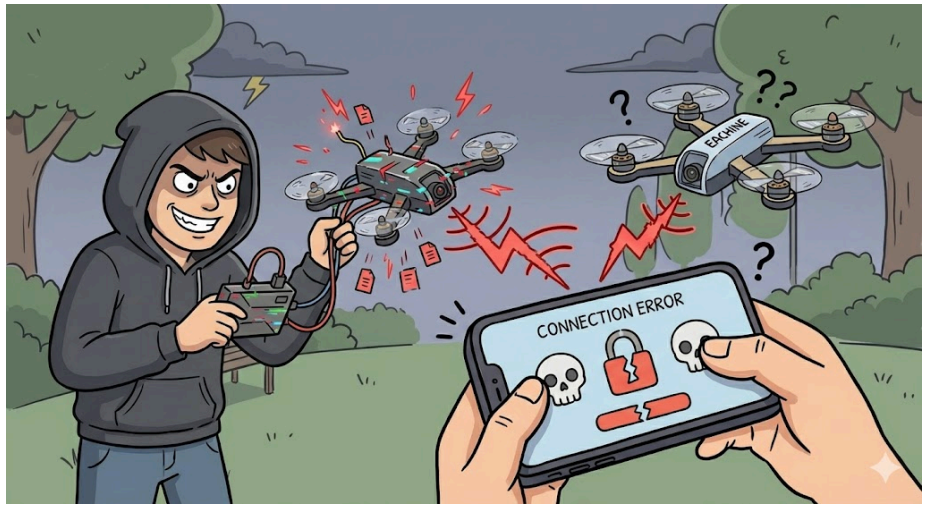
SYNACKTIV

Pwning a pilot phone via its $30 drone control app

https://www.synacktiv.com/publications/from-cheap-iot-toy-to-your-smartphone-getting-rce-by-leveraging-a-companion-app

