Privileged Programs & Program Inputs

The Set-UID Mechanism & Environment Variables (Part I)

Professor Travis Peters
CSCI 476 - Computer Security
Spring 2020

Some slides and figures adapted from Wenliang (Kevin) Du's
Computer & Internet Security: A Hands-on Approach (2nd Edition).
Thank you Kevin and all of the others that have contributed to the SEED resources!

Today

Announcements

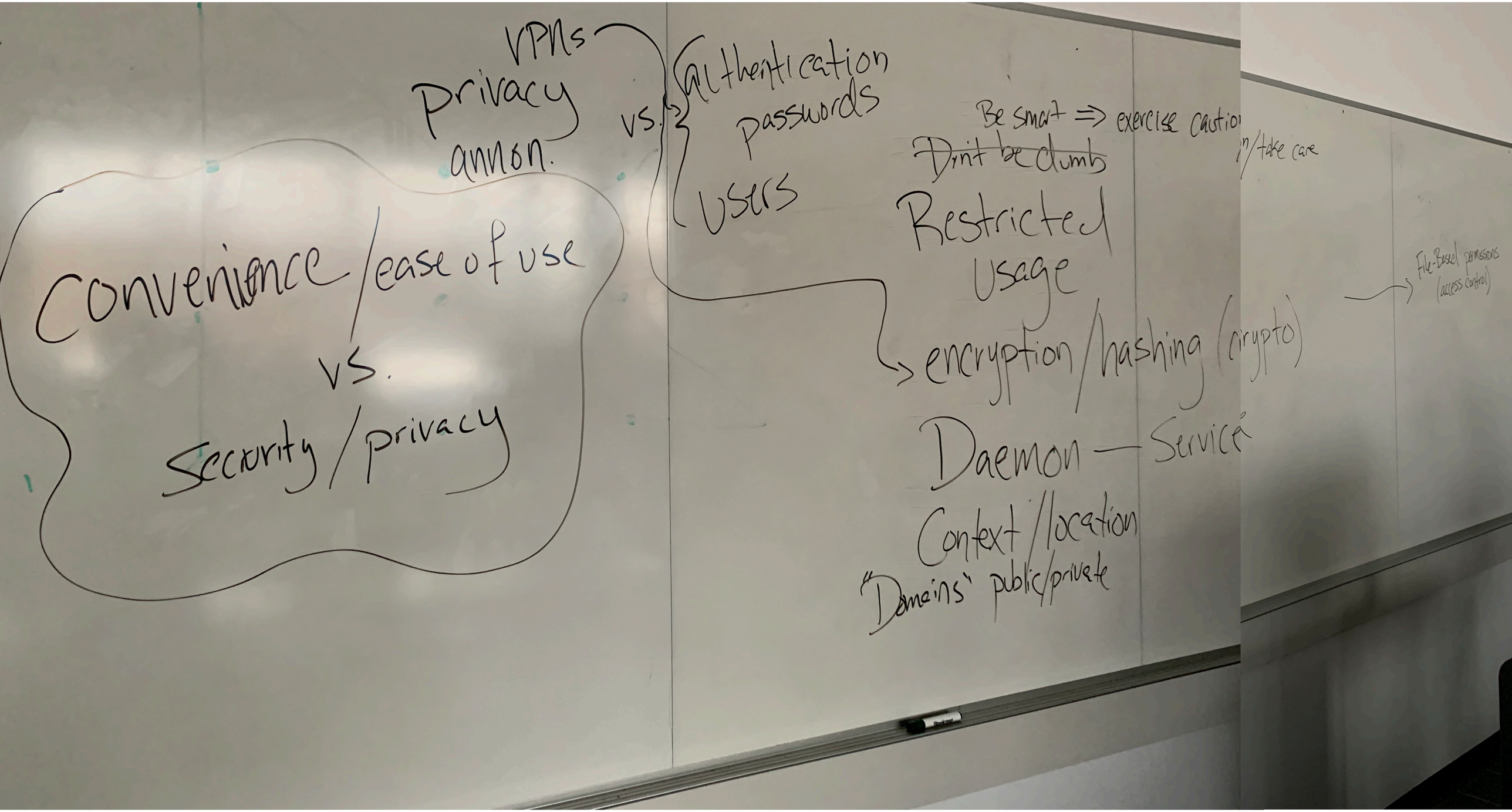
- We need a note taker for the class! ➡ ***Contact ODS if interested***
- Lab 00 ➡ ***It's up!***
- Lab 01 ➡ ***Will be posted Thursday***
- ***REMINDER:*** Bring your laptops (especially on Thursdays!)
- ***REMINDER:*** Sign-up for Slack ASAP!

