시스템 및 네트워크 보안
- Mini Project #01 -

제출일자	2018.04.09.
분 반	00
이 름	신종욱
학 번	201302423

Requirements

- 특정 파일(들)에 대하여 두 사용자만 공유함
- 해당 파일에 적용되는 연산은 읽기와 추가(append)
- 추가는 redirection의 형태로 구현 (예) % cat >> secret-file

Tasks

1. Permission만 이용하여 접근을 제한하는 체계를 구성함. 별도의 프로그램을 활용하지 않음

만족할 수 없다.

일단 파일의 소유자는 모든 것을 할 수 있고 공유를 할려면 다른 사용자가 해당 조건을 만족해야한다.
User 권한단계에서 Permission만 이용하여서는 접근 제한은 소유자,그룹원,나머지 사용자들 이렇게 세 분류로 할 수 있는데 파일을 공유할려면 그룹원, 나머지 사용자들을 통해 읽기와 쓰기에 권한을 주면 **2명 외 다른 그룹원 혹은 나머지 사용자들도 접근이 가능해진다.**

만약 관리자권한으로 특정사용자 둘만 그룹화시킬 수 있다면 가능하다 이 방법은 추 후 5번에서 설명하겠습니다.

2. 기존 command를 복사하여 동일 기능의 새로운 command를 정의하고 permission을 변형하여 접근을 제한함 (예) ls를 myls로 복사하되 Set-UID permission을 부여

만족할 수 없다.

Set-UID는 실행시 소유자권한으로 EUID가 상승하는 옵션인데 특정 두 사용자만 공유할려면 자신의 파일을 다른 **한명만 실행** 가능해야하는데 1번과 같이 **그룹원 혹은 나머지 사용자** 옵션으로 접근을 제한하는데 **단 한명에게만 제한을 하는 옵션은 없으므로** 그 외 사용자들도 접근 가능성이 있어서 만족할 수 없다.

3. Set-UID 프로그램을 별도로 작성하고 해당 프로그램을 활용하여 접근을 제한함

만족할 수 있다.

Real ID 비교를 통해 공유할 사용자의 ID를 조건을 넣어 접근시 그 외 사용자가 접근할 경우 실행을 못 하도록 막을 수 있다.

파일 생성자	seed (uid = 1000)
공유자	seed와 tw (uid = 1001)
침입자	evil (uid = 1002)
공유할 파일	shdoc (소유자인 seed만 읽기 가능 chmod 4700)
Set-Uid 프로그램	shcat

공유파일 접근법
shcat shdoc 으로 접근하면 파일에 append를 할 수 있고 append를 끝내고
저장 및 종료 할려면 exit
exit입력후 파일 전체 출력

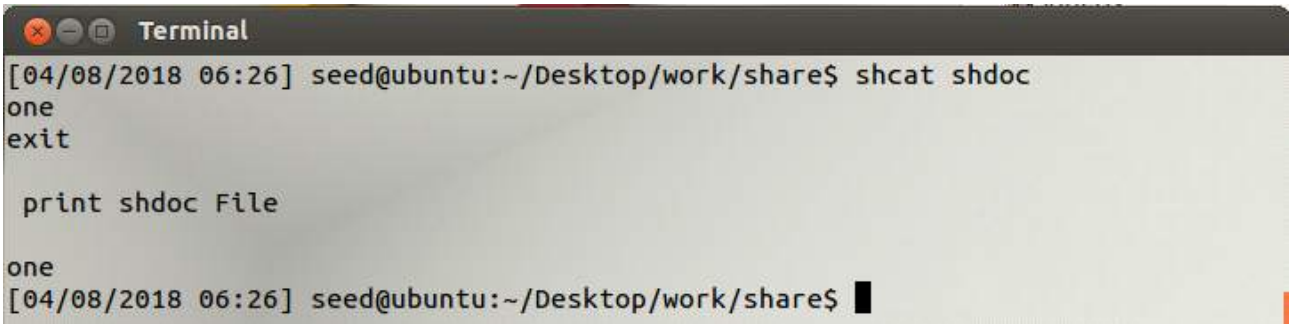
만약 tw가 생성한 파일을 seed와 공유하고 싶다면 tw가 c코드를 컴파일해서 shcat2로 접근 프로그램을 만들어서 seed만 접근가능하도록 하면된다.