Introduction

SYNACKTIV



La cible

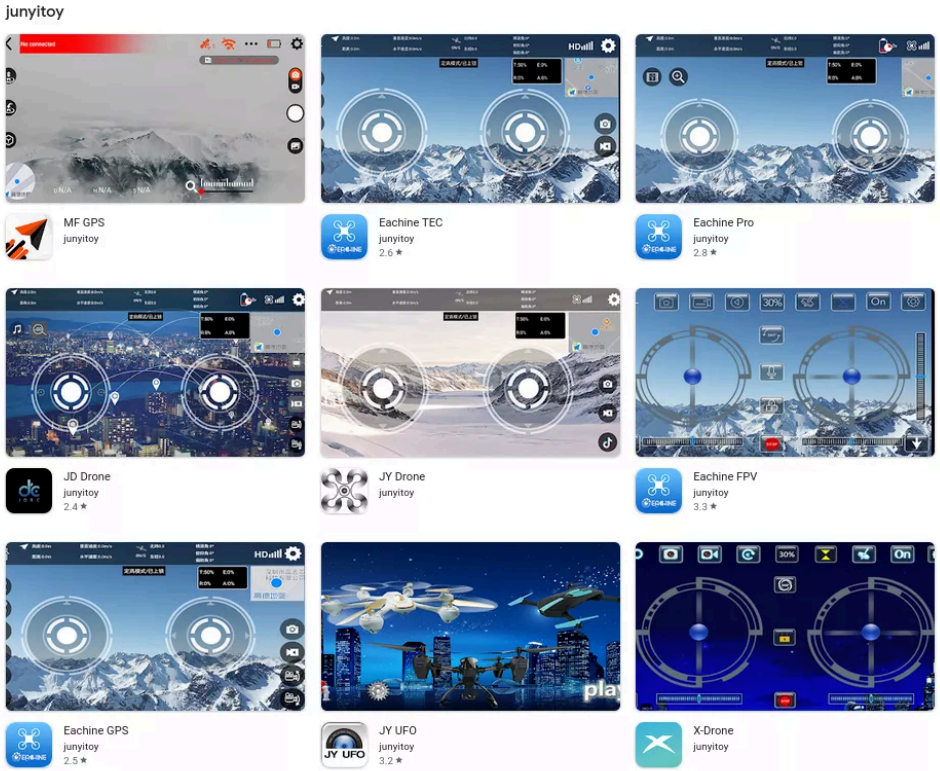
SYNACKTIV

- Eachine E58
- Pas cher, environ 30$
- **Wifi ouvert** 🛜

Application

SYNACKTIV

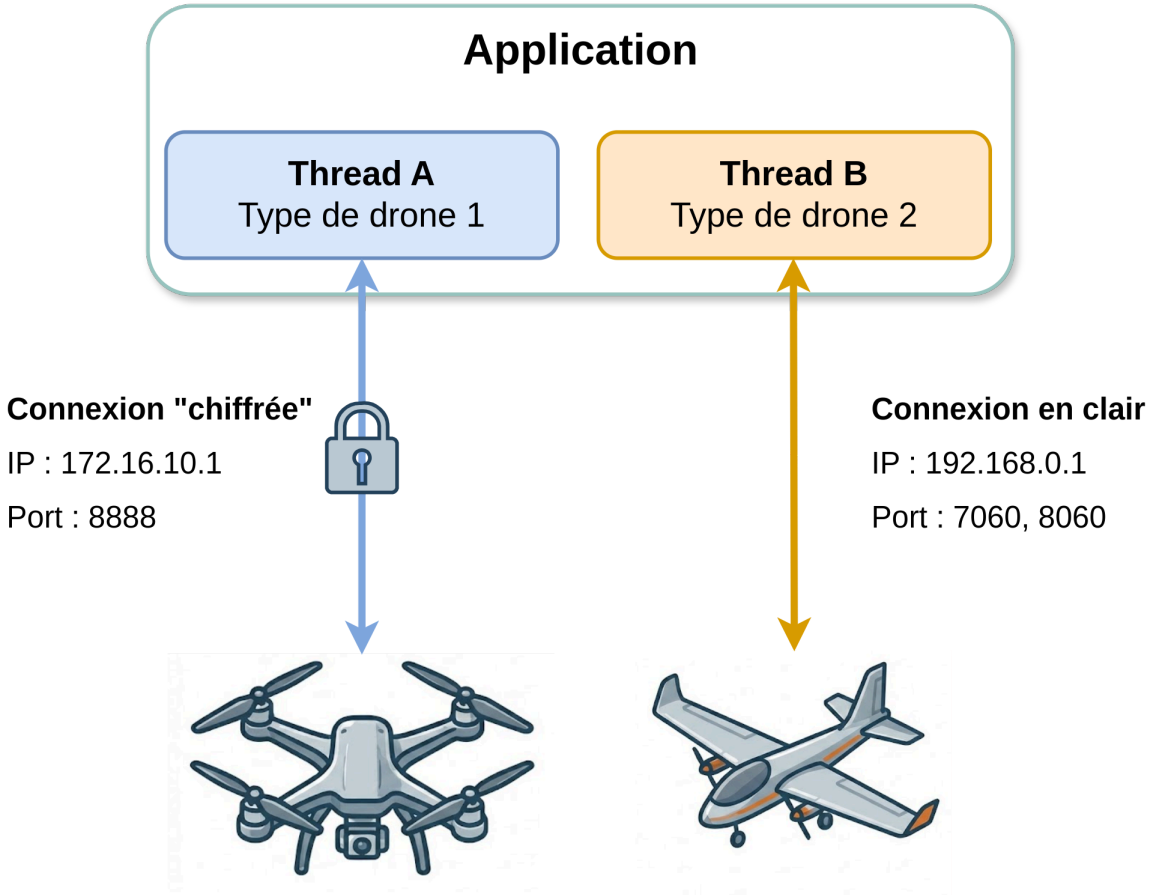
- Beaucoup de choix
- Utilisation de biblothèques natives pour les communications
- Java obfusqué[1]
- Relativement génériques : contrôlent toutes plusieurs types de drones



