SYNACKTIV

Rétro-ingénierie

Partie 2: Recherche de vulnérabilité

Paul VIEL

Sommaire

SYNACKTIV

- Methodologie d'analyse
- Type de vulnérabilités
- Techniques d'exploitation
- Mitigations

Methodologie d'analyse

SYNACKTIV

- Elaborer un scénario d'attaque en se mettant dans la peau d'un attaquant
  - Définir le but de l'attaque (ex: destruction du système, vol de données, détournement du système, etc ...)
  - Définir l'attaquant
    - Ses privilèges (administrateur, utilisateur, invité, ...)
    - Sa position (physique, local, distant, ...)

Methodologie d'analyse

SYNACKTIV

- **Etudier la faisabilité du scénario envisagé**
  - Observation macroscopique vers microscopique
    - Avoir une bonne vue d'ensemble
    - Identifier les éléments sur lesquels l'attaquant peut agir pour arriver à ses fins
    - Identifier et suivre les données maitrisables par l'attaquant
    - Trouver les mécanismes sensibles, allez a l'essentiel
    - Documenter uniquement ce qui a besoin de l'être
  - Faire des hypothèses sur le fonctionnement du produit
    - En fonction des entrées/sorties, des bibliothèques logicielles utilisées, …
      Prendre du recul, retrouver la logique du programmeur
  - Chercher à confirmer ou infirmer les hypothèses en utilisant l'outil le plus adapté
  - Ne pas perdre de vue l'objectif principal
  - Conclure: Décrire le problème et proposer des contre-mesures

Terminologie

SYNACKTIV

- **Vulnérabilité**: Bug dans un binaire qui peut donner lieu à un exploit
- **Exploit**: Données spécialement conçues qui utilisent une ou plusieurs vulnérabilités pour forcer le binaire à faire quelque chose de non attendu.
- **0day**: Vulnérabilité inconnue, non patchée qui peut être utilisée par un exploit
- **POC (Proof Of Concept)**: preuve de concept, produit seulement un crash, loin d'un exploit
- **Pwn**: fait référence à la recherche de vulnérabilité et au développement d'exploits

Types de vulnérabilités

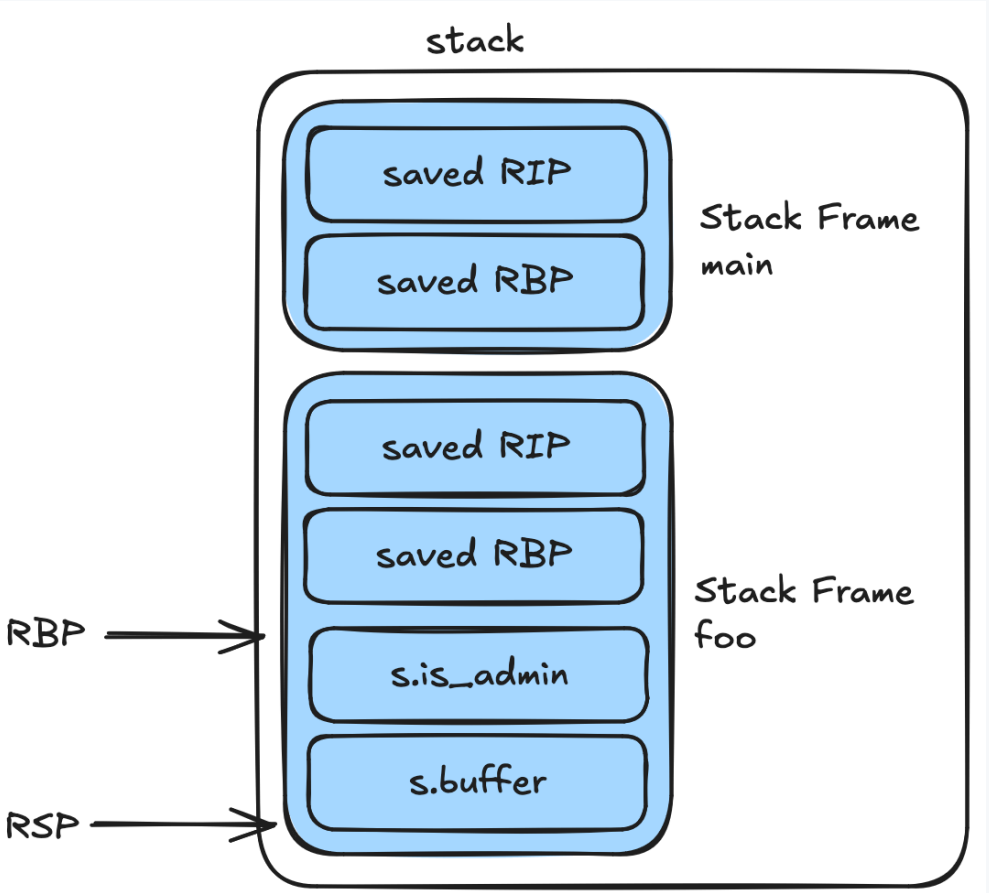
Buffer Overflow

SYNACKTIV

```
struct s{
    char buffer[16],
    int is_admin,
};

void foo(){
    struct s data;
    scanf("%s", data.buffer);
    if (data.is_admin){
        // admin thing
    }else{
        // not admin thing
    }
}

void main(){
    foo();
}
```