share.c 코드

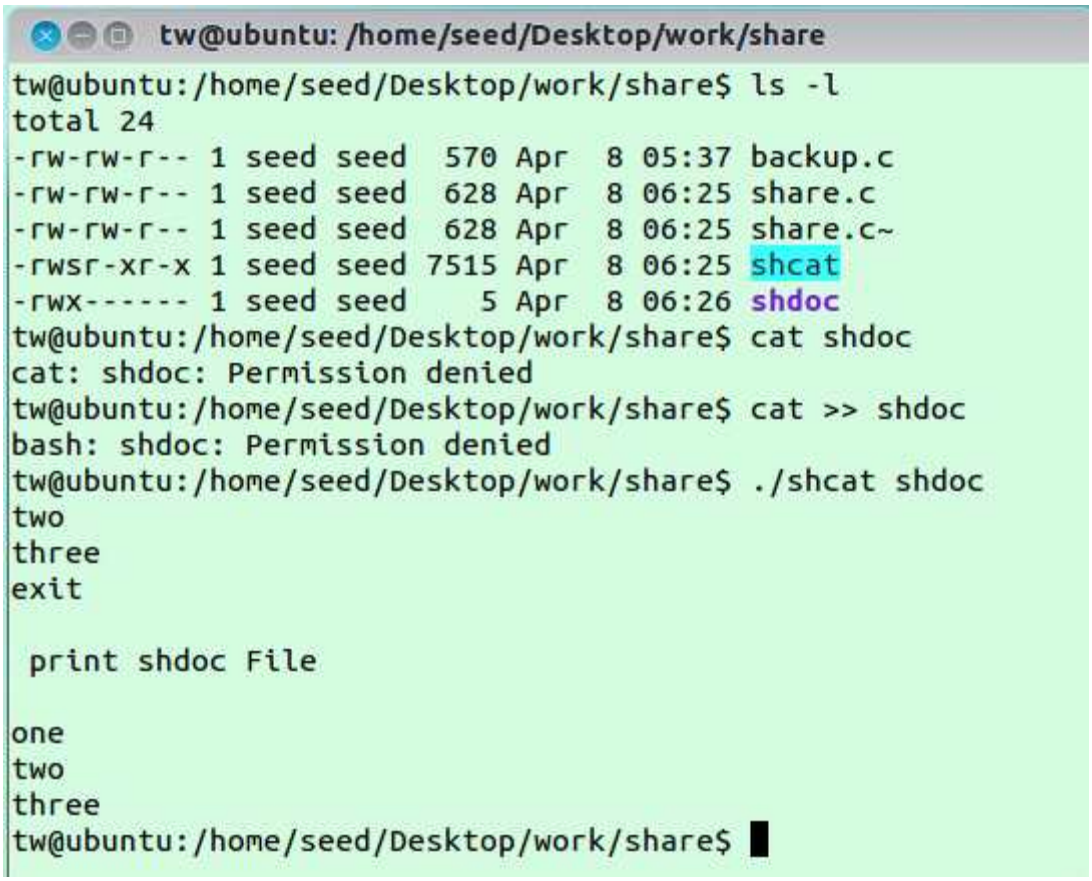
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc,char *argv[])
{
        FILE* fp;
        char buffer[60];
        uid_t uid = getuid(); //Real id를 얻기위한 변수
        if(argc != 2)
        {
                printf("argument error₩n");
                exit(1);
        }//실행 argument 잘 못 줬을 경우 에러 메시지 출력
        else if(uid==1000||uid==1001)//1000은 seed 1001은 tw의 Real ID
        {
                fp = fopen(argv[1],"a+");//파일을 append용으로 연다
                scanf("%s",buffer);
                while(strcmp(buffer,"exit"))//받은 입력이 exit일 때까지 입력 받음
                {
                        fwrite(buffer, sizeof(char), strlen(buffer), fp);
                        fwrite("₩n", sizeof(char), 1, fp);