Goals & Learning Objectives

- Wrap up review of basic Linux commands + basic ideas in Linux security
- Understand the need for privileged programs
- Understand how the Set-UID mechanism works + attack surface of (Set-UID) programs

How would you protect your computer & its resources?

Ideas?!


VPNs
Privacy
anon.
vs.
Authentication
passwords
users
Convenience / ease of use
vs.
Security / privacy
Be smart ⇒ exercise caution
~~Don't be dumb~~
Restricted
usage
encryption / hashing (crypto)
Daemon — Service
Context / location
"Domains" public/private
in/take care
File-Based permissions
(access control)

How would you protect your computer & its resources?

Modeling and managing system security the UNIX-y way: file permissions & access control

who can do **what** to **whom**

users/groups

Need notions of "identity"

A human?

Groups of humans?

A service?

An administrator? (e.g., "root")

objects

Usually *things* on a *filesystem*

(e.g., programs, data)

permissions (read/write/execute)

*OK, I know the who/whom—what are **you permitted to do**?*

Read the file?

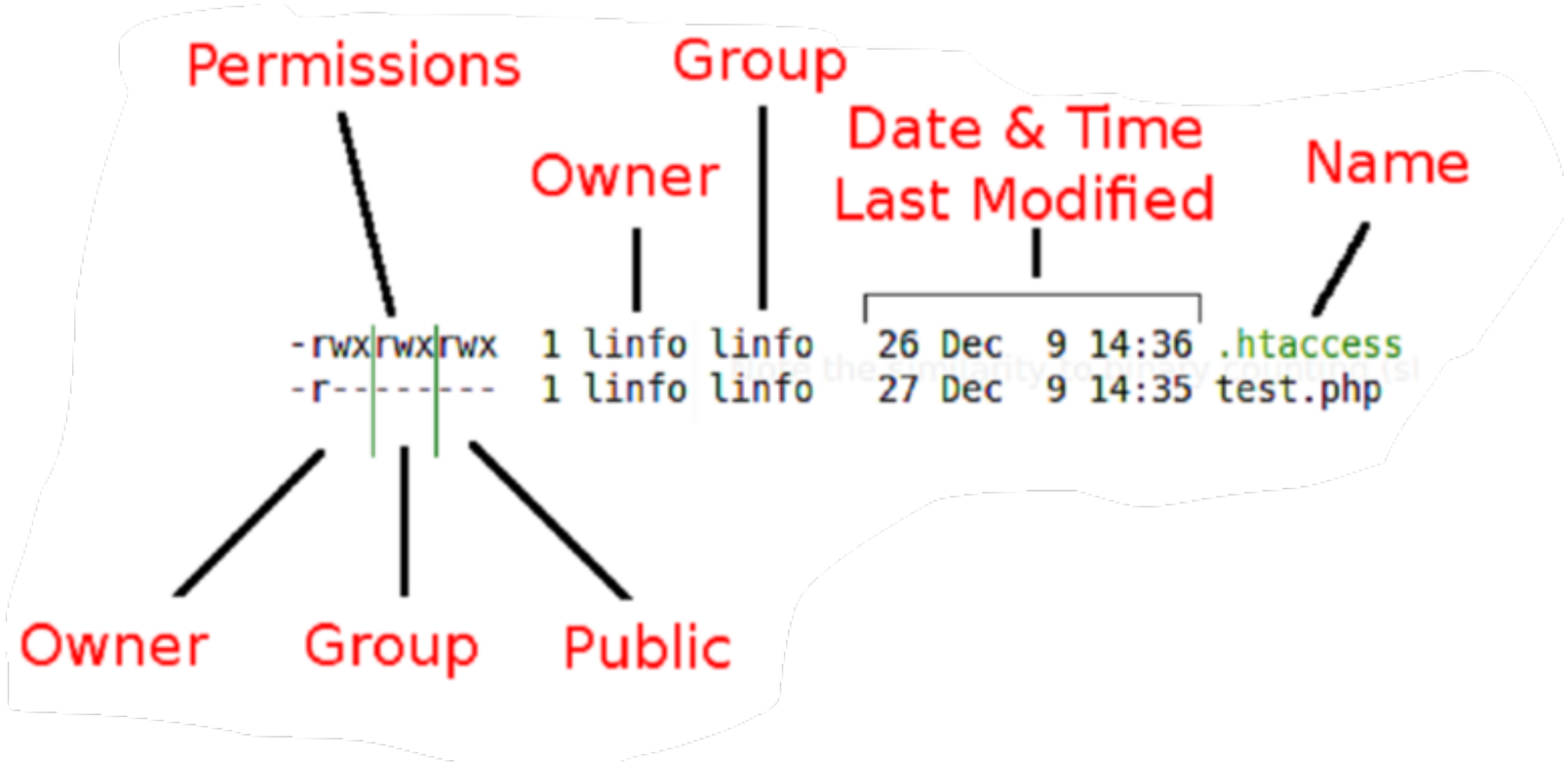
Write to it/modify it?