1: **Sauf sur les versions où ils ont oubliés** 🤡

Fonctionnement global

SYNACKTIV



Vulnérabilité

SYNACKTIV

Drone Type 2: Heap Buffer overflow

```
1  int send_command(int cmd_id, uint64_t a2, char* buffer, int* size) {
2    char* buffer = malloc(0x200uLL);
3
4    // [...]
5
6    if ( net_recv(socket_8060, buffer, 46) <= 0x2D )
7        goto EXIT_ERROR;
8    if ( strcmp(&buffer->magic, "lewei_cmd") )
9        goto EXIT_ERROR;
10
11   // [...]
12
13   net_recv(socket_8060, buffer, (int) buffer->size); // ☠☠☠☠
14
15   // [...]
16 }
```

Vulnérabilité

Drone Type 2: Heap Buffer overflow

```
Signal 7 (SIGBUS), code 1 (BUS_ADRALN), fault addr 0x0041414141414141
     x0  4141414141414141  x1  00000078654ead28  x2  0000007725 9019b0  x3  0000000000000000
     x4  0000007725901430  x5  000000000032258a  x6  0000000000000001  x7  00001118480c2e3d
     x8  0000007885466f10  x9  4141414141414141  x10 0000000000000003  x11 0000000000000000
     x12 0000000000000000  x13 00000078e55836c0  x14 0000000000000033  x15 00000078e55836c0
     x16 0000000000000001  x17 0000007b4e2583b0  x18 0000007721 31e000  x19 0000007a45491a10
     x20 0000007a45491a10  x21 0000007725902000  x22 0000007865459fd0  x23 0000000000000000
     x24 0000007725 9012e0  x25 0000000000000000  x26 0000000000000000  x27 0000007725902000
     x28 00000078e547e680  x29 0000007725901140
     lr  0000007b61516188  sp  0000007725901140  pc  0041414141414141  pst 0000000060001800
 17 total frames
 backtrace:
       #00 pc 0000004141414141  <unknown>
       #01 pc 0000000000516184  /system/lib64/libhwui.so (LinearAllocator::~LinearAllocator(...)
       #02 pc 0000000000501d74  /system/lib64/libhwui.so (skiapipeline::SkiaDisplayList::reuseDisplayList(...)
       #03 pc 0000000000526a2c  /system/lib64/libhwui.so (RenderNode::deleteDisplayList(...)
       #04 pc 0000000000527958  /system/lib64/libhwui.so (RenderNode::prepareTreeImpl(...)
       // [...]
```

LinearAllocator::~LinearAllocator()

