know: chown, sudo, sudo –u, whoami, etc.
  - Files to look at: /etc/sudoer, /etc/passwd, /etc/shadow

  - Full Access Control List: setfacl and getfacl
    - Question) Explain what Full Access Control List in few sentences.

Set-UID Privileged Programs

Need for Privileged Programs

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



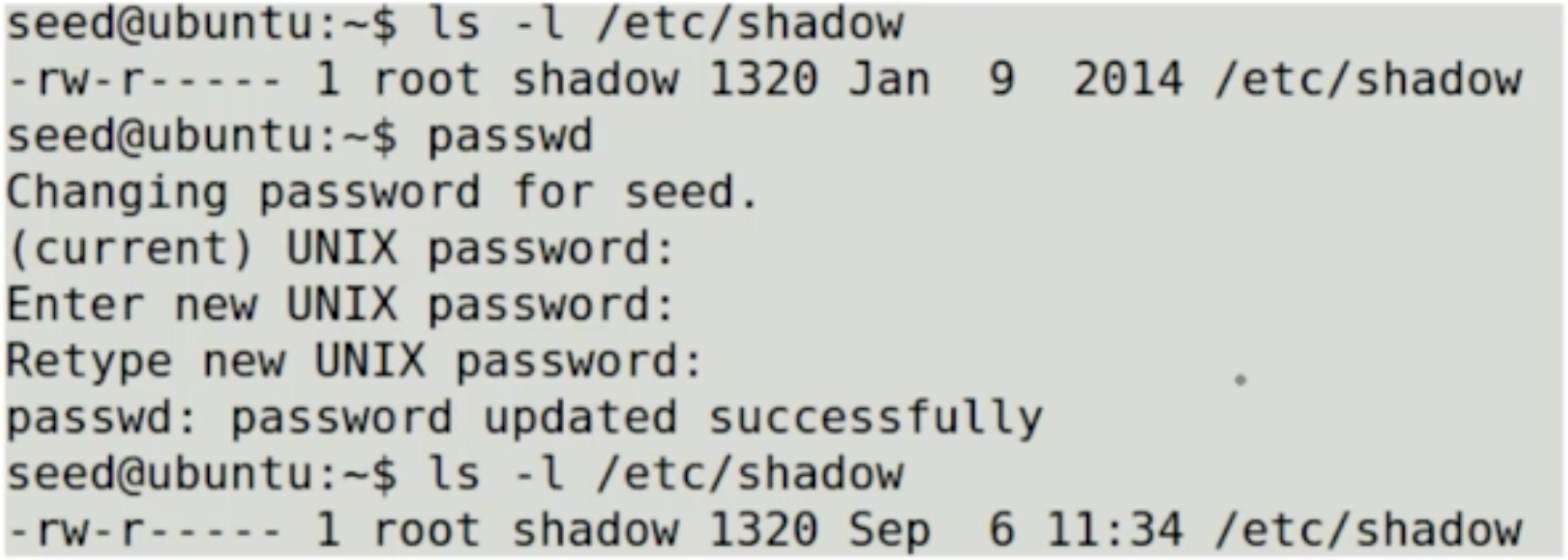
```
-rw-r----- 1 root shadow 1443 May 23 12:33 /etc/shadow
    ↑ Only writable to the owner
```

  - How such tasks are usually done:
    - Privileged programs take request from users and perform the tasks on behalf of the users.
    - However, privileged background process needs to wait for users.
    - In the old days, memory (and computing power) was expensive.

      → A new mechanism called **Set-UID** was developed!

