Forensics / Chapardeur de mots de passes

by bak, dans le cadre du France CyberSecurity Challenge.

Problème

Énoncé

> Un ami vous demande de l'aide pour déterminer si l'email qu'il vient d'ouvrir au sujet du Covid-19 était malveillant et si c'était le cas, ce qu'il risque.
>
> Il prétend avoir essayé d'ouvrir le fichier joint à cet mail sans y parvenir. Peu de temps après, une fenêtre liée à l'anti-virus a indiqué, entre autre, le mot KPOT v2.0 mais rien d'apparent n'est arrivé en dehors de cela.
>
> Après une analyse préliminaire, votre ami vous informe qu'il est probable que ce malware ait été légèrement modifié, étant donné que le contenu potentiellement exfiltré (des parties du format de texte et de fichier avant chiffrement) ne semble plus prédictible. Il vous recommande donc de chercher d'autres éléments pour parvenir à l'aider.
>
> Vous disposez d'une capture réseau de son trafic pour l'aider à déterminer si des données ont bien été volées et lui dire s'il doit rapidement changer ses mots de passe !

L'énoncé évoque le virus **KPOT**, il pourrait être intéressant d'étudier le fonctionnement de ce virus avant de commencer l'investigation. Toutefois, il est annoncé que celui a été modifié par rapport à sa version originale.

Analyse de KPOT

Un article de proofpoint décrit très bien le fonctionnement de ce virus. On y apprend que le virus exfiltre des données de la cible et qu'il communique avec son centre de commandes via une URL se terminant par `/gate.php`.

On repère également le nom de domaine `bendes.co.uk`, utilisé par un centre de commandes.

Le fonctionnement du virus est le suivant :

1. l'agent contacte le centre de commandes via une requête HTTP GET.
2. le centre de commandes répond en indiquant les informations qu'il souhaite obtenir à propos de la victime. Cette réponse est XORée avec une clée hardcodée, puis encodée en Base 64.
3. l'agent effectue une requête HTTP POST contenant les informations demandées par le centre de commandes, XORées avec la même clé que précédemment.

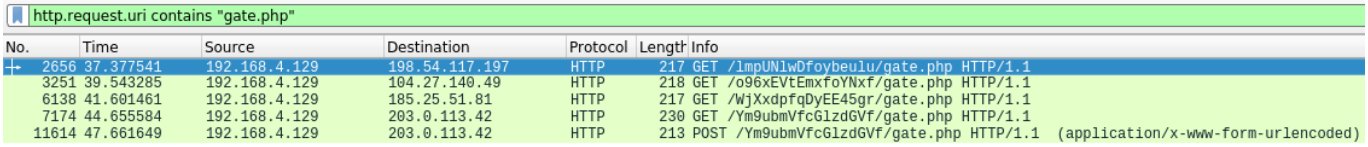
L'article montre un exemple de données telles qu'envoyées par le centre de commandes, avant chiffrement :

```
1111111111111100__DELIMM__A.B.C.D__DELIMM__appdata__GRABBER__*.log,*.txt,__
GRABBER__%appdata%__GRABBER__0__GRABBER__1024__DELIMM__desktop_txt__GRABBER
__*.txt,__GRABBER__%userprofile%\Desktop__GRABBER__0__GRABBER__150__DELIMM_
___DELIMM____DELIMM__
```

Obtention des informations chiffrées

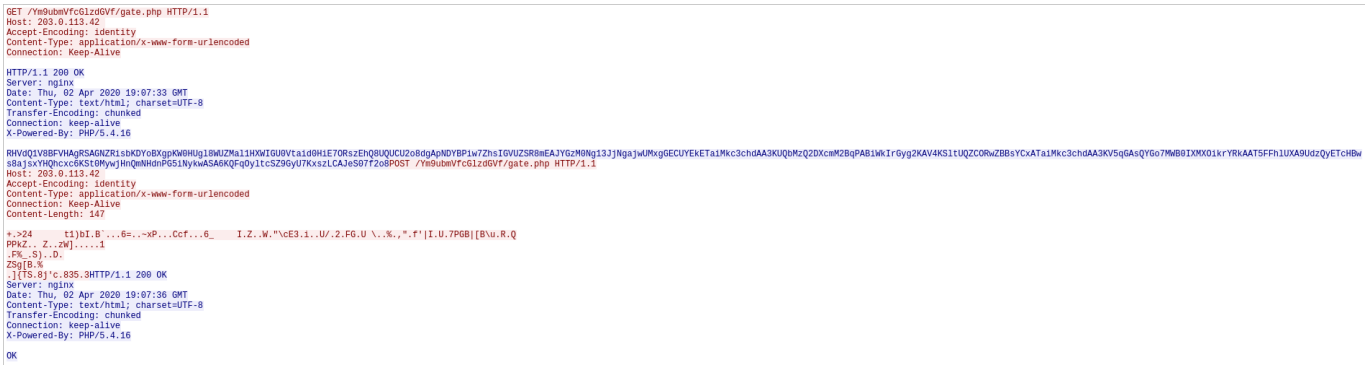
La capture réseau fournie avec le challenge peut être analysée à l'aide de wireshark. Cet outil possède un système de filtrage assez puissant.

