Solution du challenge SSTIC 2013

Emilien Girault

ANSSI/COSSI/DTO/BAI

06/06/13

Plan

- Analyse d'une capture réseau
  - Détermination des canaux cachés
  - Déchiffrement de l'archive
- Etude d'un FPGA
  - Déduction du jeu d'instructions
  - Rétroconception du programme
- Détricotage de fichier PostScript
  - Analyse des différents *stages*
  - Déchiffrement
- Désobfuscation d'une vCard
  - Récupération de l'adresse e-mail

Plan

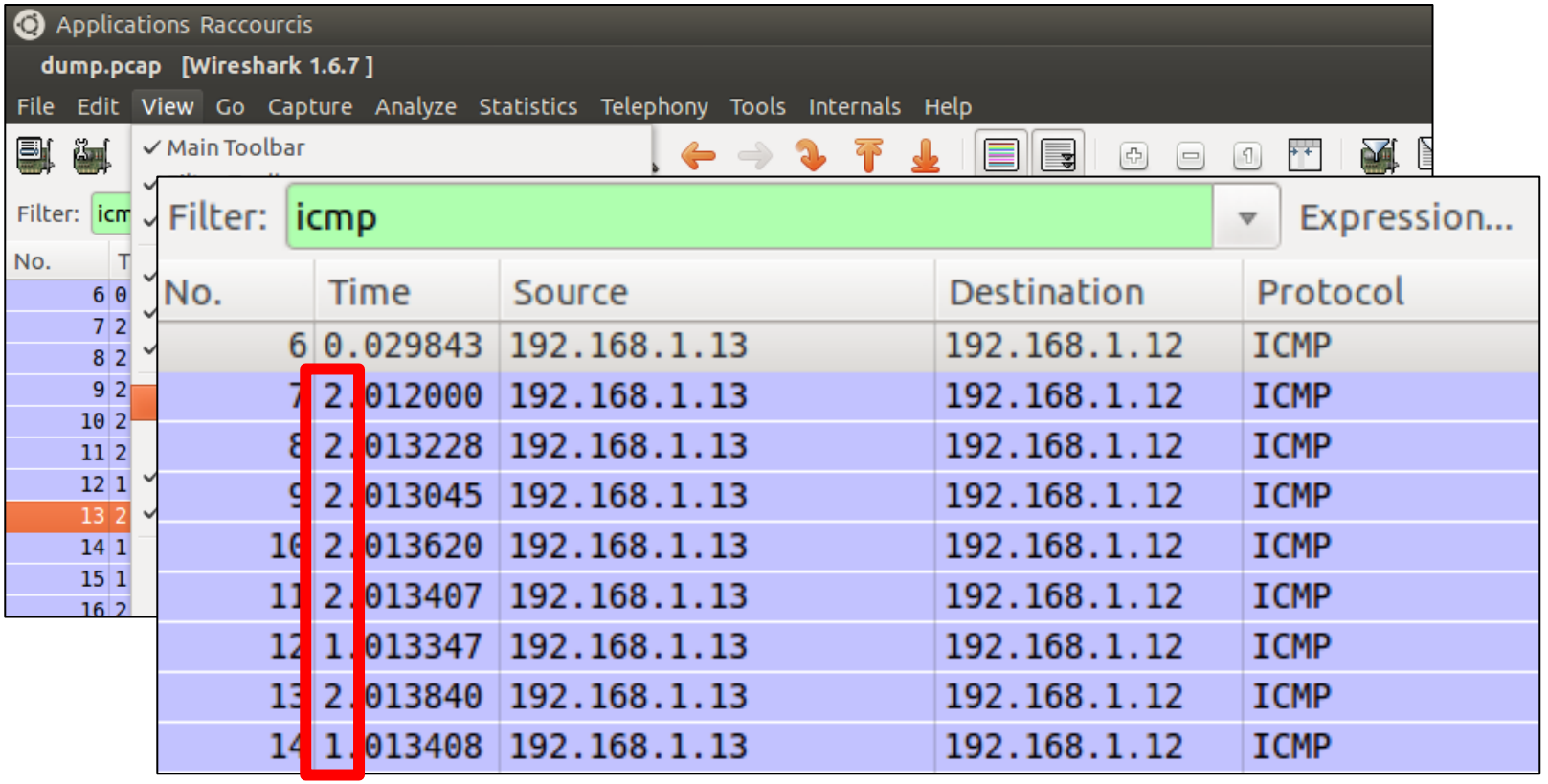
- Analyse d'une capture réseau
  - Détermination des canaux cachés
  - Déchiffrement de l'archive
- Etude d'un FPGA
  - Déduction du jeu d'instructions
  - Rétroconception du programme
- Détricotage de fichier PostScript
  - Analyse des différents *stages*
  - Déchiffrement
- Désobfuscation d'une vCard
  - Récupération de l'adresse e-mail

Capture réseau

- Capture au format PCAP
- Echanges entre deux machines A et B
- Composition :
  - Connexion TCP de A sur port 1234 de B : message
    - Une archive a été envoyée chiffrée. IV et checksum donnés.
    - La clé a été transmise par canaux cachés.
  - Série de 65 paquets ICMP « Echo Request »
  - Connexion FTP elle-même composée :
    - D'un canal de contrôle* sur le port 21
    - D'un transfert de données en mode passif (l'archive chiffrée)