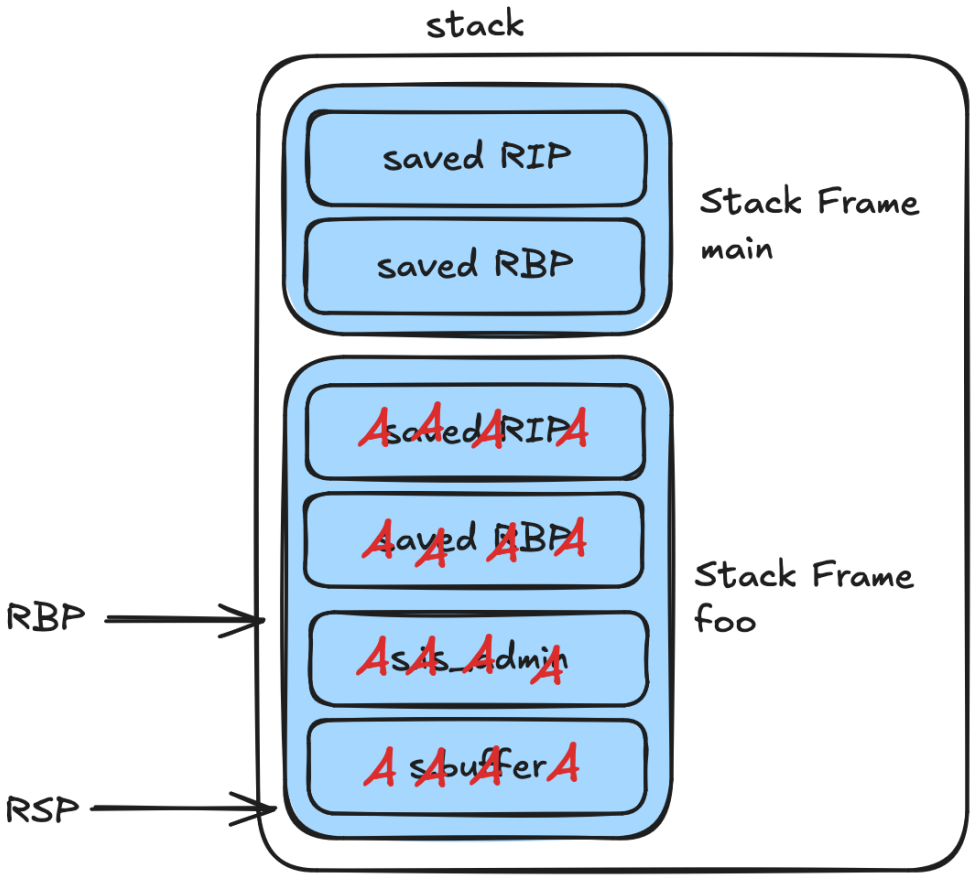
Stack Buffer Overflow

SYNACKTIV

```
struct s{
    char buffer[16],
    int is_admin,
};

void foo(){
    struct s data;
    scanf("%s", data.buffer);
    if (data.is_admin){
        // admin thing
    }else{
        // not admin thing
    }
}

void main(){
    foo();
}
```