                        scanf("%s",buffer);
                }
                fclose(fp);//입력이 다 끝났으니 파일은 닫아서 파일 쓰기를 완료
                printf("₩n print %s File₩n",argv[1]);
                execl("/bin/cat","cat",argv[1],0);//끝났으니 파일 전체를 출력
        }
        else printf("you are not our member₩n");
        //Real ID가 seed나 tw가 아닐 경우 에러출력
        return 0;
}
```

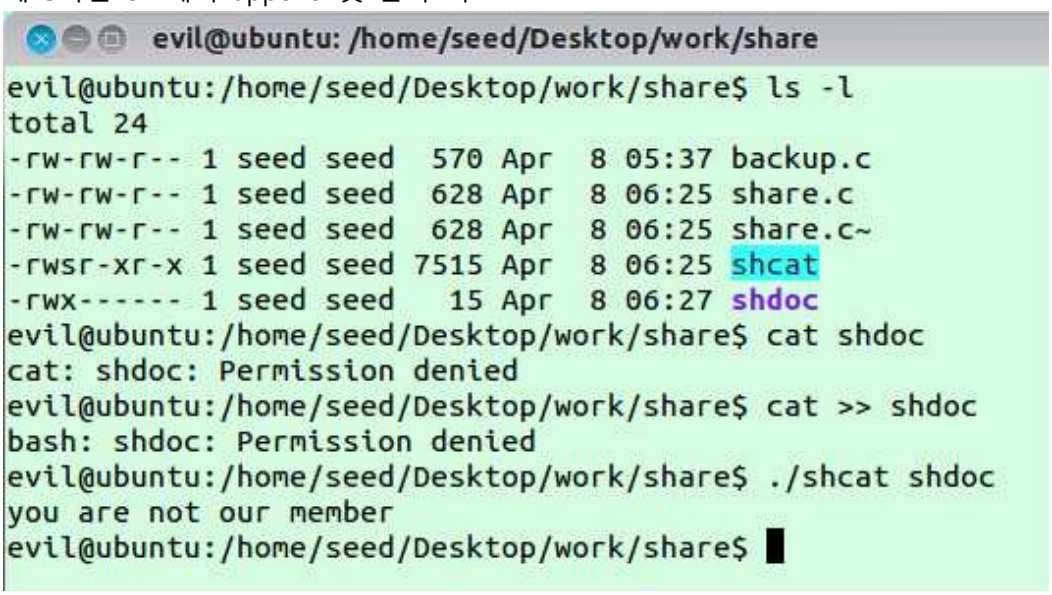
파일 소유자인 seed에서 append 및 출력시도



tw 사용자에서 append 및 출력 시도



제 3자인 evil에서 append 및 출력 시도



4. Daemon 프로그램을 활용함 (Daemon 프로그램의 동작 메커니즘을 제안하는 것으로 충분함. 구현할 필요 없음)

만족할 수 있다.

데몬 프로세스는 백그라운드로 동작하면서 특정한 서비스를 제공하는 프로그램으로서
Real ID를 이용하여서 공유하고 싶은 특정 사용자가
공유하는 파일 접근시 Real ID를 확인하여서 euid를 바꿔주는 식으로 구현 가능할 것 같다.
혹은 보안에 취약 할지라도 광범위적으로 특정 Real ID가 들어온다면 그 ID의 euid를 바꿔주는 식으로도 구현 가능 할 것이다.
