# Software Security Assessment

Race Condition

## Concurrency and Race condition

**Concurrency**

- Execution of Multiple flows (threads, processes, tasks, etc)
- If not controlled can lead to nondeterministic behavior

**Race Condition**: an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence in order to be done correctly

E.g., two people simultaneously try to modify the same account (withdrawing money)

## Race condition

- **Necessary properties for a race condition**
  - **Concurrency property**
    - At least two control flows executing concurrently
  - **Shared object property**
    - The concurrent flows must access a common shared race object
  - **Change state property**
    - At least one control flow must alter the state of the race object

## Race window

- A code segment that accesses the race object multiple times opens a window of opportunity for race condition

- ToCToU race conditions
  - Can occur during file I/O
  - Forms a RW by first checking some race object and then using it

## Example Program

SetUID Program

```
If (!access (“/tmp/XYZ”, W_OK)) {
    /* thre real user ID has access right*/
    f = open(“/tmp/XYZ”, O_WRITE);
    write_to_file(f);
}else{
    /* thre real user ID does not access right*/
    fprintf(stderr, “Permission Denied\n”);
}
```

Where is the potential window for race condition?

access(): check whether the real user ID has access to the file, even the program is a setuid program

## Vulnerable Program

SetUID Program

```
If (!access ("/tmp/XYZ", W_OK)) {
    /* thre real user ID has access right*/
    f = open("/tmp/XYZ", O_WRITE);
    write_to_file(f);
}else{
    /* thre real user ID does not access right*/
    fprintf(stderr, "Permission Denied\n");
}
```

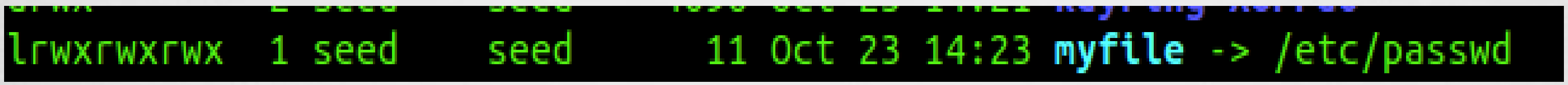
How can we use the race window to do something malicious?

Hint: There are many process running on the computer at the same time

## Symbolic link

- Unix/Linux symbolic linking is most common
  - Symlink is a directory entry that references a target file or directory

  - ln -s target linkName
  - Example: ln -s /etc/passwd /tmp/myfile, myfile is linked to /etc/passwd



```
lrwxrwxrwx  1 seed    seed      11 Oct 23 14:23 myfile -> /etc/passwd
```

-s, --symbolic make symbolic links instead of hard links
-f, --force remove existing destination files

More info about ln: https://linux.die.net/man/1/ln

## Vulnerable Program

SetUID Program

```
1. If (!access ("/tmp/XYZ", W_OK)) {
   /* thre real user ID has access right*/
2.    f = open("/tmp/XYZ", O_WRITE);
3.    write_to_file(f);
4. }else{
   /* thre real user ID does not access right*/
5.    fprintf(stderr, "Permission Denied\n");
}
```

**Before Step 1**: Link XYZ to a file that you have permission

**How?**

**Between Step 1 and 2**: Link XYZ to the file you are interested

## Vulnerable Program

SetUID Program

```
1. If (!access ("/tmp/XYZ", W_OK)) {
   /* thre real user ID has access right*/
2.    f = open("/tmp/XYZ", O_WRITE);
3.    write_to_file(f);
4. }else{
   /* thre real user ID does not access right*/
5.    fprintf(stderr, "Permission Denied\n");
}
```

Recursively link XYZ to myfile and targeted file…

## Question?

SetUID Program

```
If (!access (“/etc/shadow”, W_OK)) {
   f = open(“/etc/shadow”, O_WRITE);
   write_to_file(f);
}else{
    fprintf(stderr, “Permission Denied\n”);
}
```

Does the above SetUID program have a race condition vulnerability?

## Protection Mechanisms

```
1. If (!access ("/tmp/XYZ", W_OK)) {
   /* thre real user ID has access right*/
2.    f = open("/tmp/XYZ", O_WRITE);
3.    write_to_file(f);
4. }else{
   /* thre real user ID does not access right*/
5.    fprintf(stderr, "Permission Denied\n");
}
```

Think about protection mechanisms…

## Protection Mechanisms

```
1. If (!access (“/tmp/XYZ”, W_OK)) {
   /* thre real user ID has access right*/
2.    f = open(“/tmp/XYZ”, O_WRITE);
3.    write_to_file(f);
4. }else{
   /* thre real user ID does not access right*/
5.    fprintf(stderr, “Permission Denied\n”);
}
```

Mechanism A: Build in Portection

sudo sysctl -w fs.protected_symlinks=1

Once on: the source and target of symlink should have the same owner

## Protection Mechanisms

```
If (!access (“/tmp/XYZ”, W_OK)) {
   /* thre real user ID has access right*/
   f = open(“/tmp/XYZ”, O_WRITE);
   write_to_file(f);
}else{
   /* thre real user ID does not access right*/
   fprintf(stderr, “Permission Denied\n”);
}
```

More protections: Checkout our Lab~