* Le mot de passe utilisé (`sstic`) ne respecte pas les recommandations de l'ANSSI (http://www.ssi.gouv.fr/IMG/pdf/NP_MDP_NoteTech.pdf)

Canal caché temporel



1 bit d’information

Canaux cachés

- Examen des différences entre chaque paquet ICMP
  - ip.tos : [2, 4] ➔ 1 bit d'information
  - ip.ttl : [10, 20, 30, 40] ➔ 2 bits d'information

- 4 bits d'info par paquet * 64 paquets = 256 bits de clé

- Autres problèmes :
  - Codage de ces bits d'information ?
  - Agencement des bits d'info dans 1 *nibble* de clé ?
  - Agencement des *nibbles* au sein de la clé ?

Capture – solution

- Bruteforce de :
  - La représentation des champs
  - Leur agencement et celui des *nibbles*
- Conditions d'arrêt :
  - Padding PKCS7 valide || Hash MD5 valide || Header GZIP
- Script utilisant Scapy (cf. solution)

```
$ time ./step1.py
dd8cf2d52e69aafb734e3acd0e4a69e83ed93bc4870ecd0d5b6faad86a63ae94

real 0m3.853s
user 0m3.756s
sys 0m0.080s
```

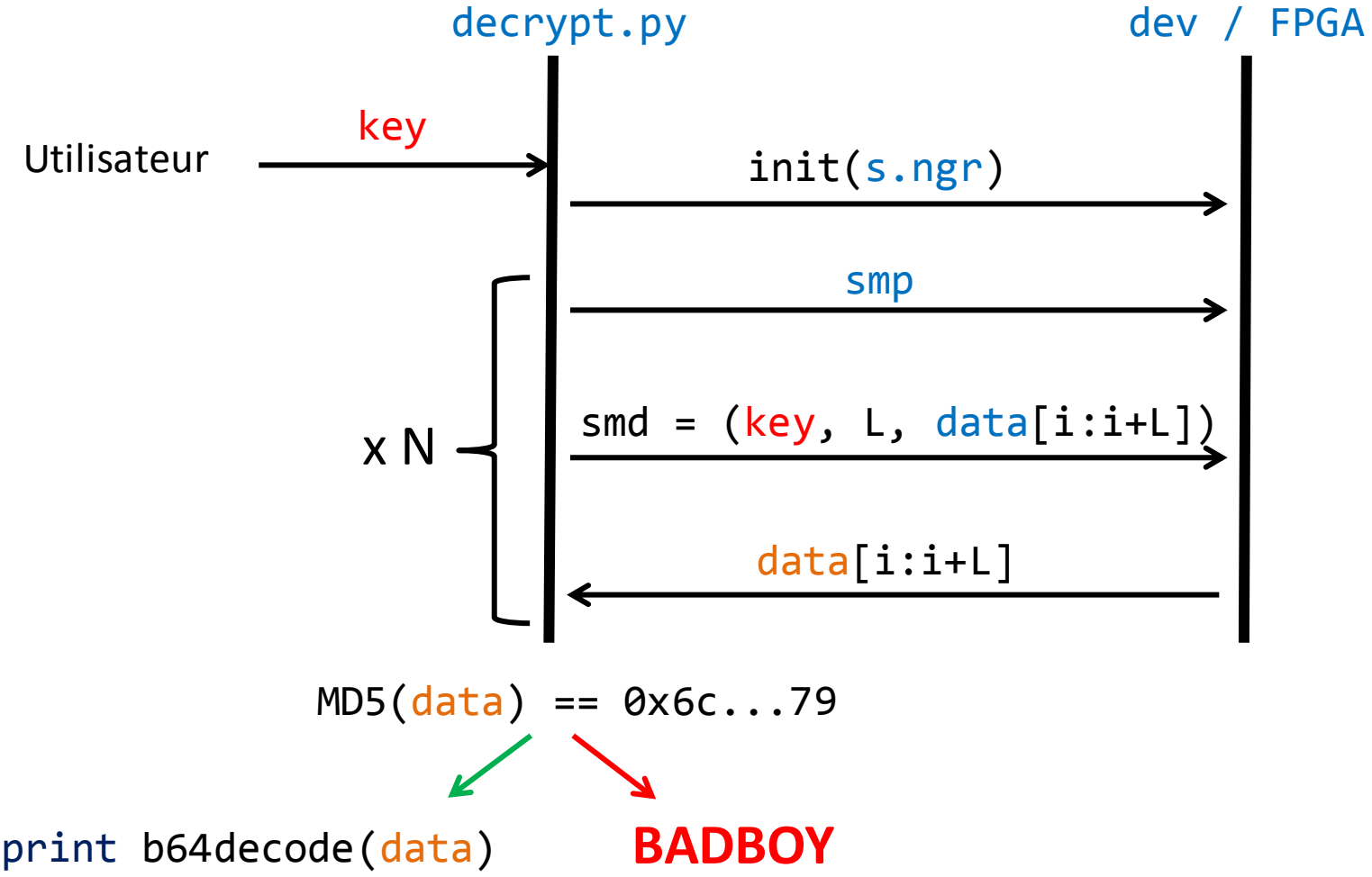
Plan

- Analyse d'une capture réseau
  - Détermination des canaux cachés
  - Déchiffrement de l'archive
- Etude d'un FPGA
  - Déduction du jeu d'instructions
  - Rétroconception du programme
- Détricotage de fichier PostScript
  - Analyse des différents *stages*
  - Déchiffrement
- Désobfuscation d'une vCard
  - Récupération de l'adresse e-mail