Run it?

How would you protect your computer & its resources?

Modeling and managing system security the UNIX-y way: file permissions & access control

Every file has…



A Typical **who** can do **what** to **whom** Flow

If **user A** asks to perform **operation O** on a **file object F**, the OS checks:

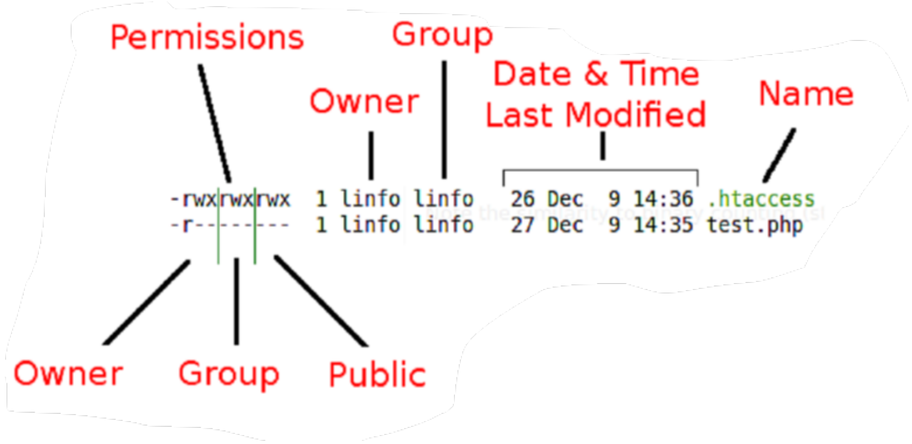
1. Is **A** the owner of **F**? >>> use **owner permissions** to decide whether A can do operation O.

   A is not F's owner

2. Is **A** a member of **F's** group? >>> use **group permissions** to decide…

   A is not F's owner or a member of F's group

3. >>> use the **"everyone else" / "others" permissions** to decide…



Some in class exploration — let's take a quick look at these ideas in a VM.

The Limitations of File-Based Access Control

Question

How can a non-privileged user ‘champ’ change their own password?