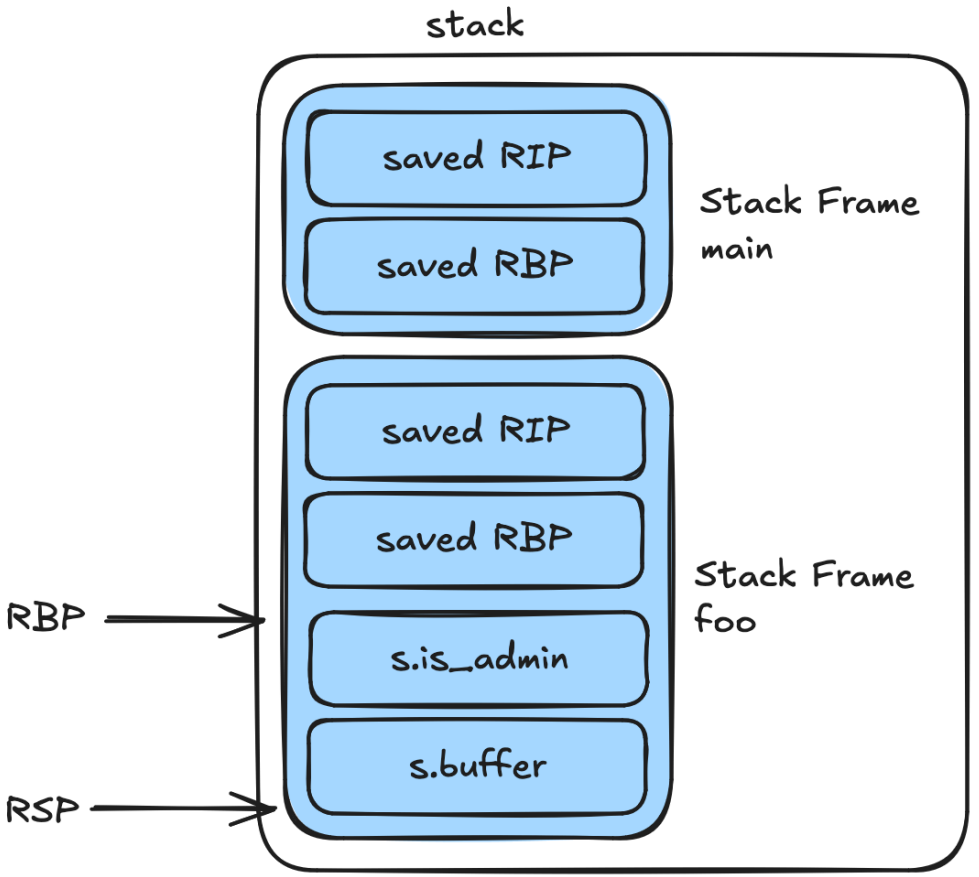


Rappel: Utilisation de la stack

SYNACKTIV

```
0X401106 foo:
  push   rbp
  mov    rbp,rsp
    /* ... */
  leave
  ret

0x40111b main:
  push   rbp
  mov    rbp,rsp
  call   401106 <foo>
  leave
  ret
```



Integer overflow

SYNACKTIV

- En informatique les entiers sont encodés sur un nombre fini de bits
- Un dépassement de valeur peut donner lieu a un comportement inattendu (ex avec un entier non signé sur 8 bits : 255 + 4 = 3)

```
void main(){
    int nb_note = ask_user_how_many_notes();
    int *notes = malloc(nb_note*sizeof(int));

    for(i=0; i<nb_note; i++)
        ask_user_a_note(notes[i]);
}
```

Type confusion

SYNACKTIV

```
void main(){
    int idx;
    char notes[10];
    scanf("%d", &idx);
    if(idx < 10){
        scanf("%c", &notes[idx]);
    }
}
```

Format String

SYNACKTIV

- `%d` : affiche un entier signé
- `%x` : affiche un entier sous forme hexa
- `%s` : affiche une chaine de caractère
- `%n` : enregistre le nombre de caractères affichés

```
void main(int argc, char ** argv){
    printf(argv[1]);
}
```

Uninitialized variable

SYNACKTIV

```
void foo1(){
    int a;
    scanf("%d", &a);
}
void foo2(){
    int b;
    printf("%d", b);
}
void main(){
    foo1();
    foo2();
}
```

Race conditions

SYNACKTIV

TOCTOU: Time Of Check Time Of Use

```
void main(){
    struct stat st;
    stat("/tmp/script.sh", &st);

    if (st.uid == getuid()){
        execve("/tmp/script.sh", NULL, NULL);
    }
}
```

Cryptographie

SYNACKTIV

Réduction d'entropie

```
void secure_random_number(){
    return rand()%10;
}
```

Caractérisation

SYNACKTIV

- Certains type de vulnérabilités sont plus ou moins facile à exploiter
- L'attaquant doit faire frd hypothèses sur la cible
  - Type de processeur
  - OS
  - Version du produit
  - Etat de processus
  - Filtre (taille, jeu de caractères, ...)
  - Produit de sécurité (Antivirus, EDR, ...)
  - Equipement de sécurité sur la route (IDS, IPS)
- Un exploit n'est pas forcément 100% fiable
- Cependant un attaquant peut tenter N fois son attaque pour augmenter ses chances (sur la même cible ou non)

Techniques d'exploitation

En général

SYNACKTIV

Objectifs:

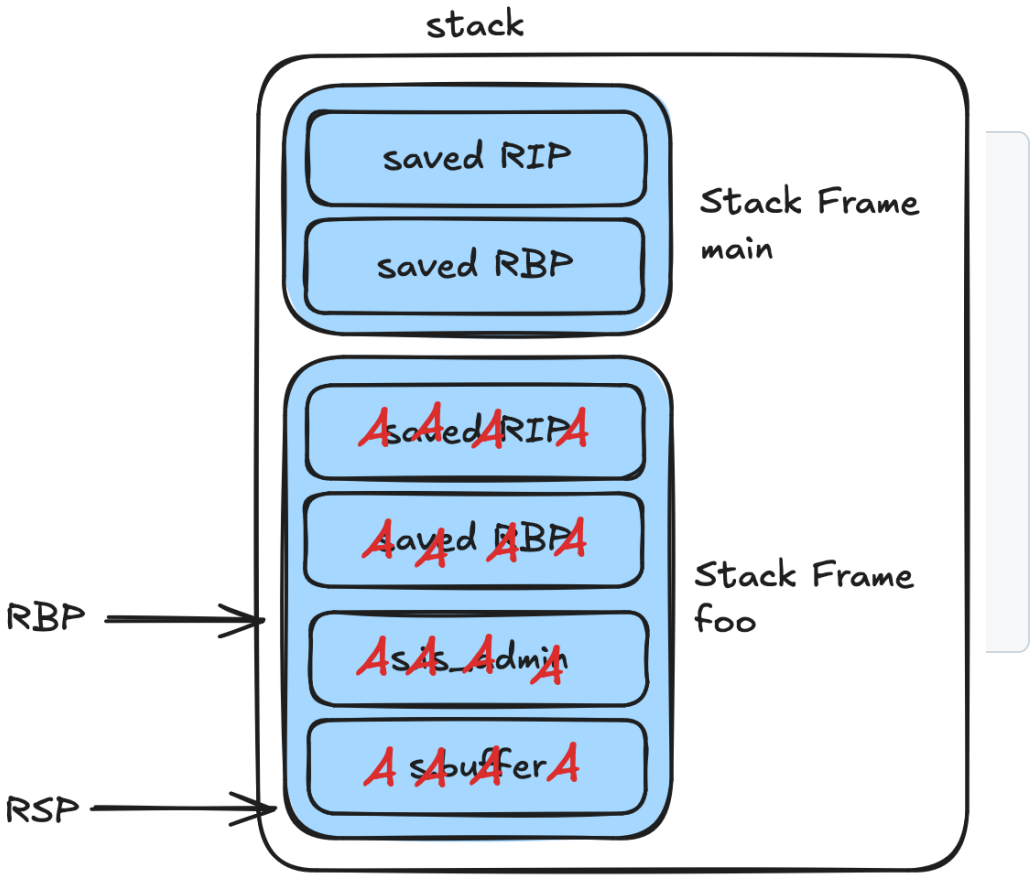
- Fuite d'information
- Contrôller le flux d'execution
- Le graal: Avoir un shell sur la machine

Stack BOF

SYNACKTIV

```
void foo(){
    char buf[2];
    scanf("%s", buf);
}

void main(){
    foo();
}
```