Etape 2 – Découverte

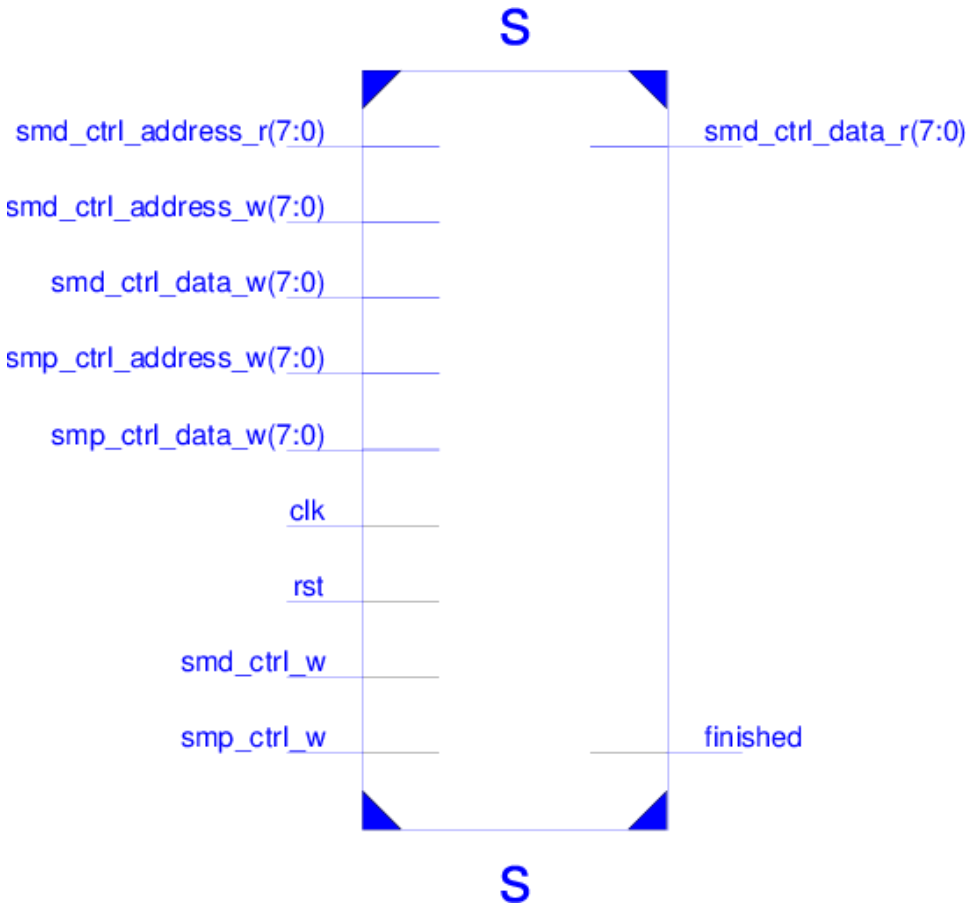
- Contenu de l'archive
  - `data` : fichier binaire, a priori chiffré
  - `decrypt.py` : script de déchiffrement (de `data` ?)
    - Fait appel à un module `dev`, inconnu
  - `smp.py` : liste d'octets
  - `s.ngr` : fichier au format a priori inconnu
    - Google : NGR = design de FPGA XILINX obtenu à partir de code HDL
    - Visualisable avec *ISE Design Suite*

