université de BORDEAUX

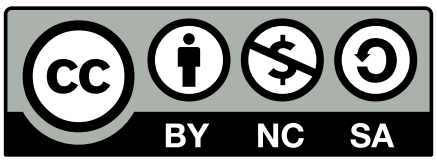
Analyse de malware statique et dynamique

Les outils et quelques cas pratiques

Erwan Grelet & Amélie Guémon

Master CSI, Université de Bordeaux, France

4 mars 2017



Introduction

Quelques statistiques . . .

- 58.31% des emails échangés durant l'année 2016 étaient des spams, soit 3 points de plus qu'en 2015 ;
- Plus de 90% des emails de phishing contenaient des ransomwares ;
- En 2016, 1 088 900 victimes de banking trojans, 30% de plus qu'en 2015 ;
- En septembre 2016, attaque DDOS de 1 Tb/s sur les serveurs d'OVH France ;
- Code source du botnet Mirai, sur Linux, disponible.

Introduction

Quelques types de menaces :

- Adware ;
- Spyware ;
- Worm ;
- Trojan ;
- Rootkit ;
- Backdoors ;
- Keyloggers ;
- Rogue security software ;
- Ransomware ;
- Browser Hijacker ;
- Botnet ;
- ...

Plan

Plan

Les Techniques d'Analyses

université de BORDEAUX

- Analyse statique : étude du binaire suspect sans l'exécuter ;

Les Techniques d'Analyses

université de BORDEAUX

- Analyse statique : étude du binaire suspect sans l'exécuter ;
- Analyse dynamique : étude au cours de l'exécution et de l'infection ;

Les Techniques d'Analyses

université de BORDEAUX

- Analyse statique : étude du binaire suspect sans l'exécuter ;
- Analyse dynamique : étude au cours de l'exécution et de l'infection ;
- Analyse mémoire : étude de l'image mémoire après infection.

université de BORDEAUX

- Examen de l'exécutable :

```
$> readelf -h ch23
En-tête ELF:
  Magique:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Classe:                            ELF32
  Données:                           complément à 2, système à octets de
poids faible d'abord (little endian)
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  Version ABI:                       0
  Type:                              EXEC (fichier exécutable)
  Machine:                           Intel 80386
  Version:                           0x1
  Adresse du point d'entrée:         0x8048450
  Début des en-têtes de programme :  52 (octets dans le fichier)
  Début des en-têtes de section :    2404 (octets dans le fichier)
  [...]
```

Figure: Données comprises dans l'en-tête d'un binaire ELF

- Recherche de chaînes de caractères :

```
$> /tmp/HB3$ strings ch23
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
strncpy
__stack_chk_fail
printf
strlen
memset
__libc_start_main
Usage : %s <your name>
.init
.text
.data
.bss
main
_init
```

Figure: Résultat partiel de la commande *strings* sur un binaire

Les Techniques d'Analyses – Statique

université de BORDEAUX

- Protections du binaire :

- Binaire stripped ;
- Détection de débugueur ;
- Anti-virtualisation (VM) ;
- Anti-dumping ;
- Anti-tampering ;
- Packing ;