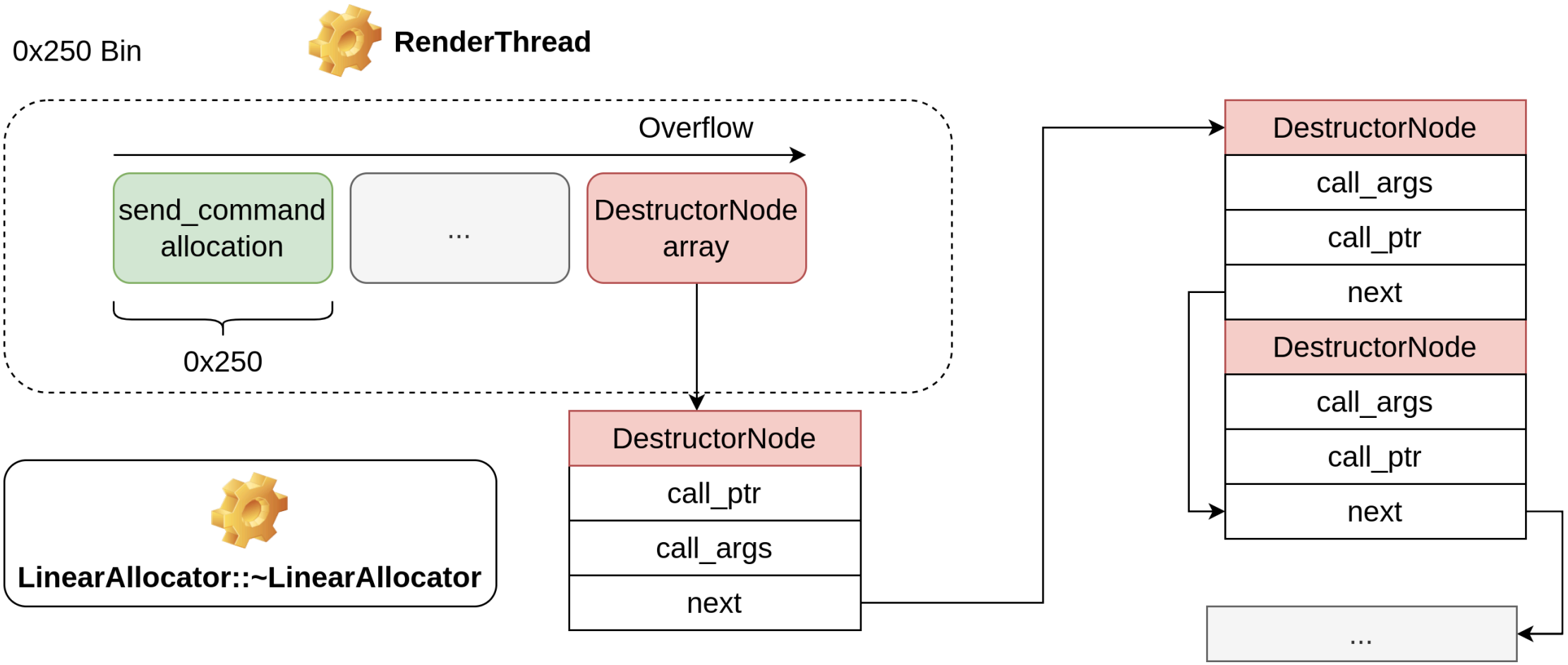
COP-Chain as a service

SYNACKTIV

```
1 LinearAllocator::~LinearAllocator(void) {
2     while (mDtorList) {
3         auto node = mDtorList;
4         mDtorList = node->next;
5         node->dtor(node->addr); // Arbitrary call
6     }
7 }
```

Chained Arbitrary Call

SYNACKTIV



Chained Arbitrary Call

SYNACKTIV

Il nous faut un leak.

Obtention d'un leak

SYNACKTIV

- Une seule fonction pour envoyer/recevoir un paquet

```
int64_t NC(int model_id, int  socket, int is_encrypted, char* user, char *password,
           char  cmd_id, char seq_id, int timeout, /* [...] */
           char* inout_buffer, int* inout_len);
```

- Utilise des paramètres in/out 🤔

Obtention d'un leak

SYNACKTIV


```
1 char buffer[0x1000];
2 int length;
3
4 NC(/* ... */ buffer, &length, /* ... */); // GetTimeZone
5
6 // [...]
7
8 NC(/* ... */ buffer, &length, /* ... */); // GetCapacity
```

Obtention d'un leak

ETOOMANYVULNS

SYNACKTIV

```
1       // [...]
2
3       // Receive the response
4       received_len = TCPSocketRecv(socket, buffer, g_iNetMsgLen, timeout, is_encrypted, a15);
5
6       // Extract message length
7       msg_length = *(unsigned short *)&buffer[79];
8
9       // Copy the data
10      if ( inout_buffer && msg_length >= 2uLL )
11          memcpy(inout_buffer, &buffer[82], msg_length - 1); // 🤡🤡🤡
12
13      // [...]
```

Obtention d'un leak

ETOOMANYVULNS

SYNACKTIV

- Loot
  - Stack cookie
  - Adresse de `libFHDEV_Net.so`
  - Adresse de `liblewei63.so`
  - Adresses de stack
  - Adresses de heap
  - ~~Adresse d'une lib système~~ 🥲
  - CRASH 💀💀