권한을 반납하는 코드를 데몬프로그램에 추가 하거나 따로 프로그램화 시켜놔야 안전할 것 같다.

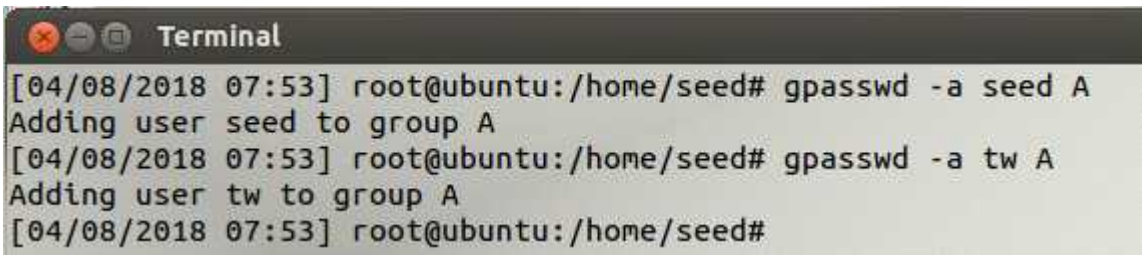
5. 앞에서 제시된 수준 "이상"의 보안성을 제공하는 방법

관리자가 공유수준을 제한 할 수 있다면 가능한 방법이다.

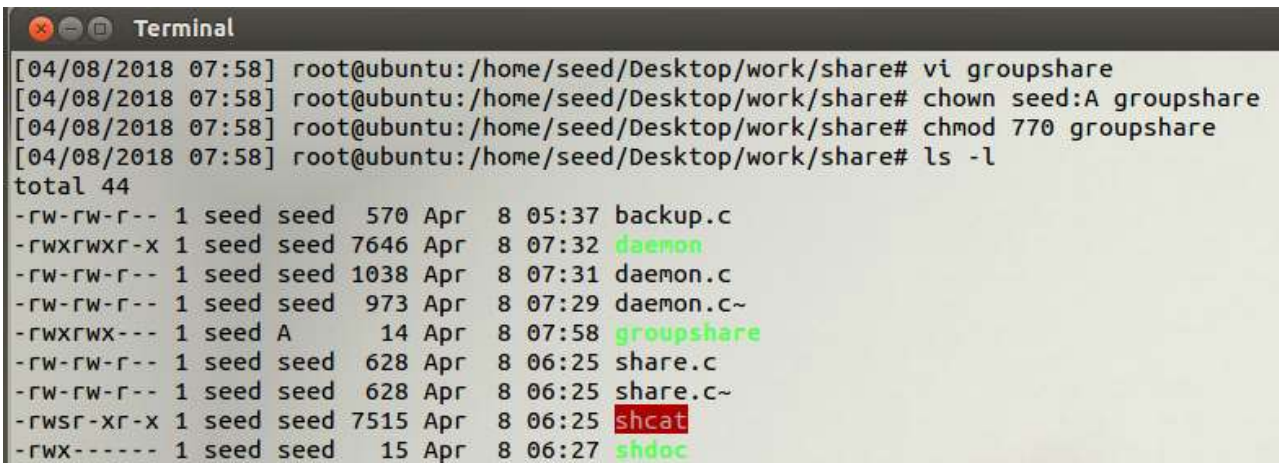
그룹정책으로 보안성을 제공하는 방법이다.

Group을 이용한 방법으로 관리자 권한으로 groupadd A로 A라는 새로운 그룹을 만든 뒤
gpasswd seed A
gpasswd tw A
로 서로 파일을 공유할 계정 두명을 같은 그룹에 추가 시킨 후
seed와 tw는 파일을 만들고 접근 권한은 설정할 때 chmod 770으로 설정하면 된다.

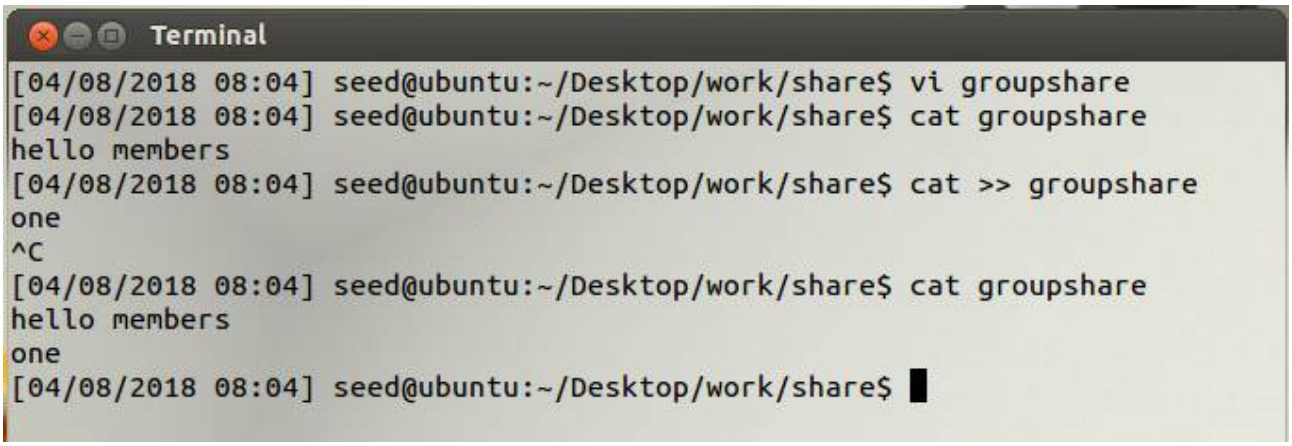
관리자 권한에서 seed 와 tw를 A그룹에 추가



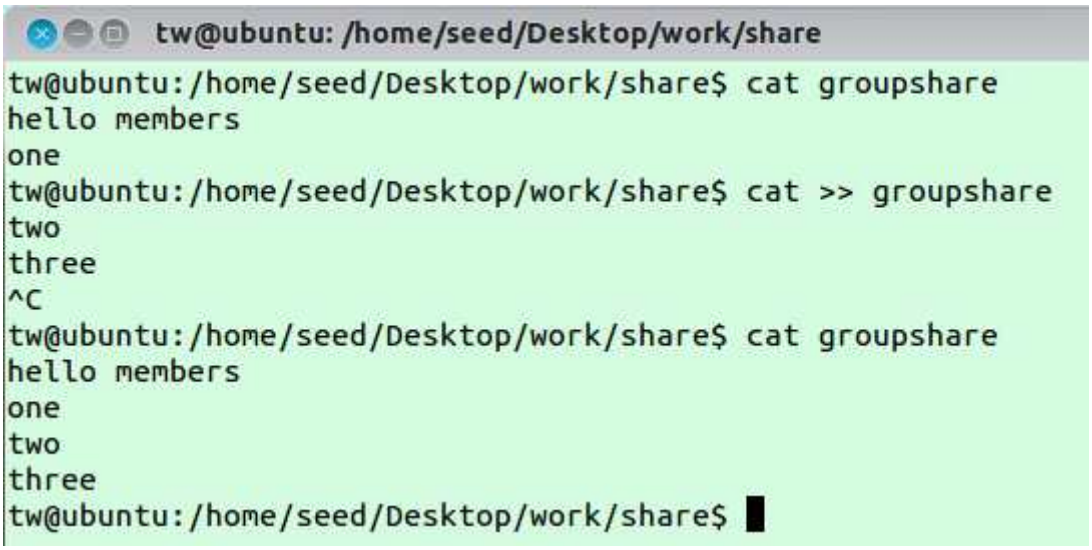
관리자 권한으로 groupshare라는 파일 생성후 파일의 생성자와 그룹을 seed이고 A그룹으로 지정
접근권한을 소유자와 그룹원만 읽기 쓰기 실행 가능하도록 chmod 770으로 부여



seed 계정으로 cat 명령어로 groupshare 파일 출력과 수정



tw 계정으로 cat 명령어로 groupshare 파일 출력과 수정



침입자인 evil계정으로 cat 명령어로 groupshare 파일 출력과 수정

■종합적 비교

조건을 만족한 3,4,5만 비교 하겠습니다.