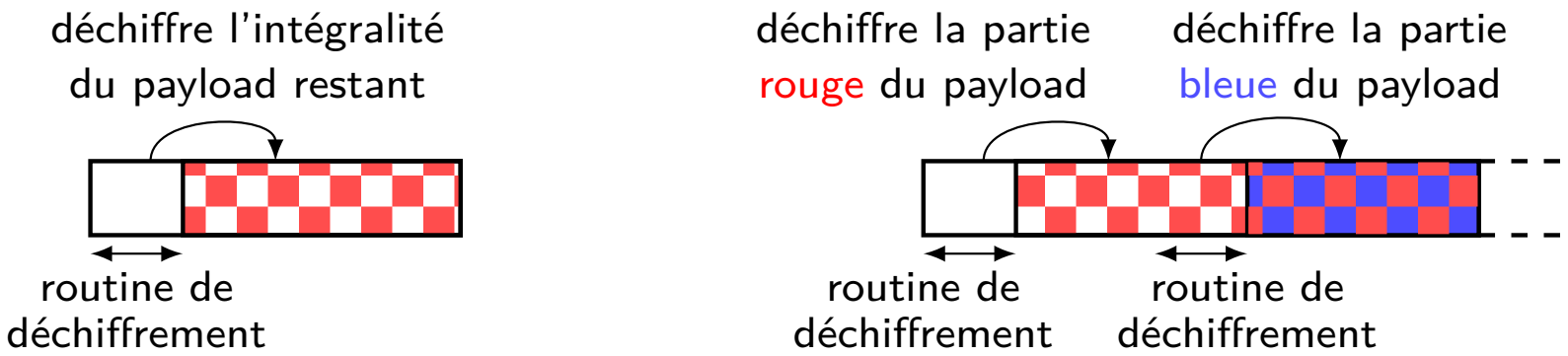


Figure: Différentes formes de packer chiffrant

Les Techniques d'Analyses – Statique

université de BORDEAUX

- Lecture de code :

```
[0x00405fa0 36% 245 /mnt/sdb/sage_dump]> pd $r @ main
/ (fcn) main 124
|   main ();
|              ; CALL XREF from 0x00406020 (entry)
|           0x00405fa0      e80bf9ffff     call init_and_check          ;[1]
|           0x00405fa5      e826fcffff     call print_debug_info        ;[2]; floating_p
|           0x00405faa      e871feffff     call c_fork                  ;[3]
|           0x00405faf      e8dcfeffff     call check_fork              ;[4]
|           0x00405fb4      85c0           test eax, eax
|        ,=< 0x00405fb6      7403           je 0x405fbb                  ;[5]
|       .--> 0x00405fb8      33c0           xor eax, eax
|       ||   0x00405fba      c3             ret
|       |`-> 0x00405fbb      e840ffffff     call init_crypto_keys        ;[6]
|       |    0x00405fc0      e87bf8ffff     call check_kb_layouts        ;[7]
|       |    0x00405fc5      85c0           test eax, eax
|       |,=< 0x00405fc7      743c           je 0x406005                  ;[8]
|       ||   0x00405fc9      56             push esi
|       ||   0x00405fca      6a02           push 2
|       ||   0x00405fcc      e8cf160000     call fingerprint_location    ;[9]; floating_p
|       ||   0x00405fd1      8b35b8c04000   mov esi, dword [sym.imp.KERNEL32.dll_Sleep]
|       ||   0x00405fd7      83c404         add esp, 4
|       ||   0x00405fda      68e0930400     push 0x493e0                 ; DWORD nSize
|       ||   0x00405fdf      ffd6           call esi                     ; LPDWORD lpThrea
|       ||   0x00405fe1      6a02           push 2
|       ||   0x00405fe3      e8b8160000     call fingerprint_location    ;[?]; floating_p
|       ||   0x00405fe8      83c404         add esp, 4
|       ||   0x00405feb      68c0270900     push 0x927c0
```

Les Techniques d'Analyses – Dynamique

université de BORDEAUX

- Mise en place de machines virtuelles (VBox, VmWare)
- Simulation de réseaux (INetSim)
- Analyse de trafic réseaux (Wireshark).

Les Techniques d'Analyses – Dynamique

université de BORDEAUX

- Étude des processus actifs
- Étude des appels systèmes
  - strace
  - Process Monitor / sysmon
- Débugueur
  - gdb
  - x64dbg

Plan

BillGates Botnet – Miscellaneous

université de BORDEAUX

- Première étude en Février 2014,
  aussi connu sous le nom de botnet Billgates

- Écrit en C++, 32bits, non packé ou strippé :

```
ganiw:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
        statically linked, for GNU/Linux 2.2.5, not stripped
```

BillGates Botnet – Miscellaneous

université de BORDEAUX

- Première étude en Février 2014,
  aussi connu sous le nom de botnet Billgates

- Écrit en C++, 32bits, non packé ou strippé :

```
ganiw:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
        statically linked, for GNU/Linux 2.2.5, not stripped
```