Obtention d'un leak

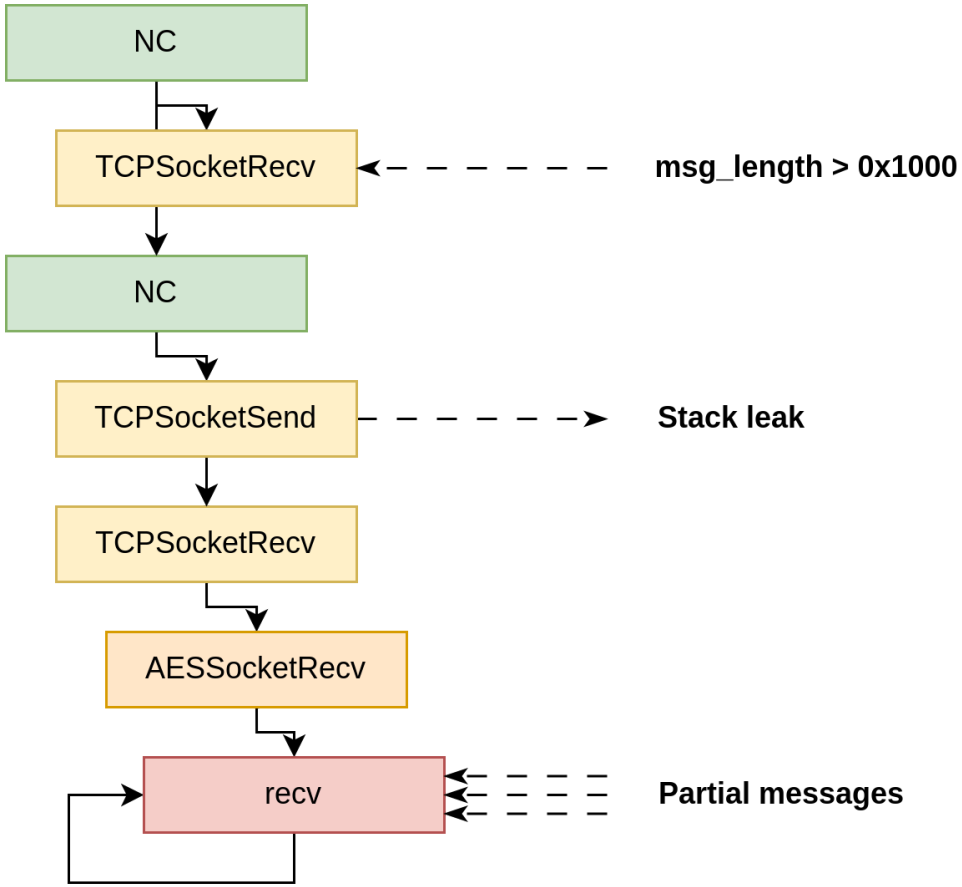
Neutralisation

SYNACKTIV

```
1      // [...]
2
3      // Receive the response
4      received_len = TCPSocketRecv(socket, buffer, g_iNetMsgLen, timeout, is_encrypted, a15);
5
6      // Extract message length
7      msg_length = *(unsigned short *)&buffer[79];
8
9      // Copy the data
10     if ( inout_buffer && msg_length >= 2uLL )
11         memcpy(inout_buffer, &buffer[82], msg_length - 1); // 🤡🤡🤡
12
13     // [...]
```

Obtention d'un leak

Neutralisation



Obtention d'un leak

ETOOMANYVULNS

SYNACKTIV

- Loot
  - Stack cookie
  - **Adresse de `libFHDEV_Net.so`**
  - Adresse de `liblewei63.so`
  - **Adresses de stack**
  - Adresses de heap
  - ~~Adresse d'une lib système~~
  - ~~CRASH~~ Thread bloqué 🚫