Decrypt.py : Algorithme

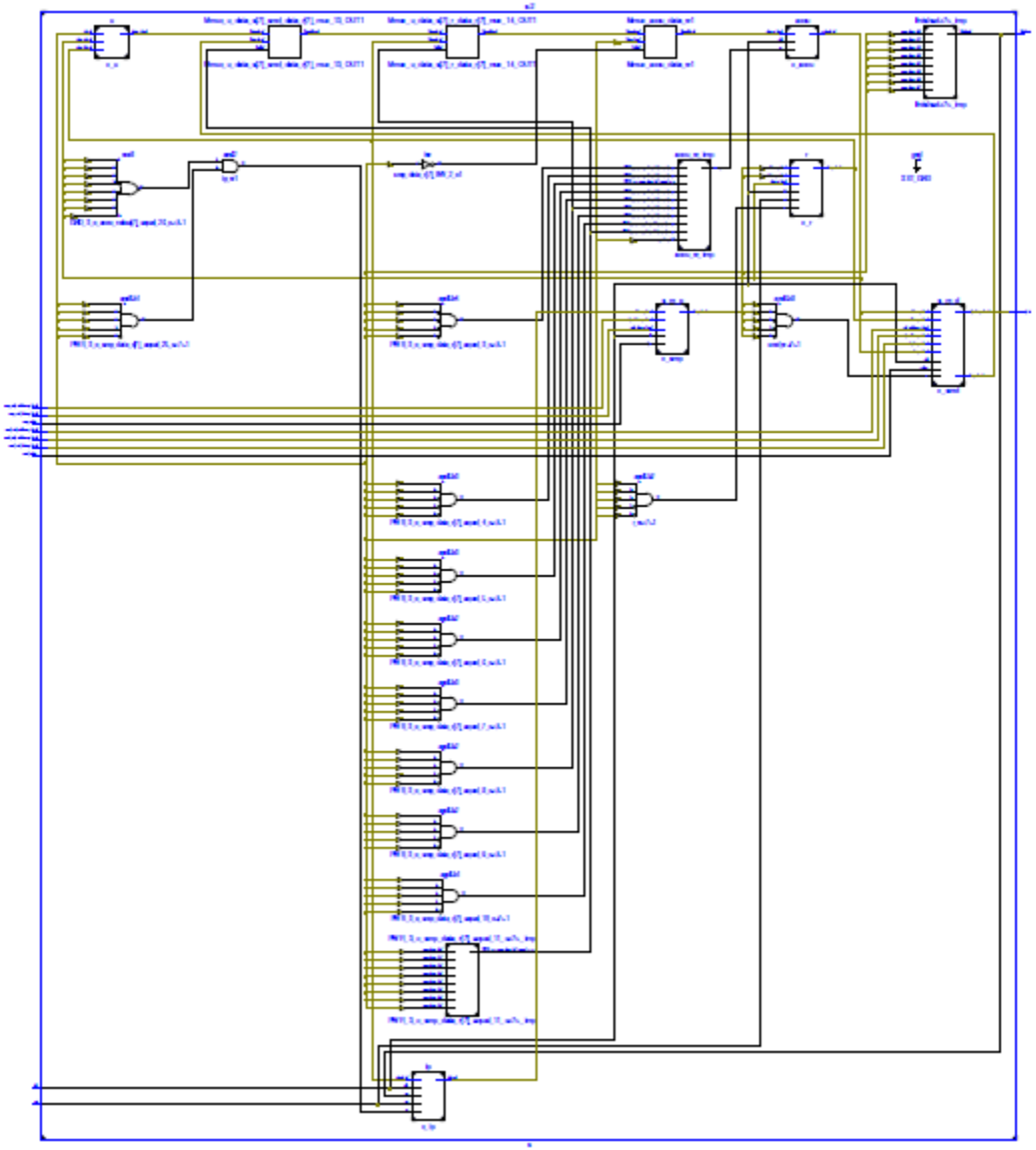


➔ Nécessité d'analyser le FPGA pour comprendre le traitement opéré sur `data`

FPGA – Vue extérieure



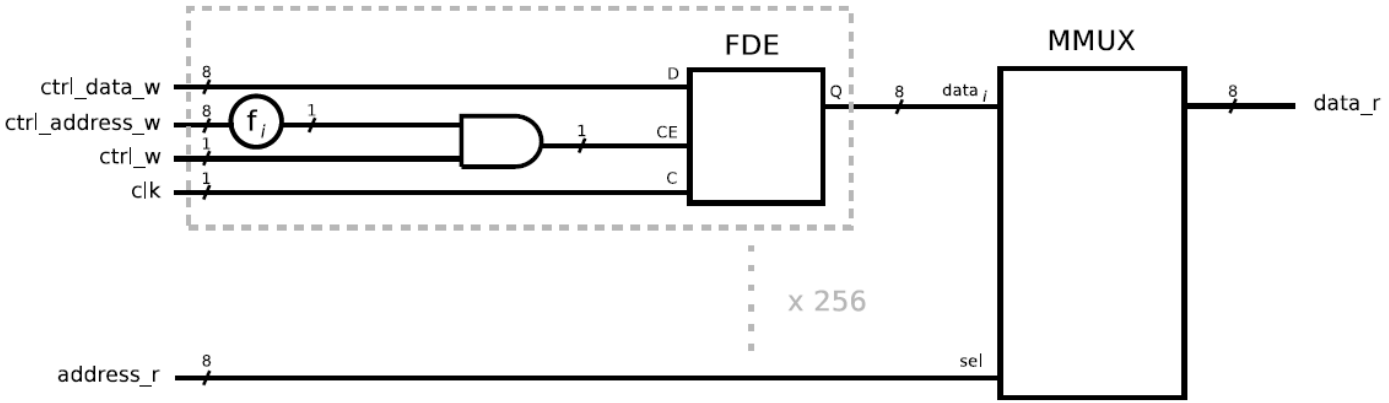
FPGA – Vue intérieure



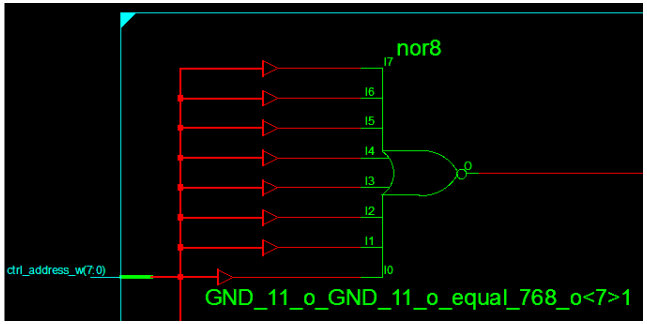
Méthodologie

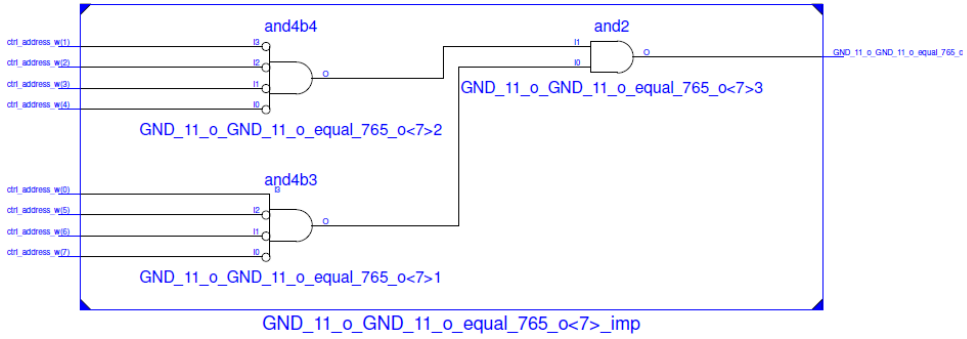
- Analyse de chaque composant, en commençant par celui ayant le plus de pins reliées aux entrées de `s` :
  - Regarder d'où viennent les entrées
  - Déterminer la logique interne du composant
    - Récursivement ☺
  - Simplifier au maximum les motifs de composants répétés
  - Voir où vont les pins de sortie
    - Repérer les *bus* reliant la sortie d'un composant à plusieurs entrées d'autres composants

Exemple : composant smp



Vue simplifiée de smp





➔ Logique d'adressage

… and so on !

- `smp` et `smd` : 2 mémoires de 256 octets
- `r` : ensemble de 8 registres
- `accu` : registre supplémentaire
- `u` : unité logique
- `ip` : registre incrémenté

- Déductions
  - Le FPGA implémente un CPU d'architecture « Harvard »
  - `smp` = programme, `smd` = données
  - sortie de `smp` = *opcode*, décodé et interprété

Jeu d'instructions

Instruction	Opcode
AND accu, r	10001rrr
OR accu, r	10010rrr
NOT accu, r	10100rrr
OR accu, 0x80	11100xxx
SHL accu, 1	11011xxx
MOV reg, accu	10110rrr
MOV accu, r	10101rrr
MOV accu, 0bccccccc	0ccccccc
LOAD accu, [accu]	11010000
STORE [r], accu	11000rrr
JZ r	10111rrr
RET	11001000

Analyse du programme

- Coder !
  - Un désassembleur
    - Option 1 : demander à Yoann Guillot ☺
    - Option 2 : le faire soi-même
      - Instructions faciles à décoder
      - Pseudo-exécution concrète pour simplifier les sauts
  - Un émulateur / débogueur