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

Only writable to the owner

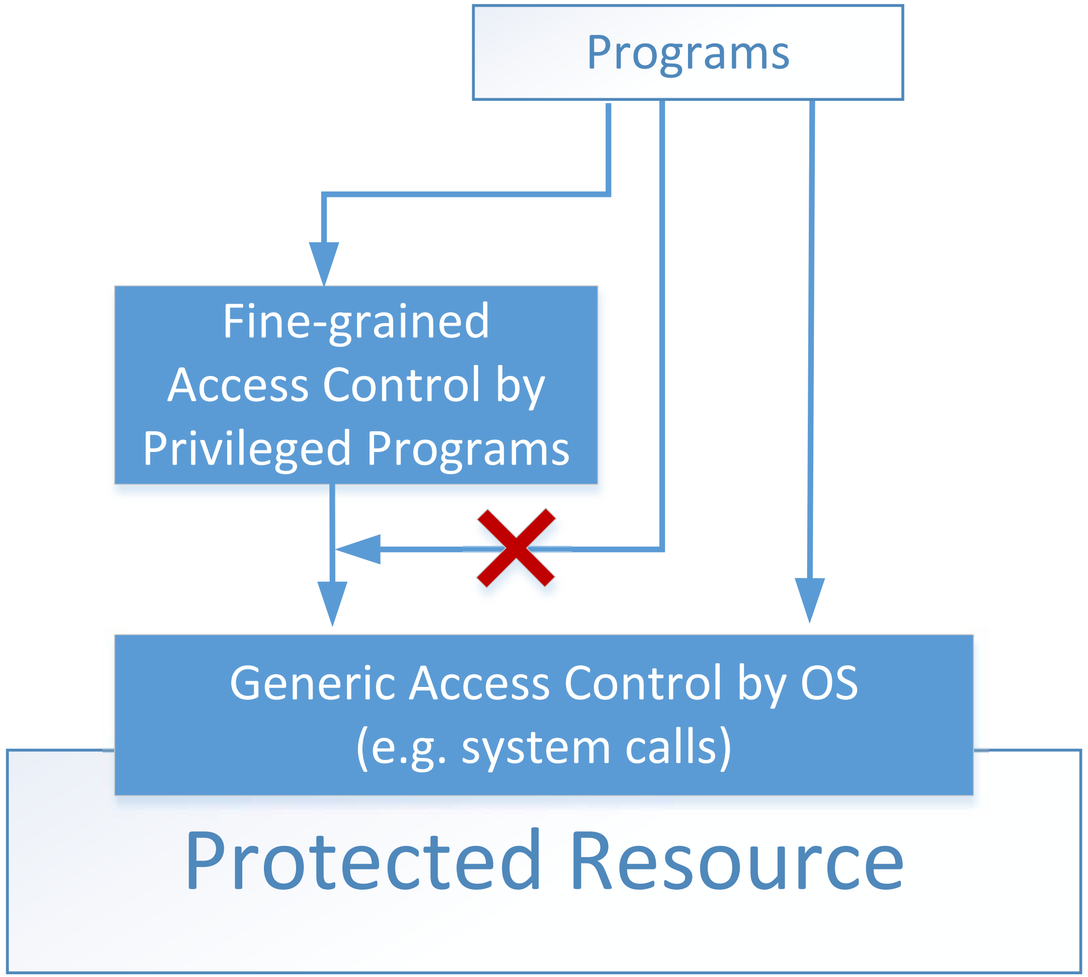
- **Idea:** grant access to user ‘champ’ to edit /etc/shadow
  Good idea or bad idea?

- **Better Idea:** grant access to user ‘champ’ to edit ONLY its own entry in /etc/shadow
  Good idea or bad idea?

- **BETTER Idea:** let user ‘champ’ run a special program that has one purpose:
  to edit the /etc/shadow file with inputs to the program
  Good idea or bad idea?