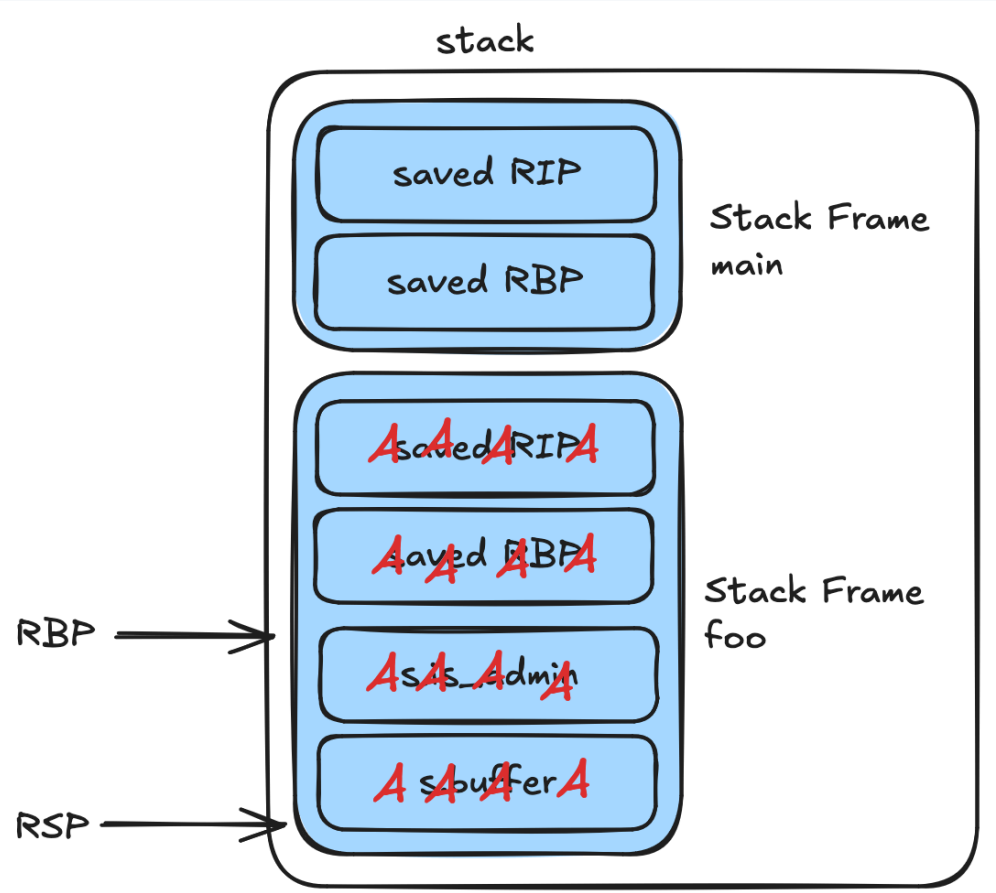


Stack BOF

SYNACKTIV

```
foo:
    push RBP
    sub RSP, XX
    /* ... */
    add RSP, XX
    leave
    ret

main:
    push RBP
    sub RSP, XX
    /* ... */
    call foo
    /* ... */
    add RSP, XX
    leave
    ret
```



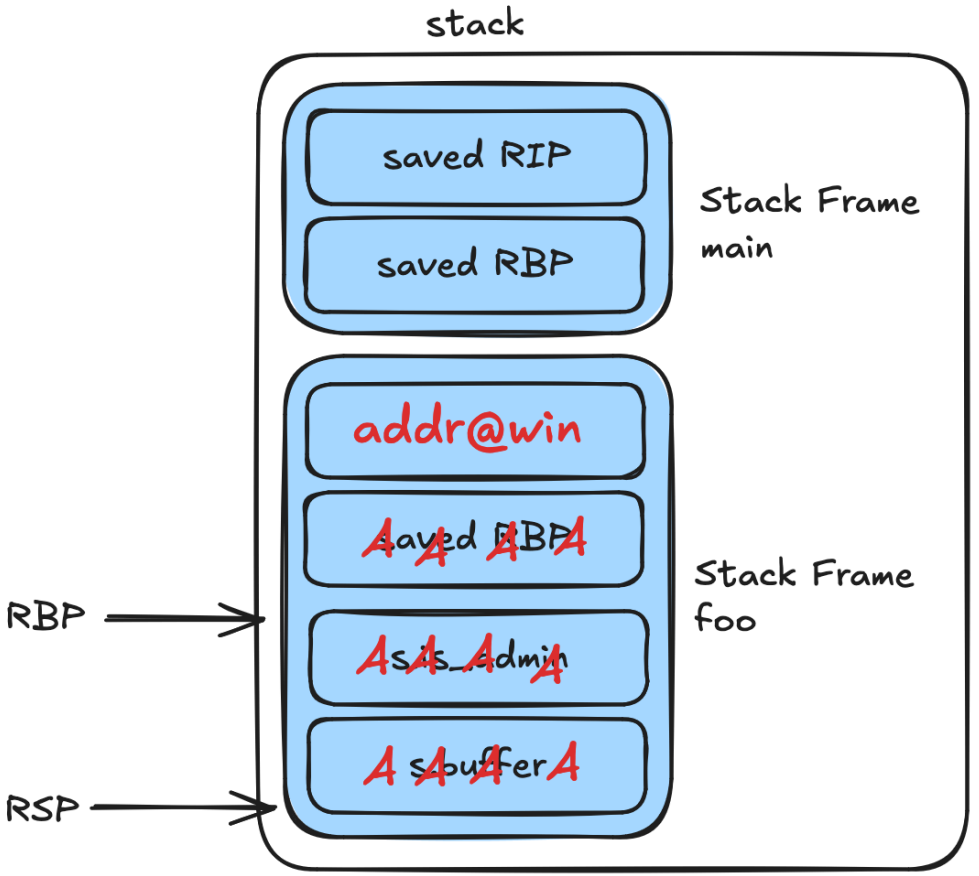
ret2win

SYNACKTIV

```
void win(){
    system("/bin/sh");
}

void foo(){
    char buf[2];
    scanf("%s", buf);
}

void main(){
    foo();
}
```

