Linux Kernel Exploitation

B01902054 許軒

Overview

- You are an unprivileged user
- The kernel or kernel module have vulnerabilities
- You write an user-space program that sends a payload to the kernel to trigger the vulnerability
- Hijack the kernel’s control flow and redirect it to the shellcode placed in your program
- Raise your credential
- Return to user mode
- Open a shell and you got a root shell

Process Memory Layout

- x86_64 uses 48 bits addresses
- Bit 47 - 63 must be identical
- 0000 0000 0000 0000 - 0000 7FFF FFFF FFFF: user space
- 0000 8000 0000 0000 - FFFF 7FFF FFFF FFFF: noncanonical
- FFFF 8000 0000 0000 - FFFF FFFF FFFF FFFF: kernel space

Hijack the Control Flow

- The kernel shares the same address space with your process
- Just let the kernel jump to the address in your process

task_struct in include/linux/sched.h

```
struct task_struct {
        volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
        void *stack;
        …
/* process credentials */
        const struct cred __rcu *real_cred;
        const struct cred __rcu *cred;
        …
}
```

cred in include/linux/cred.h

```
struct cred {
        …
        kuid_t uid;
        kuid_t gid;
        kuid_t suid;
        kuid_t sgid;
        kuid_t euid;
        kuid_t egid;
        …
}
```

Credential Operations in kernel/cred.c

- struct cred *prepare_kernel_cred(struct task_struct *daemon)

  prepare a set of credentials for a kernel service

- int commit_creds(struct cred *new)

  install new credentials upon the current task

Using Credential Operations in the Shellcode

- /proc/kallsyms exports the addresses of kernel functions
- Not available for unprivileged users
- Distributions use compiled kernels
- Set up a same environment on other machines to obtain the addresses of prepare_kernel_cred() and commit_creds()

Syscall / Sysret

- syscall saves (rip, rflags) to (rcx, r11) and switch from ring 3 to ring 0
- The kernel performs swapgs and sets up its own rsp
- sysret switch from ring 0 to ring 3 and restores (rip, rflags) from (rcx, r11)

Experiment

- The vulnerable kernel module: vul.c

  receives payloads from /dev/vul

  writes payloads to its stack frame

- The user-space exploitation: exp.h, exp.c, exp.asm

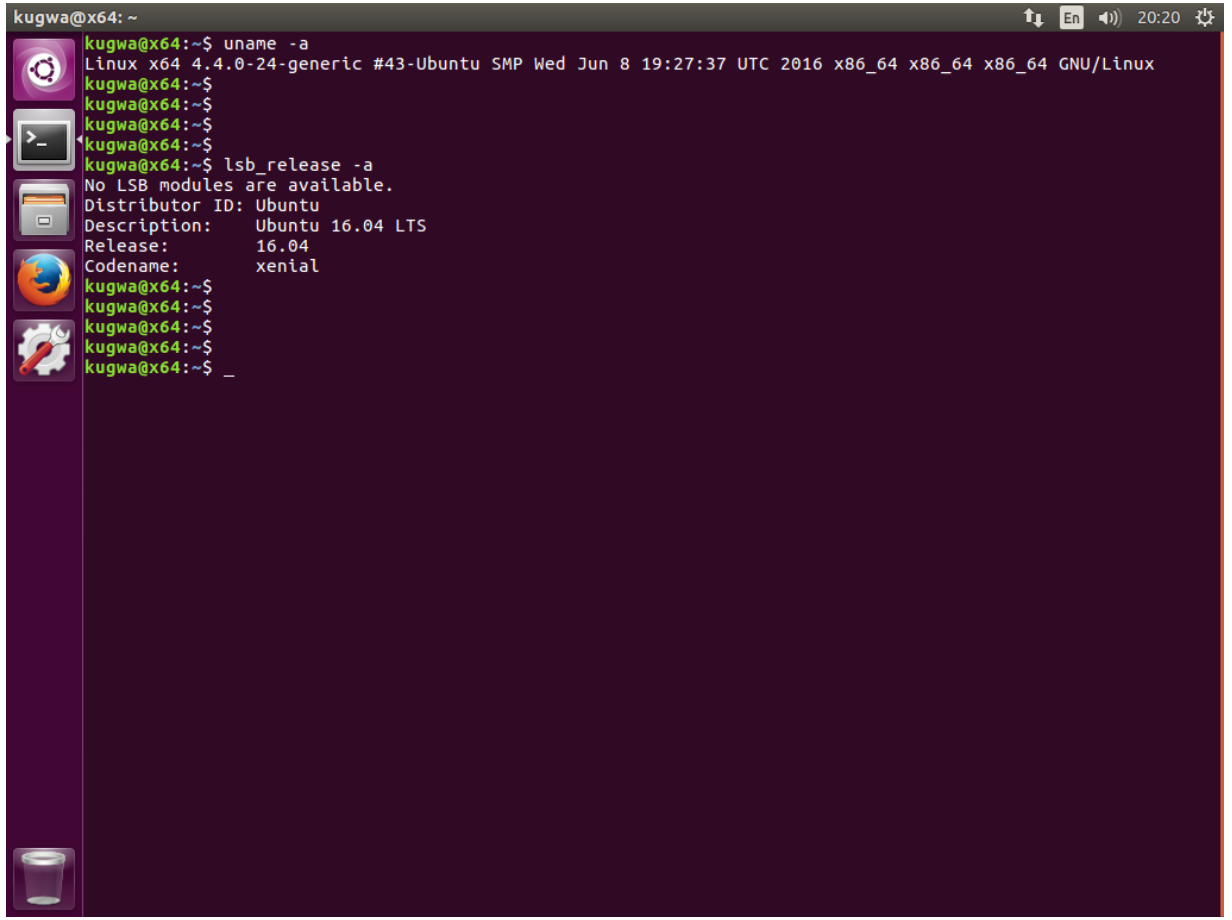
  saves (rip, rflags)

  sends a payload to /dev/vul to gain full CPU privilege
  commit_creds(prepare_kernel_cred(NULL))

  rsp = a prepared buffer; (rcx, f11) = saved (rip, rflags); swapgs; sysret;

  system(“sh”)

Test Environment



Execution Result