Better leak

SYNACKTIV

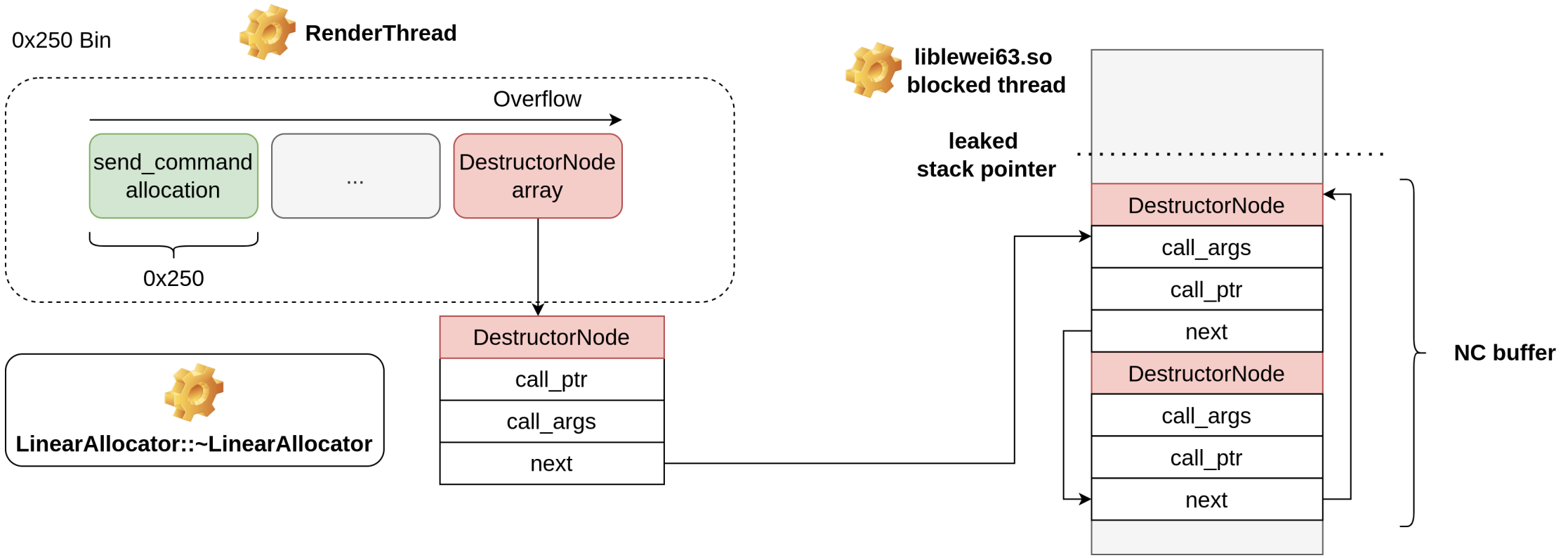
Les primitives actuelles :

- Arbitrary call + un argument controlé
- Un leak de la bibliothèque `libFHDEV_Net.so`
- Leak d'une adresse de stack

Il nous faut une primitive plus forte pour avoir l'adresse la `libc.so`

Better leak

Setup des calls



Better leak

SYNACKTIV

Bricolage d'une primitive: Gadget 1

```
1  int sub_738E8(device_t *user_info) {
2      // [..]
3      // Connect to target
4      int socket = TCPSocketCreate(
5      &user_info->target_ip,
6      user_info->target_port,
7      // [...]
8      );
9
10     // Send request
11     if ( !NC(...) ){
12         SocketClose(socket);
13         return 0LL;
14     }
15     // [...]
16 }
```

Better leak

Bricolage d'une primitive: Gadget 2

SYNACKTIV

```
1 int FHDEV_NET_SetCryptKey(char *key_addr)
2 {
3   // set a global variable
4   *(int128 *) g_aes_key = key_addr;
5   return result;
6 }
```