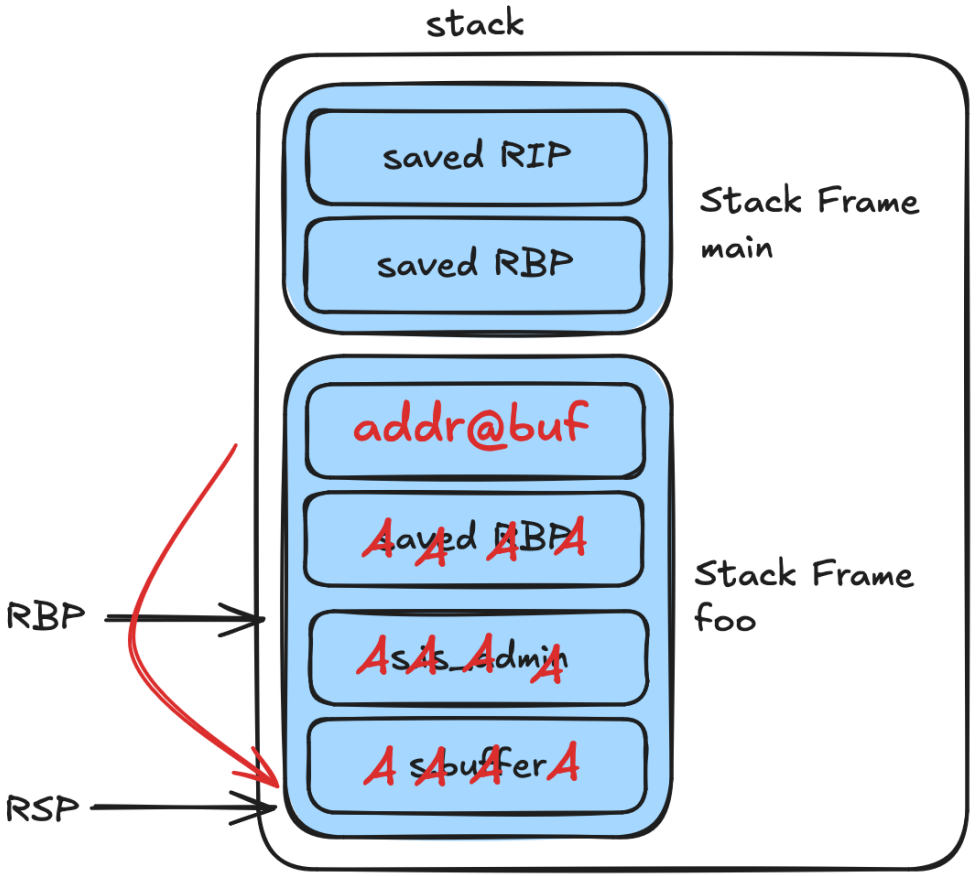


ret2stack

SYNACKTIV

```
void foo(){
    char buf[2];
    scanf("%s", buf);
}

void main(){
    foo();
}
```