- Vise les systèmes Linux mais a été porté sur Windows un peu plus tard

BillGates Botnet – Miscellaneous

université de BORDEAUX

- Première étude en Février 2014,
  aussi connu sous le nom de botnet Billgates

- Écrit en C++, 32bits, non packé ou strippé :

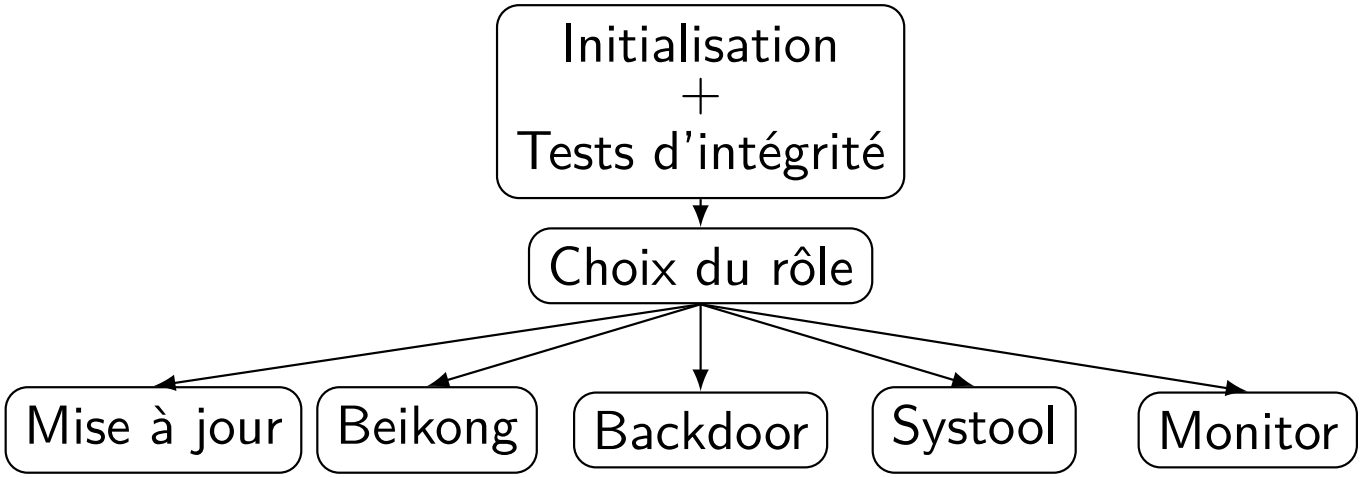
```
ganiw:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
        statically linked, for GNU/Linux 2.2.5, not stripped
```

- Vise les systèmes Linux mais a été porté sur Windows un peu plus tard

- Amplement utilisé afin de générer des attaques DDOS

BillGates Botnet – Main

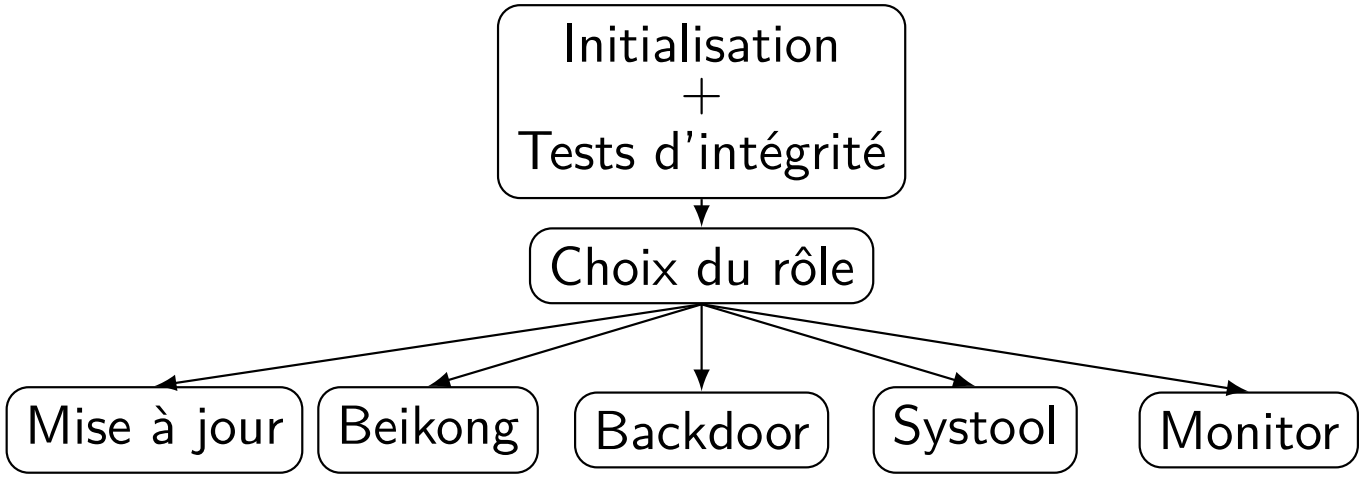
- Main :