Better leak

GOT

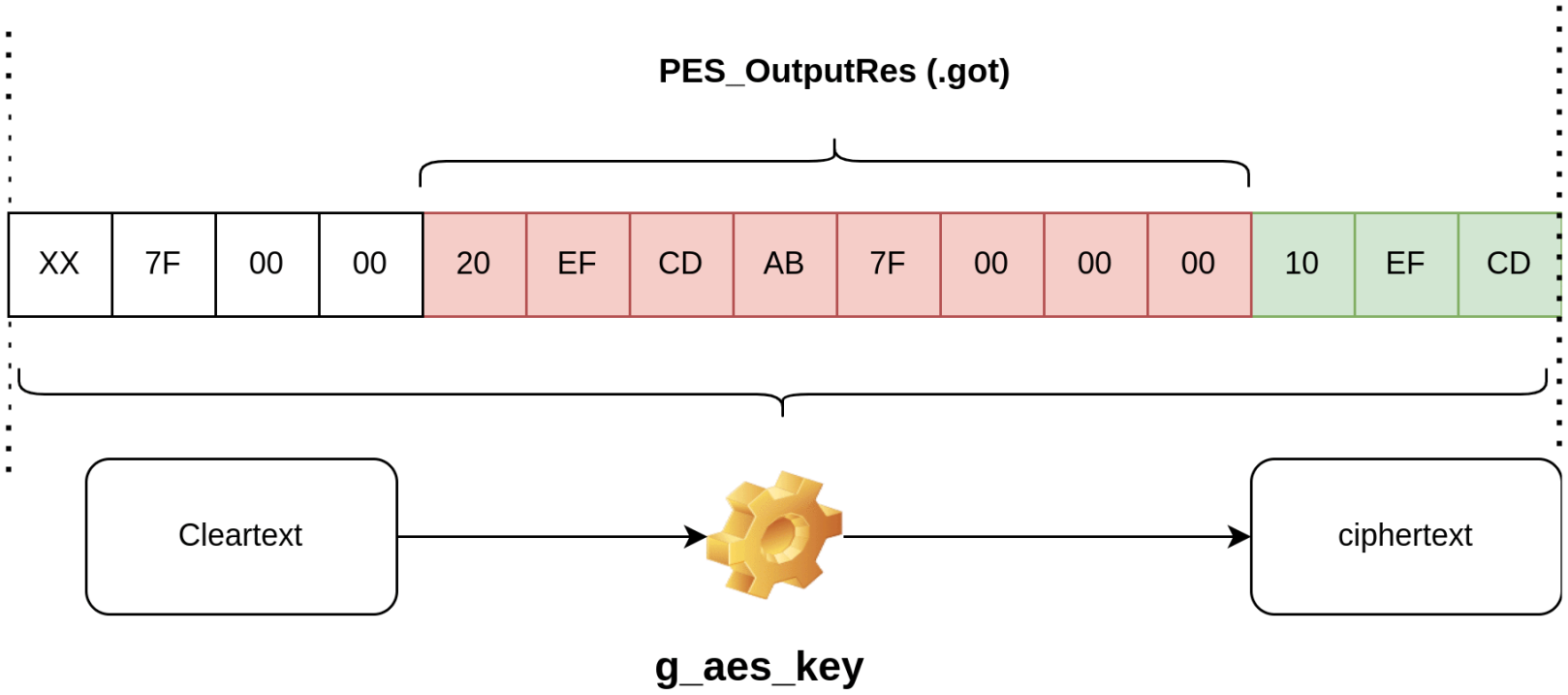
libc.so

```
.got:00000000000C80C0 pthread_create_ptr DCQ pthread_create    ; DATA XREF: .pthread_create↑o
.got:00000000000C80C0                                         ; .pthread_create+4↑r ...
.got:00000000000C80C8 PES_OutputPes_ptr DCQ PES_OutputPes      ; DATA XREF: .PES_OutputPes↑o
.got:00000000000C80C8                                         ; .PES_OutputPes+4↑r ...
.got:00000000000C80D0 Dev_GetHandleCount_ptr DCQ Dev_GetHandleCount
```

liFHDEV_NET.so

Better leak

Bruteforce !