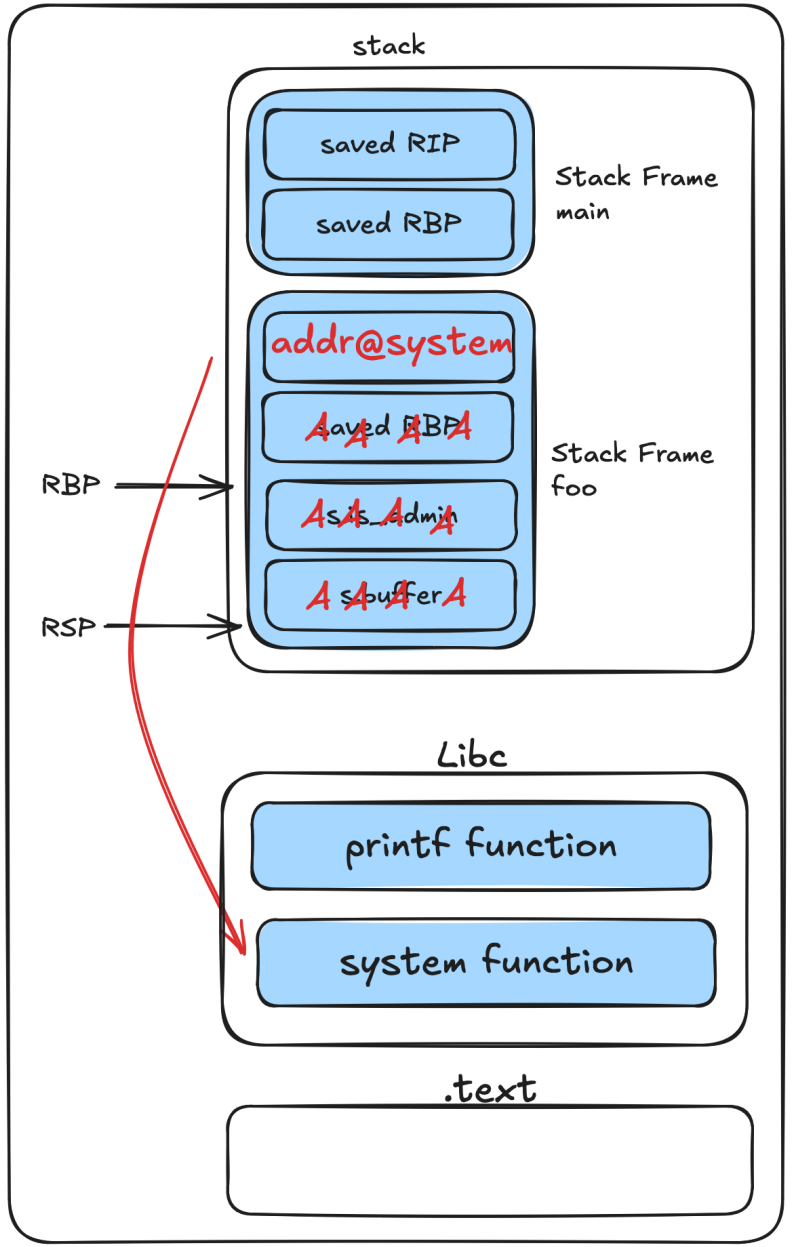


ret2libc

SYNACKTIV

```
void foo(){
    char buf[2];
    scanf("%s", buf);
}

void main(){
    foo();
}
```



ropchain

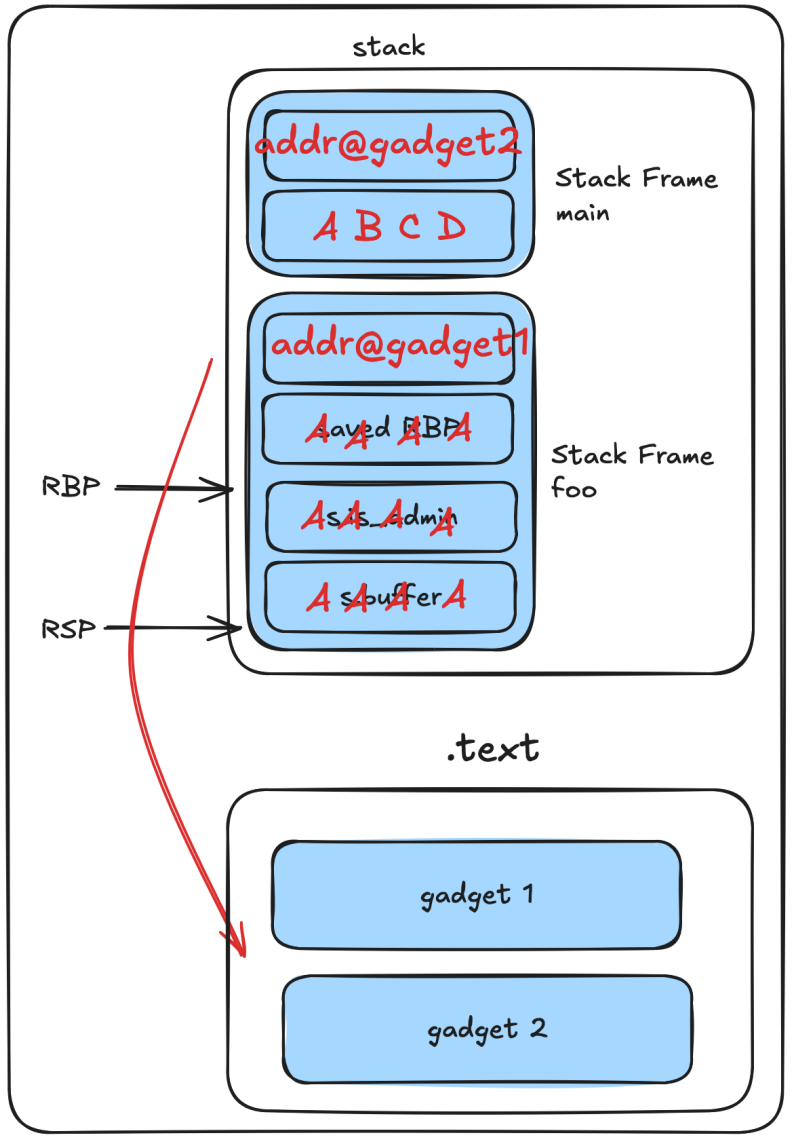
SYNACKTIV

```
gadget1:
    ret

gadget2:
    pop RAX
    ret

gadget3:
    pop RBP
    ret

gadget4:
    mov [RBP], RAX
    ret
```



Autre techniques d'exploitation

SYNACKTIV

- ret2dlresolve
- ret2plt
- ret2init
- JOP
- FSOP
- dtor freelist
- ...