BillGates Botnet – Main

université de BORDEAUX

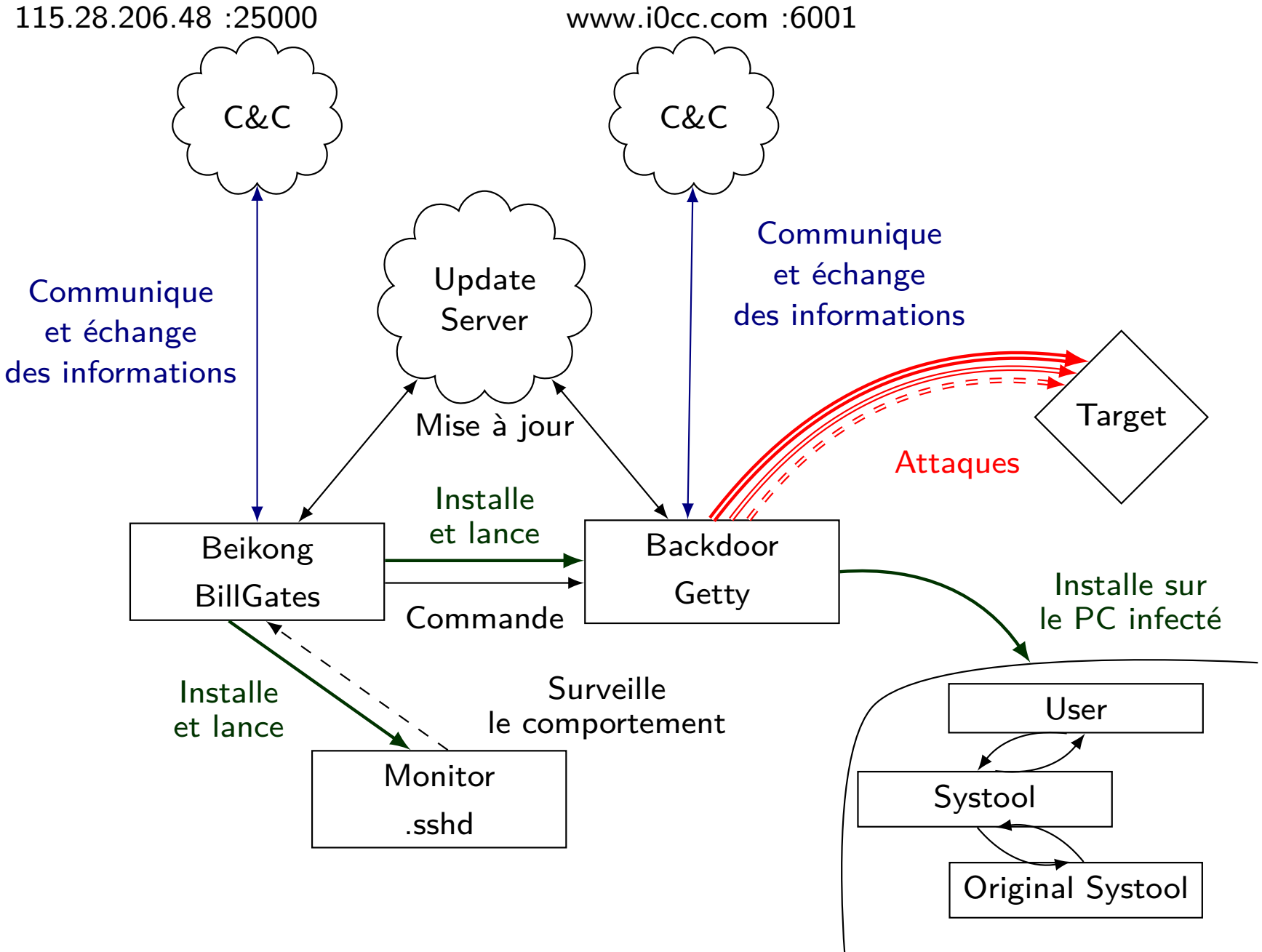
- Main :