Need for Privileged Programs

- Password Dilemma
  - Example)



```
seed@ubuntu:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1320 Jan  9  2014 /etc/shadow
seed@ubuntu:~$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
seed@ubuntu:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1320 Sep  6 11:34 /etc/shadow
```

Superman Story

- Power Suit V1.0
  - Superpeople: Directly give them the power
  - Issues: bad superpeople
- Power Suit V2.0
  - 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

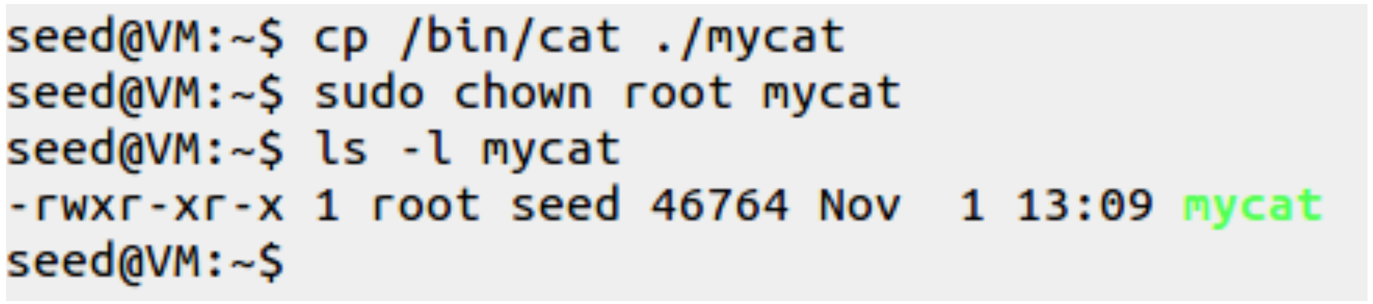
```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 41284 Sep 12  2012 /usr/bin/passwd
```

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.

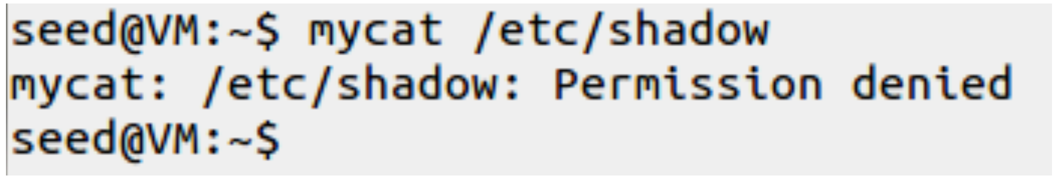
Turn a Program into Set-UID

- Change the owner of a file to root :



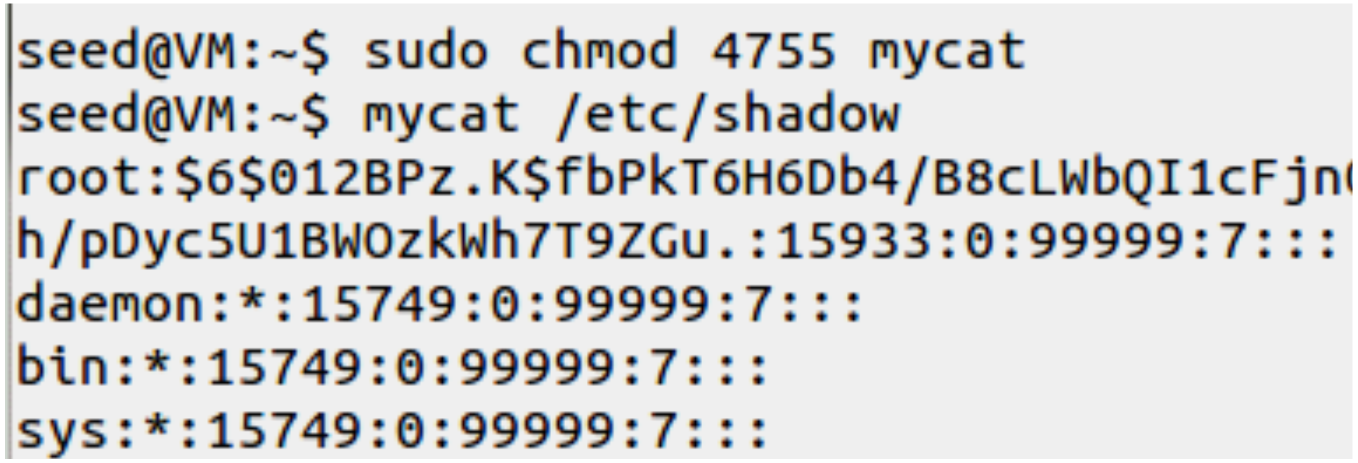
```
seed@VM:~$ cp /bin/cat ./mycat
seed@VM:~$ sudo chown root mycat
seed@VM:~$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Nov  1 13:09 mycat
seed@VM:~$
```