En filtrant uniquement les requêtes HTTP dont les URLs contiennent `gate.php`, le résultat suivant est obtenu :



Quatre centres de commandes sont contactés, mais seul le dernier semble répondre car on peut distinguer une requête HTTP POST à destination du serveur `203.0.113.42`, présageant d'une réponse antérieure de ce même serveur.

La totalité de l'échange HTTP peut être observée en utilisant l'option `Follow HTTP stream` :



Il est possible d'extraire les corps des deux requêtes, toujours grâce à wireshark, afin de tenter de déchiffrer leur contenu.

Déchiffrement du contenu des requêtes

Obtention de la clé de XOR

Le corps de la première requête va nous permettre de retrouver la clé utilisée pour le XOR, car nous connaissons une partie du clair de ce message, grâce à l'exemple fourni par l'article.

```
$ echo -ne
"RHVdQ1V8BFVHAgRSAGNZRisbKDYoBXgpKW0HUgl8WUZMal1HXWIGU0Vtaid0HiE7ORszEhQ8UQ
UCU2o8dgApNDYBPiw7ZhsIGVUZSR8mEAJYGzM0Ng13JjNgajwUMxgGECUYEkETaiMkc3chdAA3K
UQbMzQ2DXcmM2BqPABiWkIrGyg2KAV4KSltUQZCORwZBBsYCxATaiMkc3chdAA3KV5qGAsQYGo7
MWB0IXMXOikrYRkAAT5FFhlUXA9UdzQyETcHBws8ajsxYHQhcxc6KSt0MywjHnQmNHdnPG5iNyk
wASA6KQFqOyltcSZ9GyU7KxszLCAJeS07f2o8" | base64 -d > get_body.raw
```

Pour ce faire, le script xorknown a été utilisé :

```
$ python ~/src/scripts/xorknown.py get_body.raw "__GRABBER__%user" 20
Searching XOR-encrypted get_body.raw for string '__GRABBER__%user'
(max_key_length = 20)
Key length: 16
Partial Key: tDlsdL5dv25c1Rhv
Plaintext:
0110101110111110__DELIMM__218.108.149.373__DELIMM__appdata__GRABBER__*.log,
*.txt,__Gerprofile%\Desktop__GRABBER__0__GRABBER__0__DELIMM____DELIMM____DE
LIMM__
```

En spécifiant une partie du texte clair, nous pouvons donc retrouver la clé utilisée : `tDlsdL5dv25c1Rhv`.

Déchiffrement de la requête HTTP POST

Il est recommandé d'extraire le corps de la requête POST sous forme hexadécimale (Option `Copy as a Hex stream`).

Maintenant en possession de la clé utilisée pour XORer les données échangées, nous pouvons déchiffrer le corps de la seconde requête :

```
$ echo -ne "tDlsdL5dv25c1Rhv" | xxd -p
74446c73644c35647632356331526876
$ python3
Python 3.7.3 (default, Dec 20 2019, 18:57:59)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> key = bytearray.fromhex("74446c73644c35647632356331526876")
>>> flag =
bytearray.fromhex("2b003e3234097431296249164260181301363d06017e78501a131543
63661b0501365f09491a5a10045718225c63453300691a1c552f04321946470655205c06112
5192c220f66277c49015508375047427c5b425c750c5213510d50506b5a1717205a15017a57
5d1502060007310d1246255f125329020544020d5a53675b4216250d165d7b54530b386a276
3133833351133")
>>> ''.join(chr(x^y) for (x,y) in zip(flag,cycle(key)))
'_DRAPEAU_P|us2peurQue2M4l!  R4ssur3z-Votre-Am1-Et-
vo1c1Votredr4peau_FCSC\n{469e8168718996ec83a92acd6fe6b9c03c6ced2a3a7e7a2089
b534baae97a7}\n_DRAPEAU_'
```

Flag : `FCSC{469e8168718996ec83a92acd6fe6b9c03c6ced2a3a7e7a2089b534baae97a7}`.

Il semblerait au final que le le virus n'ait pas été tant modifié que ça par rapport à sa version originale.