- Légère obfuscation des données :
  "681A1C1543072E0140491F162F0B55545C55775F55565E57745E5D545652705D
  5E55585F70585C5659577D5C5F565C0423575B025A51720A56".

g_lud77	681A1C154 [...] 0423575B025A51720A56		
g_strMonitorFile	/usr/bin/.sshd	g_uHiddenPt	30000
g_iFileSize	1223123	g_iHardStart	772124
g_iSoftStart	773152	g_strDoFun	3010ad84e645e9

BillGates Botnet – Modules



BillGates Botnet – Attaques

Attaques "normales" :

- TCP flood avec header TCP choisi (fragment attack/Tear-drop) ;
- TCP flood (500 octets par connection) ;
- UDP packet flood ;
- DNS flood (sous-domaine DNS) ;
- DNS amplification ;
- Indeterminé, mais utilisant DNS ;
- ICMP-Request flood ;
- HTTP request flood, partiellement implémentée ;
- 2 attaques non implémentées.

Attaques depuis le noyaux à l'aide de l'outil *Pktgen*.

Plan

SageCrypt Ransomware – Origine

Les influences notables :

- CryLocker
  - Géolocalisation
  - Communication C&C
  - Persistance
  - Shadow Copy
  - Note de rançon
- Petya
  - Choix des algorithmes de chiffrement

SageCrypt Ransomware – Fonctionnement

Mais comment se propagent les ransomwares ?

SageCrypt Ransomware – Fonctionnement

université de BORDEAUX

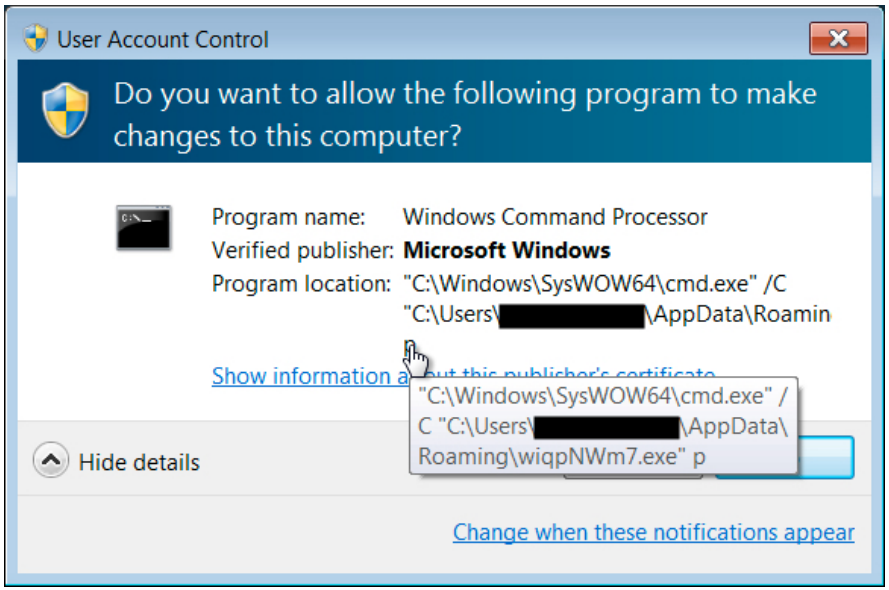
Plus précisement ...