Code désassemblé

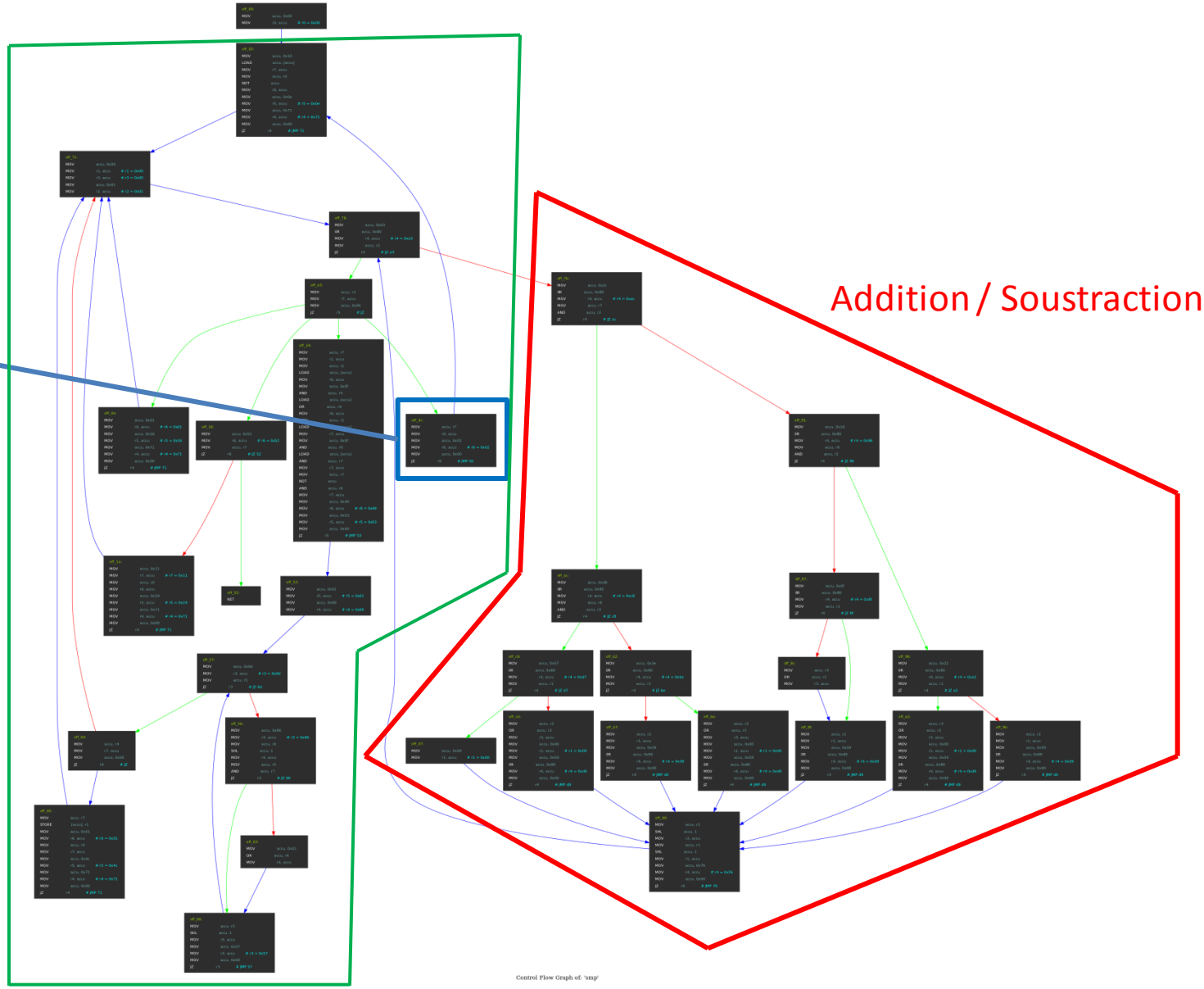
```
00: MOV accu, 0x00
01: MOV r0, accu        # r0 = 0x00
02: MOV accu, 0x10
03: LOAD accu, [accu]
04: MOV r7, accu
05: MOV accu, r0
06: NOT accu
07: MOV r6, accu
08: MOV accu, 0x0e
09: MOV r5, accu        # r5 = 0x0e
0a: MOV accu, 0x71
0b: MOV r4, accu        # r4 = 0x71
0c: MOV accu, 0x00
0d: JZ r4               # JMP 71

0e: MOV accu, 0x01
0f: MOV r6, accu        # r6 = 0x01
10: MOV accu, 0x16
11: MOV r5, accu        # r5 = 0x16
[...]
```

Graphe

Boucle principale

Cas d'arrêt

Addition / Soustraction



FPGA – Solution

- Algo final

```
for i in xrange(len(bloc)):
  c = (bloc[i] ^ key[i & 0xf])
  c = bitmirror(c)
  bloc[i] = c
```

- Bruteforce en Python
  - Chaque caractère de la clé est indépendant des autres
  - Condition d'arrêt : contenu déchiffré == Base64 valide
  - Attention au *charset* utilisé (newlines…)

```
$ time ./step2.py
25a16d1ed323ac65c658ef16bcdc83e6

real 0m0.206s
user 0m0.200s
sys 0m0.004s
```