■조건 3

- 장점은 프로그램이 직관적이고 구현도 쉬운편이고 조건도 잘 만족한다.
- 단점은 seed가 만든 파일에 접근 할때와 tw가 만든 파일에 접근할 때 다른 프로그램을 써야 해서 사용자 입장에서 불편함이 있고 본질적으로 Set-UID를 사용하기 때문에 보안의 취약점이 생길 수 도 있다.

■조건4

- 조건 4 는 구현을 하지 못해서 정확히 판단하긴 어렵지만 백그라운드로 돌고 있기 때문에 파일 접근시 데몬프로세스가 자동으로 특정 사용자의 Real ID를 보고 접근권한(Euid)을 줄지 선택하기 때문에 사용자 입장에선 조건 3보단 편하다.
- 단점은 백그라운드인 만큼 컴퓨터자원이 낭비되고 조건3가 마찬가지로 euid를 부여하는 행위이기 때문에 보안의 취약점이 발생할 수 있다.

■조건5

- 장점은 Setuid가아닌 그룹으로 접근관리를 해서 관리자의 실수가 아닌 이상 조건 3,4보다 접근을 강력히 막을 수 있다.
- 단점은 최초 그룹 설정과 새로운 파일을 생성할때 관리자 권한이 필요하다. 추가와 읽기 말고도 삭제도 가능하다.
- 삭제를 막을려면 Sticky bit를 사용하여서 막을 수 있었다. share 폴더를 root에게 소유를 주고 chmod 1777로 Stickybit를 주면 해당 폴더안의 파일은 소유자만 삭제할 수 있게 되었다.

■ 관리자가 있고 관리자 권한아래 파일을 공유 생성하고 철저한 보안이 필요하다면 조건5로 구현하고 사용자간의 합의하에 공유하는 파일이라면 구현하기 어렵고 직관적이지 않은 조건4보단 조건3이 더 좋다.