Méthodes d'infection classiques

- Emails
  - Macros Microsoft Office
  - Exécutable déguisé
- Exploit kits
  - Adobe Flash Player
  - Oracle Java
  - Apple Quicktime
  - Mozilla Firefox

User Account Control (UAC)

Sage demande les droits administrateurs pour commencer le chiffrement et fait afficher une fenêtre UAC tant que l'utilisateur n'accepte pas la demande :



Nationalités épargnées

Sage évite d'attaquer les victimes qui sont de certaines nationalités.
Il fait cela en regardant la liste des layouts clavier utilisés par la victime.

En l'occurence, Sage épargne les nationalités suivantes :

- Biélorusse
- Kazakh
- Russe
- Ukrainien
- Ouzbek
- Sakha

SageCrypt Ransomware – Fonctionnement

université de BORDEAUX

Choix des fichiers

Sage chiffre uniquement les fichiers dont les noms utilisent les extensions faisant partie de sa liste blanche regroupant 667 extensions.
Quelques extensions présentes dans cette liste :

```
.mov .wmv .mpg .mpeg .avi .mp4 .rar .zip .tgz .7z
.tar .vmx .vmdk .vdi .qcow .ini .db .cpp .html
.gif .gpg .key
```

Truc chiant

```
0040C760    51          push ecx
0040C761    52          push edx
0040C762    8B 54 24    mov edx,dword ptr ss:[esp+C]        delta
0040C766    8B 4C 24    mov ecx,dword ptr ss:[esp+8]        saved_eip
0040C76A    81 C1 FF    add ecx,FF
0040C770    29 D1       sub ecx,edx
0040C772    41          inc ecx
0040C773    41          inc ecx
0040C774    89 4C 24    mov dword ptr ss:[esp+8],ecx
0040C778    5A          pop edx
0040C779    59          pop ecx
0040C77A    C2 04 00    ret 4
```

Figure: Fonction simulant un jmp

```
00416BF4    01 D7       add edi,edx
00416BF6    8D B3 3B    lea esi,dword ptr ds:[ebx+553B]
00416BFC    50          push eax
00416BFD    03 3D A8    add edi,dword ptr ds:[401AA8]
00416C03    68 77 8E    push 8E77
00416C08    E8 17 DD    call sage.404924
```

Figure: Exemple d'utilisation de ces fonctions

SageCrypt Ransomware – Obfuscation

Sage déchiffre la partie principale de son code pendant son exécution

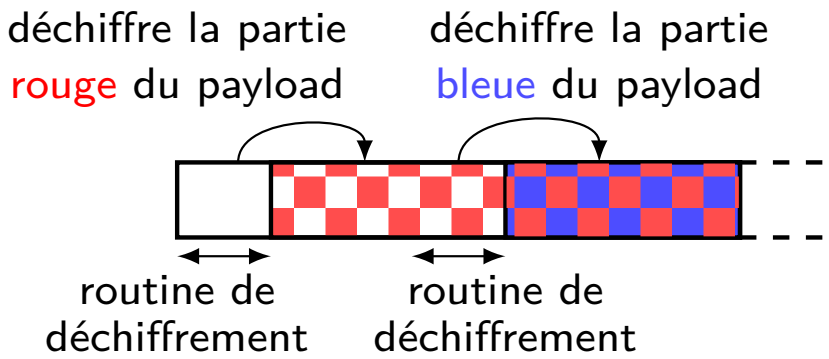


Figure: Résumé du processus d'unpacking

SageCrypt Ransomware – Chiffrement

université de BORDEAUX

Choix cryptographiques

Algorithmes utilisés pour le chiffrement :

- Diffie-Hellman basé sur les courbes elliptiques
  - Curve25519
  - 128 bits de sécurité
  - Génération simple de clés privées
- ChaCha
  - Chiffrement à flot
  - Variante de Salsa20
  - Taille de clés : 256 bits

SageCrypt Ransomware – Chiffrement

université de BORDEAUX

Notations :

- Curve25519 : Courbe d'équation $y^2 = x^3 + 486662 * x^2 + x$ sur $\mathbb{F}_p = GF(2^{255} - 19)$
- G : Point générateur de la courbe vérifiant $x(G) = 9$