- Before Enabling Set-UID bit:



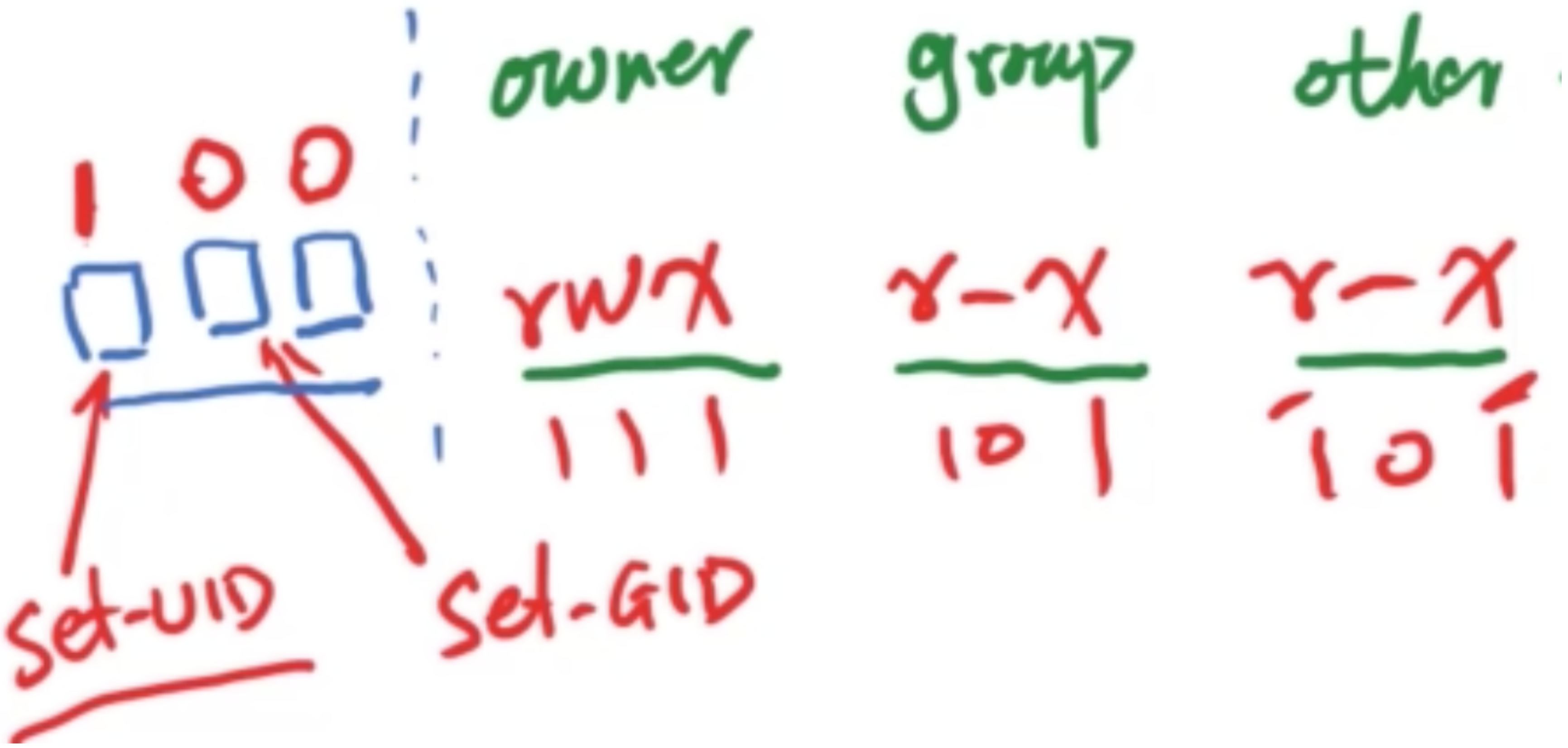
```
seed@VM:~$ mycat /etc/shadow
mycat: /etc/shadow: Permission denied
seed@VM:~$
```