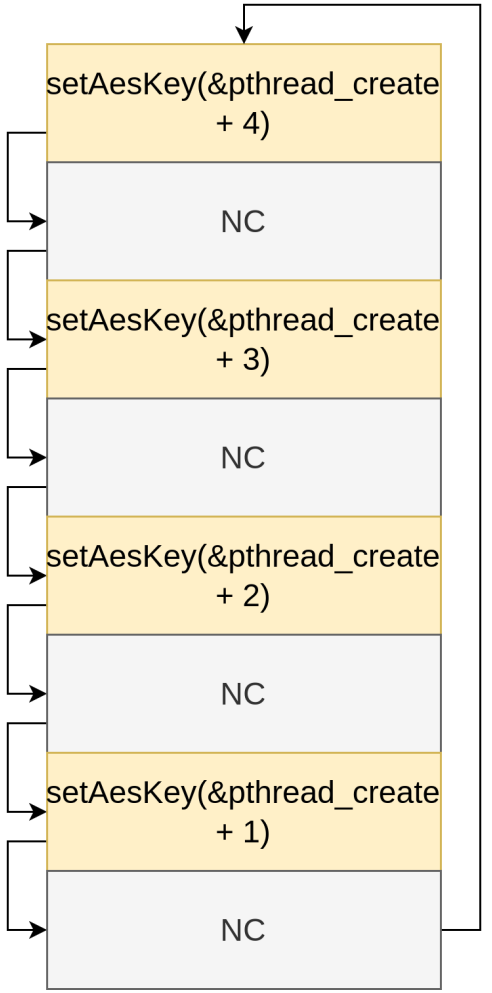


Finish him !

SYNACKTIV

On a :

- Leak de l'adresse de `libc.so`
- Leak de `libFHDEV_NET.so`
- Une boucle d'appels qui change la clé en boucle

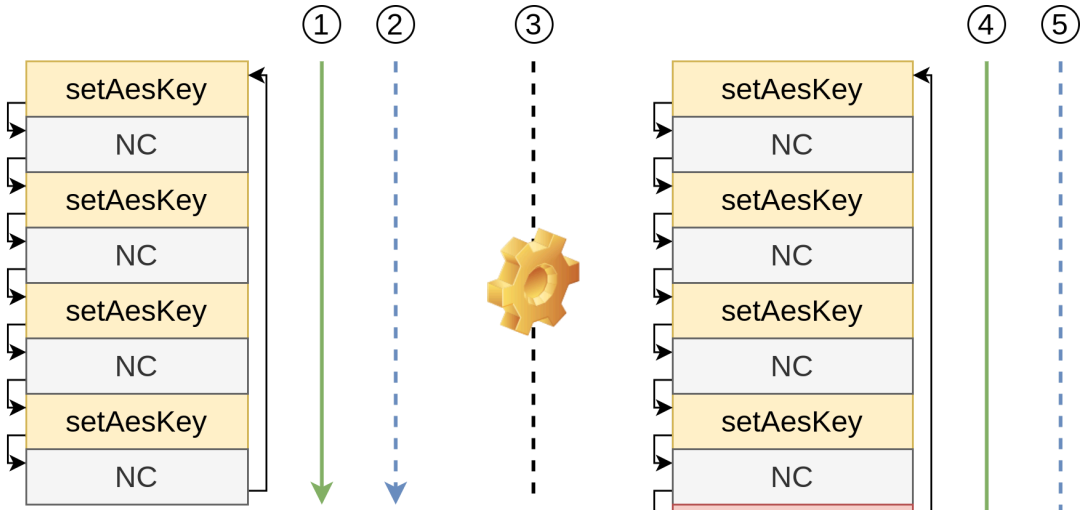


Finish him !

SYNACKTIV

Dans le thread bloqué :

- Si la réponse envoyée ne contient pas le bon `cmd_id`, la fonction rééssaye
- Cela réécrit quand même la stack



1 - Full write of leak COP chain
2 - Byte-per-byte write (crash neutralisation)
3 - Brute force of received leak
4 - Full write of command execution
5 - Byte-per-byte write (crash neutralisation)

Finish him !

SYNACKTIV

```
$ adb logcat -s EXPLOIT
--------- beginning of main
03-05 12:36:15.803 15692 15692 I EXPLOIT : uid=10322(u0_a322) gid=10322(u0_a322)
groups=10322(u0_a322),3003(inet),9997(everybody),20322(u0_a322_cache),50322(all_a322)
context=u:r:untrusted_app_32:s0:c66,c257,c512,c768
```

 Bonne bière à tous 

Si vous avez des questions on vous attend au bar :D

SYNACKTIV

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

https://x.com/synacktiv

https://bsky.app/profile/synacktiv.com

https://synacktiv.com

Un incident ? Contactez csirt@synacktiv.com