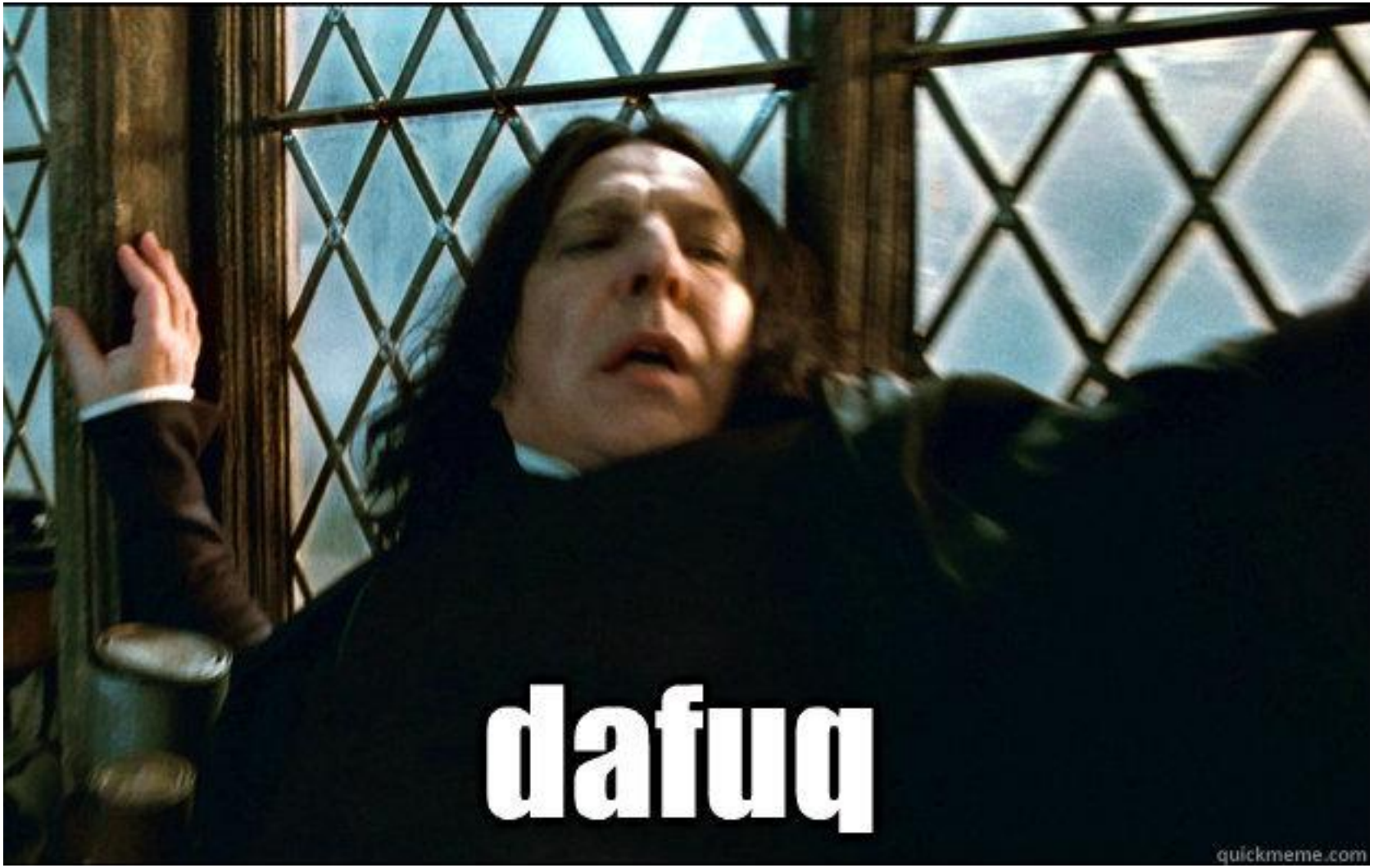
Plan

- Analyse d'une capture réseau
  - Détermination des canaux cachés
  - Déchiffrement de l'archive
- Etude d'un FPGA
  - Déduction du jeu d'instructions
  - Rétroconception du programme
- Détricotage de fichier PostScript
  - Analyse des différents *stages*
  - Déchiffrement
- Désobfuscation d'une vCard
  - Récupération de l'adresse e-mail

Contenu base64-décodé

```
/I1 currentfile 0 (cafebabe) /SubFileDecode filter /ASCIIHexDecode filter /ReusableStreamDecode filter
cf760bc77db1f282e881ede9a10122b220887466b973b854218b85c230d6733ab459fda9a879973664130312d5ff3e1a8[...]
cafebabe def /I2 currentfile 0 (cafebabe) /SubFileDecode filter /ASCIIHexDecode filter
/ReusableStreamDecode filter c598d7cc5b5354440b0613490c45483d4e7b0f6c0368120f100107050103020212 09[...]
cafebabe def /I3 currentfile 0 (cafebabe) /SubFileDecode filter /ASCIIHexDecode filter
/ReusableStreamDecode filter 338f25667eb4ec47763dab51c3fa41cba329e18536b83159b3a690a0265ec519aae9[...]
cafebabe def /I4 currentfile 0 (cafebabe) /SubFileDecode filter /ASCIIHexDecode filter
/ReusableStreamDecode filter 8ae98ae900000000020002000316121411455556000855540012 4e5245114e011d1b[...]
cafebabe def /error { (%stderr)(w) file exch writestring } bind def errordict /handleerror { quit }
put /main { mark shellarguments { counttomark 1 eq { dup length exch /ReusableStreamDecode filter exch
2 idiv string readhexstring pop dup length 16 eq { I1 32 exch mark 1 index resetfile 1 index {
counttomark 1 sub index counttomark 2 add index 4 mul string readstring pop dup () eq {pop exit} if }
loop counttomark -1 roll counttomark 1 add 1 roll ] 4 1 roll pop pop pop I2 0 index resetfile 61440
string readstring pop dup 3 index 2 2 getinterval dup exch dup length 2 index length add string dup
dup 4 2 roll copy length 4 -1 roll putinterval 0 0 1 1 {pop 2 index length} for exch 1 sub { 3 copy
exch length getinterval 2 index mark 3 1 roll 0 1 3 -1 roll dup length 1 sub exch 4 1 roll { dup 3 2
roll dup 5 1 roll exch get 3 1 roll exch dup 5 1 roll exch get xor 3 1 roll } for pop pop ] dup length
string 0 3 -1 roll { 3 -1 roll dup 4 1 roll exch 2 index exch put 1 add } forall pop 4 -1 roll dup 5 1
roll 3 1 roll dup 4 1 roll putinterval exch pop } for 0 1 1 {pop pop} for cvx exec I3 resetfile I4 0
index resetfile 61440 string readstring pop dup 3 index 0 2 getinterval dup exch dup length 2 index
length add string dup dup 4 2 roll copy length 4 -1 roll putinterval 0 0 1 1 {pop 2 index length} for
exch 1 sub { 3 copy exch length getinterval 2 index mark 3 1 roll 0 1 3 -1 roll dup length 1 sub exch
4 1 roll { dup 3 2 roll dup 5 1 roll exch get 3 1 roll exch dup 5 1 roll exch get xor 3 1 roll } for
pop pop ] dup length string 0 3 -1 roll { 3 -1 roll dup 4 1 roll exch 2 index exch put 1 add } forall
pop 4 -1 roll dup 5 1 roll 3 1 roll dup 4 1 roll putinterval exch pop } for 0 1 1 {pop pop} for cvx
exec } if false } { (no key provided\n) error true } ifelse } { (missing '--' preceding script
file\n) error true } ifelse { (usage: gs -- script.ps key\n) error flush } if } bind def main clear
quit
```