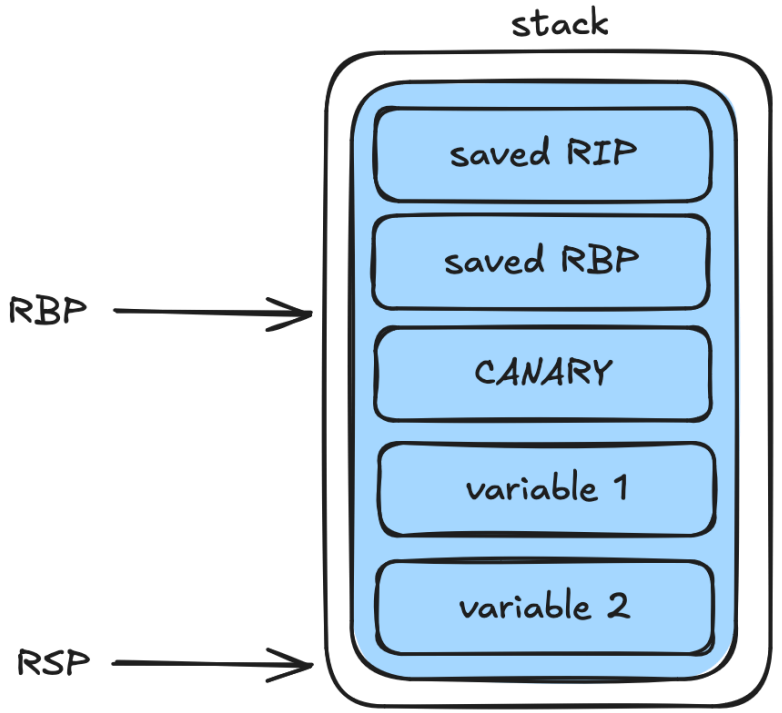
Mitigations

contre-mesures pour rendre certains type de vulnérabilités difficilement exploitables

Canary / stack cookie

SYNACKTIV

- Permet de détecter les stack buffer overflow
- Au début de la fonction le programme place une valeur aléatoire (canary) entre les variables locales et l'adresse de retour
- Avant de quitter la fonction le programme vérifie la valeur du canary, si celle-ci a changé c'est qu'il y'a eu buffer overflow



Not eXecutable (NX)

SYNACKTIV

- Rend les zones de données (comme la stack) non exécutable

Address Space Layout Randomization (ASLR)

SYNACKTIV

- La distribution aléatoire de l'espace d'adressage (ASLR) est une technique permettant de placer de façon aléatoire les zones de données dans la mémoire virtuelle
- Limites : si le programme fait fuiter des adresses l'attaquant peut recalculer les adresses de la section associée
- Sous Windows : les adresses sont distribuées de manière aléatoires à chaque boot.
- Sous Linux : les adresses sont distribuées de manière aléatoire à chaque exécution

autres mitigations

SYNACKTIV

- Control Flow Guard
- Return Flow Guard
- KASLR
- Code Integrity
- Sandoxing
- SMEP
- SMAP
- Randomizing structure layout
- Arbitrary code guard
- Pointer Authentication Code
- Intel CET
- ...

Cas des autres langages

SYNACKTIV

- **CPP**

  Mêmes problèmes que le C mais utilise des structures plus complexes

- **Rust**

  Memory safe: s'assure qu'il n'y a pas de corruption lors de la compilation

  Grace à ses concept clés : Ownership, Lifetime, Immutability

- **Python**

  Langage interprété -> nécessite une sortie de machine virtuelle

Types de logiciels à exploiter

SYNACKTIV

- Userland
- Browser
- Kernel
- VM
- Full chain

Conclusion

SYNACKTIV

- Les exploits liés aux corruptions de mémoires sont de plus en plus difficiles à exploiter
  - Beaucoup de mitigations sur les systèmes actuels
  - Utilisation de langage sécurisé tels que le Rust
- Mais …
  - Remote stack based buffer overflow
  - Pas de canary
  - Pas ASLR
  - Pas de séparation des privilèges

Conclusion

SYNACKTIV

- Lorsqu'il y a une vulnérabilité, ce n'est pas parce qu'il y a des mitigations qu'elle n'est pas exploitable
- On peut prouver que la vulnérabilité est exploitable seulement en l'exploitant.

SYNACKTIV

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

https://twitter.com/synacktiv

https://synacktiv.com