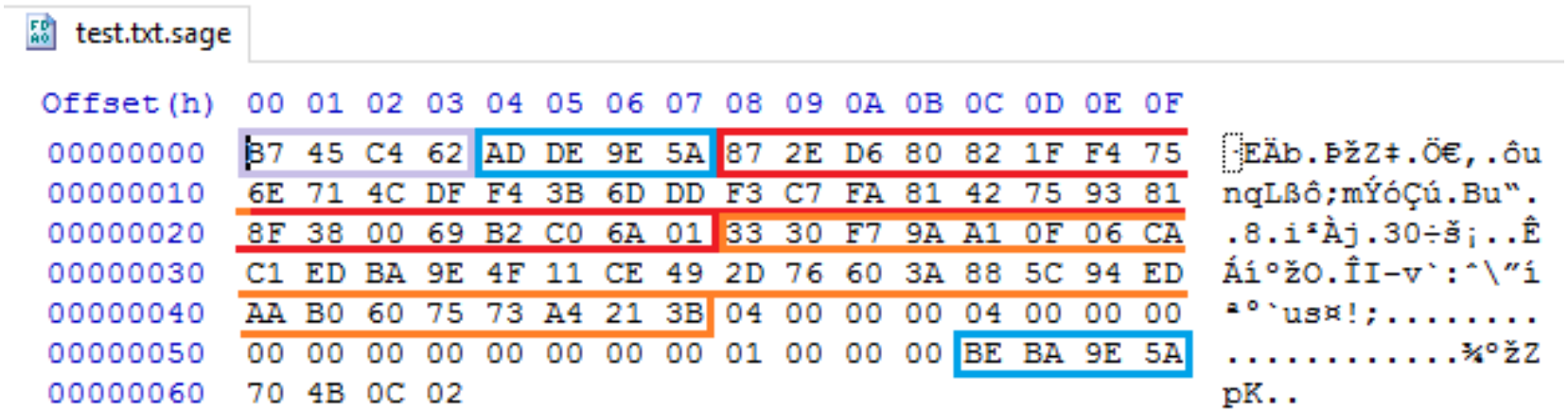
Chiffrement d'un fichier :

- On génère $k_c \in \mathbb{F}_p$ aléatoirement
- On calcule le point $Q_c = k_c * G$
- On calcule le secret partagé $S = k_c * Q_s = (k_c * k_s) * G$
- On dérive du secret partagé, un entier $sh \in \mathbb{F}_p$
- On calcule le point $P = sh * G$
- On génère $n \in \mathbb{F}_p$ aléatoirement
- On calcule les points $chacha\_key = n * P = (n * sh) * G$ et $chacha\_pub = n * G$
- On chiffre le fichier en utilisant ChaCha avec une clé dérivée de *chacha_key*
- On ajoute les valeurs de Q_c et *chacha_pub* à la fin du fichier

SageCrypt Ransomware – Chiffrement

États des fichiers après exécution de Sage

En pratique, Sage génère des fichiers qui ont la structure suivante :



Légende :

- Mauve : Contenu chiffré du fichier original
- Rouge : Q_c
- Orange : *chacha_pub*
- Bleu : Valeurs constantes

Conclusion

Conclusion

- Les malwares sont de structures et de caractéristiques très variés ;

Conclusion

- Les malwares sont de structures et de caractéristiques très variés ;
- Possible automatisation de certaines tâches, par le biais de sandbox comme *Cuckoo* ou *Limon* ;

Conclusion

- Les malwares sont de structures et de caractéristiques très variés ;
- Possible automatisation de certaines tâches, par le biais de sandbox comme *Cuckoo* ou *Limon* ;
- Nouvelles formes de classification des malwares (ex : CFG) ;

Conclusion

- Les malwares sont de structures et de caractéristiques très variés ;
- Possible automatisation de certaines tâches, par le biais de sandbox comme *Cuckoo* ou *Limon* ;
- Nouvelles formes de classification des malwares (ex : CFG) ;
- Mais automatisation totale impossible car la reconnaissance de malware est un problème indécidable, réductible au problème de l'arrêt.

Questions ?

Rapport et slides de présentation disponibles à l'adresse :
https ://github.com/pouwapouwa/Malware-Analysis-Project