Contenu base64-décodé



PostScript Oriented Programming

- PostScript est un véritable langage de programmation
  - Interpréteur : `gs`
  - Cf. documentation de référence
- Pile d'opérateurs, notation polonaise inversée
- Aucun débogueur fonctionnel
  - *PSAlter* : conçu pour Windows 95, ne gère pas les filtres…
  - Débogage manuel, **très** laborieux
  - Rajout massif d'instructions de debug
    - `dup ==` : affiche le dernier élément empilé
    - `pstack quit` : affiche toute la pile et quitte

Script.gs

- Prend en paramètre une clé `key` de 16 octets
- Déclare 4 variables (chaînes) : `I1`, `I2`, `I3` et `I4`
- Déchiffre `I2` avec `key[2:4]`
  - Algo : « XOR CBC »
- Exécute `I2`
- Déchiffre `I4` avec `key[0:2]`
  - Même algo
- Exécute `I4`


PostScript in PostScript
We need to go deeper.

I2 et I4

- I2
  - Très fastidieux à reverser
  - Ne fait que déclarer une fonction `calc`
  - Nombreuses constantes
    - Google ➔ MD5
- I4
  - Boucle de déchiffrement d'`I1` à l'aide de `key[i*2:i*2+4]`
  - Chaque tour :
    - Sous-boucle avec 10240 itérations
    - Compare le buffer obtenu avec `I3[i*16:(i+1)*16]`
  - Longue pause si échec d'un tour

PostScript – Solution

- Déchiffrement de `I2` et `I4`
  - Bruteforce du début de la clé pour obtenir du PS valide
- Déchiffrement d'`I1`
  - Premier essai : *wrapper* Python + PostScript
    - Trop lent et fastidieux
  - Solution retenue : réimplémentation de l'algo en Python
    - Bruteforce par blocs de 2 caractères
    - Relativement long, mais parallélisable

```
$ time ./step3.py
w00t! See output.bin

real 53m30.735s     # sur PC scientifique 12 coeurs
user 631m22.780s
sys 0m2.208s
```

Plan

- Analyse d'une capture réseau
  - Détermination des canaux cachés
  - Déchiffrement de l'archive
- Etude d'un FPGA
  - Déduction du jeu d'instructions
  - Rétroconception du programme
- Détricotage de fichier PostScript
  - Analyse des différents *stages*
  - Déchiffrement de la *payload*
- Désobfuscation d'une vCard
  - Récupération de l'adresse e-mail

vCard

```
$ file output.bin
output.bin: vCard visiting card
$ mv output.bin vcard.txt
$ grep EMAIL vcard.txt
EMAIL;INTERNET:AngelaJMatthews@dayrep.com
EMAIL;INTERNET:PatriciaDBuenrostro@armyspy.com
EMAIL;INTERNET:sys_socketpair stub_fork sys_socketpair sys_getsockopt
sys_socketpair sys_ptrace sys_shutdown sys_ptrace sys_getsockopt
sys_bind sys_getuid sys_bind sys_ptrace sys_getsockname sys_ptrace
stub_fork stub_fork sys_getpeername sys_setsockopt sys_getrusage
sys_sysinfo sys_getsockname sys_shutdown sys_getsockopt sys_getuid
sys_sysinfo sys_getsockopt sys_getrlimit sys_setsockopt sys_shutdown
stub_clone sys_times sys_shutdown sys_getrusage sys_socketpair
sys_setsockopt stub_clone sys_getpeername sys_socketpair stub_clone
sys_semget sys_sysinfo sys_getgid sys_getrlimit sys_getegid
sys_getegid sys_ptrace sys_getppid sys_syslog sys_ptrace
sys_sendmsg sys_getgroups sys_getgroups sys_setgroups sys_setuid
sys_sysinfo sys_sendmsg sys_getpgrp sys_setregid sys_syslog
EMAIL;INTERNET:MarleneREllender@teleworm.us
EMAIL;INTERNET:ErickCGould@teleworm.us
[...]
```

One line to rule them all…

```
$ for p in $( \
    for s in $(grep socket vcard.txt | cut -d: -f2); do echo $s; done \
    | sed -re 's/(sys|stub)_/__NR_/' \
    ); \
  do \
    grep $p /usr/src/linux-headers-$(uname -r)/arch/x86/include/asm/unistd_64.h \
    | grep define; \
  done \
  | python -c "import sys; print ''.join(chr(int(i)) for i in
              sys.stdin.read().split() if i.isdigit())"
```

59575e0e71f1e3e9946bc307fc7a608d0b568458@challenge.sstic.org

Questions ?

- Solution complète (bientôt) disponible en ligne

- Remerciements
  - root@challenge.sstic.org
  - Yoann Guillot
  - Julien Perrot
  - Axel Souchet