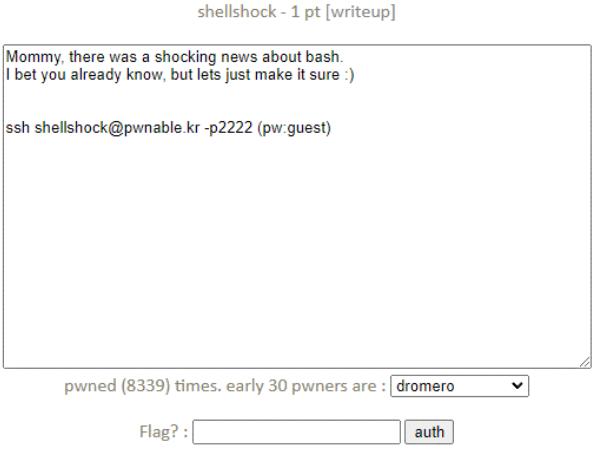
Pwnable.kr
- shellshock -

ssh shellshock@pwnable.kr -p2222
pw : guest

0. 문제 살펴보기



bash에서의 shellshock에 대해 얘기하고 있다.

1. SSH 접속 및 살펴보기

SSH를 이용해 상단에 표기해 놓은 주소와 포트 번호로 접속하였다.
디렉토리의 파일들을 살펴보자 C 코드 파일이 존재하여 확인해보니 위와 같은 코드를 알 수 있었다.
유저 id와 그룹 id를 설정하고 현재 디렉토리에 있는 bash 쉘을 이용하여 echo를 사용하는 것을 볼 수 있다.

2. 취약점 파악

문제에서 shellshock에 대한 얘기를 하고 있는 것을 보아 현재 사용하고 있는 쉘이 보안성이 좋지 않은 bash 쉘인 것 같다.
따라서, 흔히 사용하는 shellshock 방법인 환경변수를 이용한 shellshock를 사용하였다.

```
shellshock@pwnable:~$ export x='() { echo 1; }; bash'
```

위와 같이 환경변수를 함수 형태로 넣음과 동시에 세미콜론(;)을 이용하여 문장을 나누고 현재 디렉토리에 있는 bash를 실행하게 하였다.
이 원리는 취약한 bash 쉘에서만 가능한 방법이다. 정상적인 동작이라면 위와 같은 방법으로 환경변수를 등록하면 x라는 환경변수에 “() { echo 1; }; bash”라는 문자열이 저장되어야 한다.
하지만, 옛날의 취약한 bash 쉘은 위와 같이 환경변수를 등록하면 x라는 “echo 1”이라는 명령을 수행하는 함수가 되며, 하나의 문단이 끝나고 현재 쉘에서 세미콜론(;)으로 나눠진 다음 명령을 수행한다.

환경변수는 자식 프로세스가 실행될 때, 상속되게 되는데 이때 우리가 환경변수로 등록한 x도 같이 상속되어 bash가 실행되는 것이다.
취약한 쉘인지 확인하는 방법은 환경변수를 살펴보면 된다.

```
shellshock@pwnable:~$ env | grep x
x=() { echo 1; }; bash_
```

위와 같이 문장으로 삽입되지 않았다면 취약한 쉘인 것이다.

3. 공격

현재 우리가 확인한 코드는 자신의 Set-UID를 소유자의 것으로 설정하고 있다. 따라서, Set-UID가 설정된 상태로 bash를 실행하여 “echo shock_me”를 실행하는데, 이 bash가 실행될 때, 우리의 환경변수가 같이 상속되어 bash라는 쉘을 다시 한번 실행할 것이다.
이때는 자신의 Set-UID가 소유자의 것으로 바뀐 상태에서 실행되기 때문에 새로 실행된 bash는 소유자의 권한을 가지고 있을 것이다.

```
shellshock@pwnable:~$ ./shellshock
shellshock@pwnable:~$ id
uid=1019(shellshock) gid=1020(shellshock_pwn) groups=1020(shellshock_pwn),1019(shellshock)
```

shellshock 프로그램을 실행시키고 id를 확인해보자 gid가 정상적으로 바뀐 것을 볼 수 있다.

```
shellshock@pwnable:~$ cat flag
{[redacted]
```

이때 cat을 이용하여 flag 파일을 확인할 수 있으며 이곳에 우리가 원하는 flag 문장이 있다.