- 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:::
```

Turn a Program into Set-UID



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

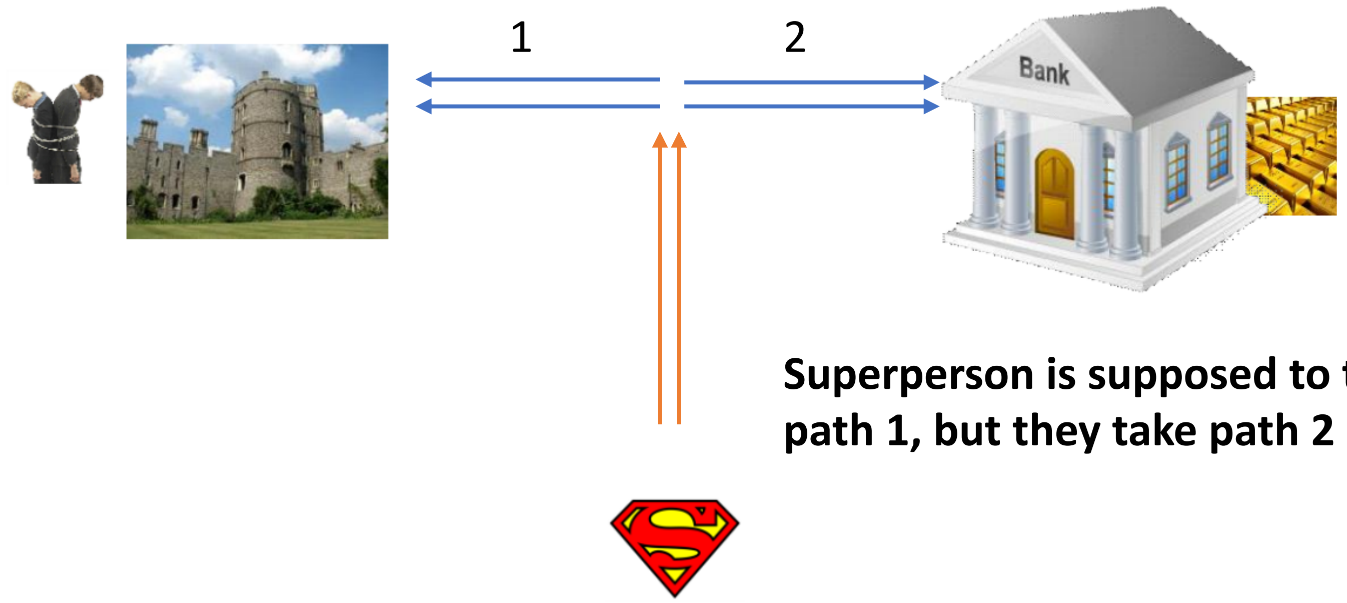
How is Set-UID Secure?

- 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 Mallroy
  - Fly north then turn left
  - How to exploit this code?
- Superperson Malorie
  - Fly North and turn West
  - How to exploit this code?

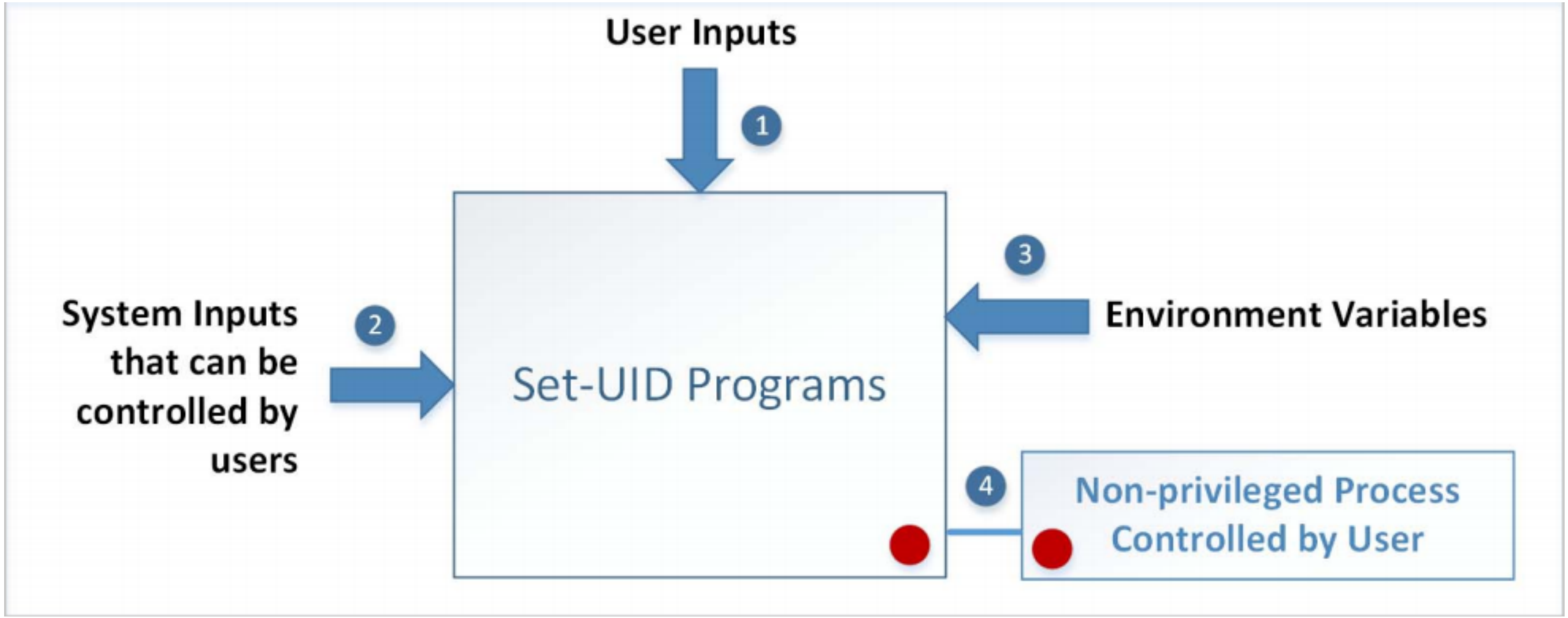


Superperson is supposed to take path 1, but they take path 2

Attack Surface

- **Description**
  - The attack surface of a software environment is the sum of the different points where an unauthorized user can try to enter data to or extract data from an environment. Keeping the attack surface as small as possible is a basic security measure.

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 – More information in Future lecture and lab
  - Overflowing a buffer to run malicious code
- Format String Vulnerability – More information in Future lecture and lab
  - 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

- Race Condition – More information in Future lecture and lab
  - Symbolic link to privileged file from an 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 in next week lab.

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 can be found in next week lab.

Few questions

- Question) umask 0077 is considered to more secure over 0002. Please explain why.
- Question) Explain what Full Access Control List in few sentences.