

Réunion flash

Point hebdomadaire

Duzes Florian




Sommaire

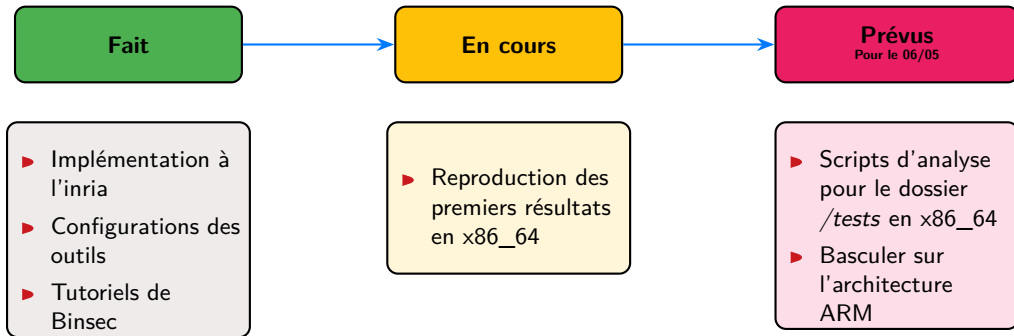
1. Un peu de hauteur
2. Protocole x86_64
3. Protocole ARM
4. Conclusion

01

Un peu de hauteur

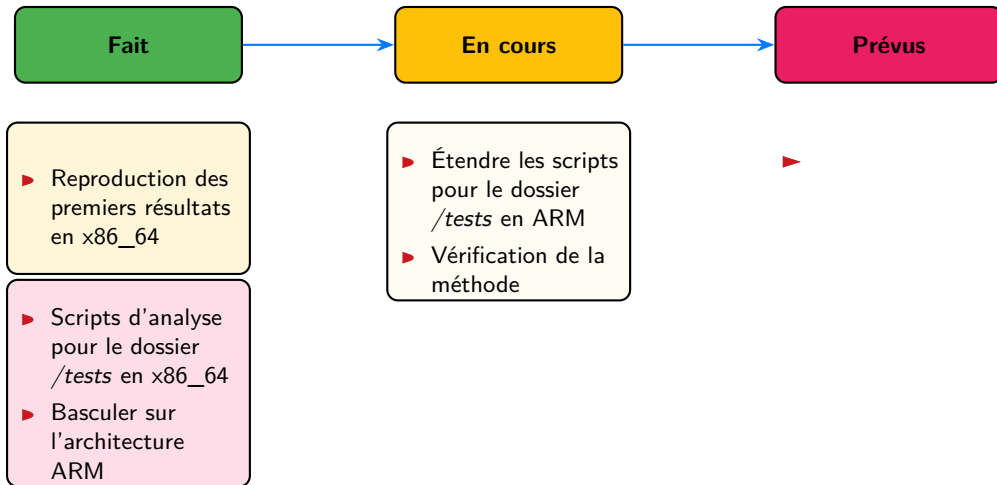


État des missions






Travail réalisé



02

Protocole x86_64



En amont

Avant la compilation

- Cibler la fonction à analyser

```
1 int main (int argc, char *argv[])
2 {
3     Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt
4     (cipher, tag, plain, BUF_SIZE, aead_aad, AAD_SIZE,
5      aead_key, aead_nonce);
6     exit(0);
7 }
```

Code 1 – chacha20Poly1305-128-binsec-test.c



En amont

Avant la compilation

- ▶ Cibler la fonction à analyser
- ▶ Dresser la structure des paramètres

```
1 void
2 Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt(
3     uint8_t *output,
4     uint8_t *tag,
5     uint8_t *input,
6     uint32_t input_len,
7     uint8_t *data,
8     uint32_t data_len,
9     uint8_t *key,
10    uint8_t *nonce
11 );
```

Code 2 – Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt.h



En amont

Avant la compilation

- ▶ Cibler la fonction à analyser
- ▶ Dresser la structure des paramètres
- ▶ Identifier nos paramètres

```
1 void
2 Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt(
3     uint8_t *output,
4     uint8_t *tag,
5     uint8_t *input,          //secret
6     uint32_t input_len,      //secret
7     uint8_t *data,           //secret
8     uint32_t data_len,       //secret
9     uint8_t *key,            //secret
10    uint8_t *nonce            //secret
11 );
```

Code 3 – Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt.h

Après la compilation

- Réalisation du coredump

```
break Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt  
run  
generate-core-file core.snapshot  
kill  
quit
```

Code 4 – script_dump

En aval

Après la compilation

- Réalisation du coredump
- Réalisation du script.ini

```
starting from core

halt at @[rsp, 8]

hook <Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt>
  (cipher, mac, msg, len, aad, aad_len, key, nonce) with
  @[len, 16384] := secret
  for i<64> in 0 to len - 1 do
    @[msg + i] := secret
  end
  @[aad_len, 12] := secret
  for i<64> in 0 to aad_len - 1 do
    @[aad + i] := secret
  end
  @[key, 32] := secret
  @[nonce, 12] := secret
end

explore all
```

Code 5 – study.ini

Après la compilation

- ▶ Réalisation du coredump
- ▶ Réalisation du script.ini
- ▶ Exécution de Binsec

```
TARGET = $(wildcard *.exe)
SNAP=core.snapshot
SCRIPT = study.ini

all: core-dump binsec

core-dump:
    gdb -x script_coredump $(TARGET)

binsec:
    binsec -sse -sse-depth 1000000 -sse-script $(SCRIPT)
        -checkct $(SNAP)
```

Code 6 – Makefile

03

Protocole ARM





Point de départ

Objectif fort

Ne pas utiliser de *coredump*.