Two-Tier Approach to Access Control

- Implementing fine-grained access control in the OS makes OS code really complicated
  (we generally try to avoid this…)
- OS relies on other extensions/features to enforce fine-grained access control
- "Privileged programs" are such extensions



Types of Privileged Programs

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

- **Set-UID Programs**
  - Widely used in UNIX systems
  - A normal program… but marked with a special bit

Superman's Past *(The Stories You Never Heard…)*

- **Superman's 1st Attempt—The Power Suit**
  - **Superman** got tired of saving *everyone*
  - **Super*people*:** give normal people superman's power!
  - **Problem:** not all superpeople are good…
- **Superman's 2nd Attempt—Power Suit 2.0**
  - Power suit w/ a sweet computer in it
  - Power suit can only perform a specific task
  - No way to deviate from the pre-programmed task……

The Set-UID mechanism is a lot like superman's power suit 2.0 (but implemented in UNIX systems…)

Set-UID In A Nutshell

There is also a Set-GID (Set Group ID), which works in basically the same way, but applies to group privileges

- Allow user to run a program with the program ***owner's*** privilege
  - i.e., a UNIX mechanism for changing user/group identity
  - Allows users to run programs w/ temporarily elevated privileges

- Created to deal with inflexibilities of UNIX access control
  - *Why might this be useful?*
  - *Why might this be a bad idea?*

- Example: the **`passwd`** program

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 41284 Jan 21  2020 /usr/bin/passwd
```

Set-UID In A Nutshell *(cont.)*

- Every process has two User IDs
  - **Real UID (RUID)** — identifies the owner of the process
  - **Effective UID (EUID)** — identifies privilege of the process
  - Access control decisions are based on EUID!

- When a normal program is executed,

  RUID == EUID (== user who ***runs*** the program)

- When a Set-UID program is executed,

  RUID != EUID ➡ EUID == ID of program's owner

If program owner == root, the program runs with root privileges

Demos! *If we have time… **else** look at some slides*

So Uh… How Do You Set… The Set-UID Bit Thingy

- **Change the owner** of a file to root

```
seed@VM:~/csci_sandbox$ cp /bin/cat ./mycat
seed@VM:~/csci_sandbox$ sudo chown root mycat
seed@VM:~/csci_sandbox$ ls -al mycat
-rwxr-xr-x 1 root seed 51036 Jan 21 02:02 mycat
```

- **Before** enabling the Set-UID bit

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

- **After** enabling the Set-UID bit

```
seed@VM:~/csci_sandbox$ sudo chmod 4755 mycat
seed@VM:~/csci_sandbox$ mycat /etc/shadow
root:$6$NrF4601p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
```

How it Works

A Set-UID program is just like any other program, except that it has a special bit set *(the Set-UID bit)*

```
seed@VM:~/csci_sandbox$ cp /usr/bin/id ./myid
seed@VM:~/csci_sandbox$ sudo chown root myid
seed@VM:~/csci_sandbox$ ./myid
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),
```

```
seed@VM:~/csci_sandbox$ sudo chmod 4755 myid
seed@VM:~/csci_sandbox$ ./myid
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(
```

Is Set-UID Secure?

- Allows normal users to escalate privileges
  - This is different from directly giving escalated privileges (e.g., **`sudo`**)
  - Restricted behavior — similar to superman's power suit

- Unsafe to run all programs as Set-UID programs……

  Examples?
  - `/bin/sh` — *why?*
  - `vi(m)` — *why?*

Software (and Superman's PS…) Is Only As Good As We Make It…

- Shouldn't assume that user can only do whatever is coded…
- ***Recall Superman: What sorts of attacks are possible on Superman's PS2.0?***
- **Mallory (v1)**
  - Fly north, turn left,
    knock down wall, capture bad guy
    Where could this go wrong?
- **Mallory (v2)**
  - Fly north, *turn west*,
    knock down wall, capture bad guy
    Where could this go wrong?