Point de départ

Objectif fort

Ne pas utiliser de *coredump*.

Conclusion

Analyser le binaire.



Point de départ

Objectif fort

Ne pas utiliser de *coredump*.

Conclusion

Analyser le binaire.

- ▶ Adresse de la fonction cible.
- ▶ Adresse des paramètres
- ▶ Adresse de fin de fonction



Premier essai

```
sp<64> := 0x400fc0

# End of disas binsec
@[sp, 64] := 0x401120 as return_address

# Information from the objdump -x
@[<.plt>, 22] from file
@[<.text>, 348] from file
@[<.fini>, 2] from file
@[<.rodata>, 30] from file
@[<.eh_frame_hdr>, 17] from file
@[<.eh_frame>, 84] from file
@[<.init_array>, 1] from file
@[<.fini_array>, 1] from file
@[<.dynamic>, 60] from file
@[<.got>, 2] from file
@[<.got.plt>, 12] from file
@[<.data>, 42] from file
@[<.bss>, 0] from file
```

```
# Looking in binsec disas
@[0x400fc0, 64] := 0x4294967184 as
    Hacl_Chacha20_chacha20_encrypt
hook <Hacl_Chacha20_chacha20_encrypt> (len,
    out, text, key, n, ctr) with
    for i<64> in 0 to len - 1 do
        @[out + i] := secret
    end
    for i<64> in 0 to len - 1 do
        @[text + i] := secret
    end
    @[key, 32] := secret
    @[n, 12] := secret
end

starting from <main>
explore all
```

Code 7 – something_like_that.ini



Essai n° ?

```
#fin de la zone de .text
@[sp, 8] := 0x00404860 as return_address

#fin arbitraire : fin de calculs
@[sp, 8] := 0x004005c0 as return_address

# load common sections from ELF file
load sections .plt, .text, .rodata, .data, .got, .got.plt, .bss from file

secret global plain, aead_aad, aead_key, aead_nonce

starting from <main>
with concrete stack pointer

halt at return_address
explore all
```

Code 8 – script.ini

Ça marche !?

Exécution Binsec ne lève pas d'erreur

- ▶ **Uknwon** avec l'adresse de fin de la section `.text` - fin d'exécution de Binsec.

Ça marche !?

Exécution Binsec ne lève pas d'erreur

- ▶ **Uknwon** avec l'adresse de fin de la section `.text` - fin d'exécution de Binsec.
- ▶ **Secure** avec l'adresse de fin d'exécution (section `.plt`).

Ça marche !?

Exécution Binsec ne lève pas d'erreur

- ▶ **Uknown** avec l'adresse de fin de la section `.text` - fin d'exécution de Binsec.
- ▶ **Secure** avec l'adresse de fin d'exécution (section `.plt`).

Est-ce que la fonction est entièrement analysée ?

Tentatives sur cas simple

```
1 void f(uint8_t msg[BUF_SIZE], uint8_t
   key[KEY_SIZE])
2 {
3     for (int i=0; i<BUF_SIZE; i++)
4     {
5         msg[i] = msg[i] ^ key[i%
           KEY_SIZE];
6     }
7 }
```

Code 9 – good_ct.c

```
1 void f(uint8_t msg[BUF_SIZE], uint8_t
   key[KEY_SIZE])
2 {
3     if( key[0] > key[1] )
4     {
5         msg[0] = msg[0] ^ key[0];
6     }
7 }
```

Code 10 – bad_ct.c



Vérification

Tentatives sur cas simple

```
1 void f(uint8_t msg[BUF_SIZE], uint8_t  
   key[KEY_SIZE])  
2 {  
3     for (int i=0; i<BUF_SIZE; i++)  
4     {  
5         msg[i] = msg[i] ^ key[i%  
           KEY_SIZE];  
6     }  
7 }
```

Code 11 – good_ct.c

```
1 void f(uint8_t msg[BUF_SIZE], uint8_t  
   key[KEY_SIZE])  
2 {  
3     if( key[0] > key[1] )  
4     {  
5         msg[0] = msg[0] ^ key[0];  
6     }  
7 }
```

Code 12 – bad_ct.c

Binsec :

Secure

Insecure



Comment vérifier ?

Idées de vérifications

Dans l'objectif d'automatisation :

- ▶ Détection fine des adresses de fin.




Comment vérifier ?

Idées de vérifications

Dans l'objectif d'automatisation :

- ▶ Détection fine des adresses de fin.
- ▶ Application d'une fonction "*fin_test*".

04 Conclusion



Conclusion

Protocole x86_64

- ▶ terminé

Protocole ARM

- ▶ target=aarch64-none-linux-gnu
- ▶ à éclaircir/terminé

Lancement vers d'autres architectures ou avancement dans l'automatisation ?

Merci.

