AMM - Laboratoire 7

Analyse du malware Jackal

Maxime Chantemargue, Ruben Pereira Lopes, Charles Matrand

5 mai 2024

Table des matières

Résumé

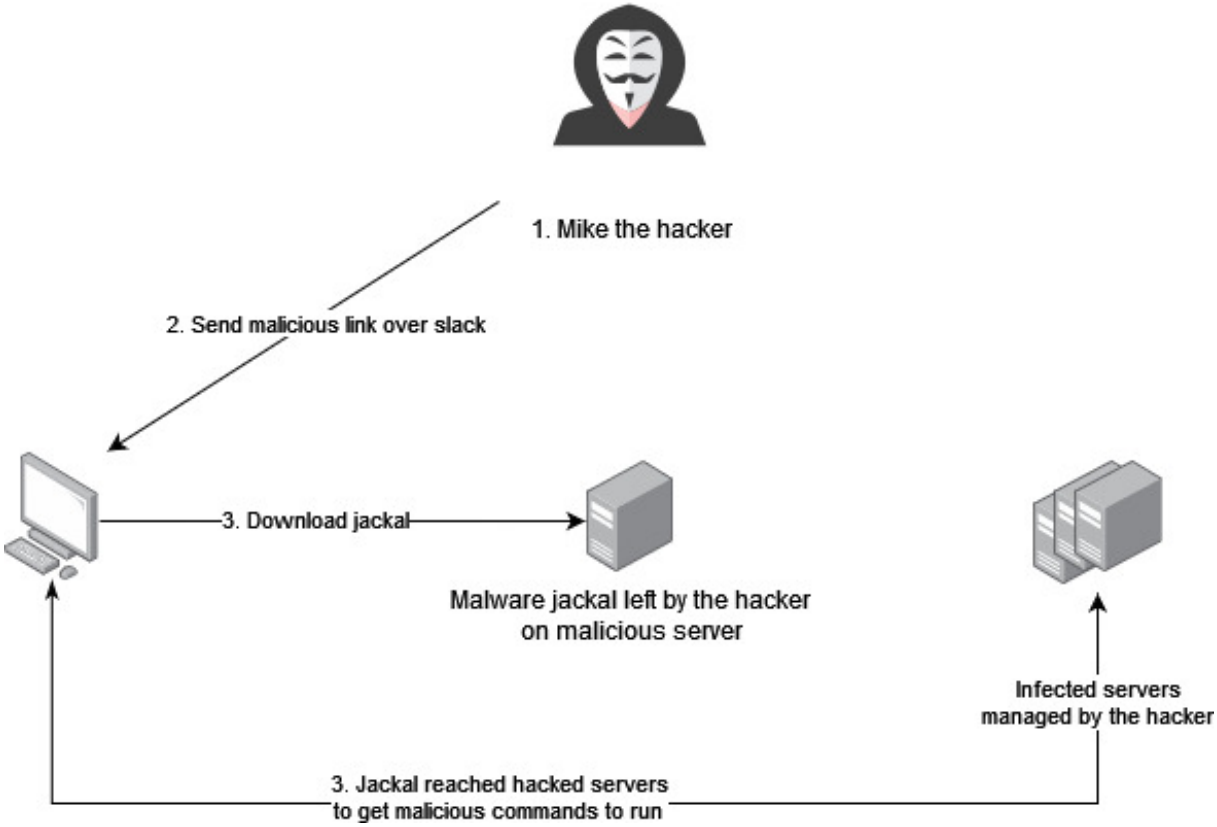
Un échantillon du malware Jackal a été analysé, provenant d'une image mémoire et d'une capture des communications entre l'appareil affecté et un serveur externe. L'objectif de cette analyse était de comprendre les fonctionnements et les méthodes d'infection du malware.

Il a été découvert que la victime avait reçu un lien vers le malware via l'application Slack, qu'elle a ensuite téléchargé et exécuté, résultant en l'infection de l'appareil.

Le malware a par la suite établi une connexion avec des serveurs contrôlés par l'attaquant pour recevoir des commandes à exécuter. Des artefacts ont également été identifiés, lesquels pourront aider à détecter si un appareil est infecté et à prévenir cette infection dans certains cas.

Chronologie d'infection

En analysant plus en détail le dump mémoire, il est possible d'accéder à une base de données de cookie Firefox montrant que la victime a installé slack et que depuis ici, le malware a été transmis sur la machine de la victime. Le lien malveillant venant de slack et menant directement à `jackal.exe` a été envoyé par "mike".



Analyse

Analyse de l'exécutable

Tout d'abord, il est important d'effectuer un peframe pour en savoir plus sur l'exécutable infecté :

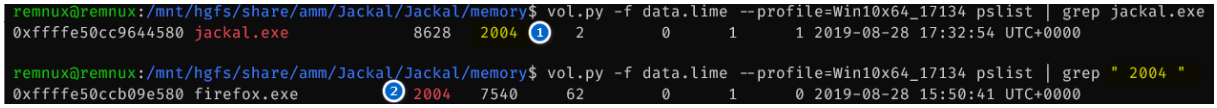
```
remnux@remnux:/mnt/hgfs/share/amm/Jackal/Jackal/INFECTED/jackal.exe$ peframe -i jackal.exe
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)


---------------------------------------------------------------------------
File Information (time: 0:00:00.734023)
---------------------------------------------------------------------------
filename          jackal.exe
filetype          PE32 executable (console) Intel 80386, for MS Windows, UPX comp
filesize          21504
hash sha256       9be951e80f3c2ad49b2d53464c924bb4ab8c81f2d771d46c17afad258aca8fc9
virustotal        /
imagebase         0x400000
entrypoint        0x10840
imphash           bde8f5b22ebc6a808b3c477a73c3d9ac
datetime          2013-03-08 05:29:51
dll               False
directories       import, tls, resources, relocations
sections          UPX0, .rsrc, UPX1 *
features          packer


---------------------------------------------------------------------------
Interactive mode (press TAB to show commands)
---------------------------------------------------------------------------
[peframe]>
```

Il est important de noter qu'à première vue, le malware est packé, ce qui peut le rendre très compliqué à analyser.

Il est donc plus simple de directement aller analyser la mémoire, pour voir s'il n'a pas été dépacké et directement interpréter des actions malveillantes ou suspectes. Il va falloir retrouver son PID et le dumper depuis la dump mémoire.

Liste des processus

En utilisant pslist, il est possible de voir les différents processus chargés en mémoire, il est possible de trouvé jackal.exe dont le parent semble être firefox.exe. Ce qui montre clairement que la victime a lancé l'exécutable malveillant par le biais de firefox.exe après l'avoir téléchargé.



Dump exécutable

Maintenant que nous avons son PID, nous pouvons dumper l'exécutable avec vol.pyprocdump -f data.lime --profile=Win10x64_17134 -D process_dump --memory -p 8628 , le --memory permet d'obtenir plus d'informations sur l'exécutable (slack space, …).

Mais pourquoi dump l'exécutable via le dump mémoire et pas juste récupérer l'exécutable infecté directement (dans le dossier du laboratoire) ? Car il est packé, donc contient des données non exploitable pour notre analyse.

Slack space

Il est important de noter une différence avec ou sans le `--memory`. Avec, il inclut la slack space contenue entre les PE sections non alignées (page). Si nous faisons un `strings -a` sur l'exécutable sans la slack space, nous remarquons qu'il manque les valeurs en base64 (C2 IP). Nous pouvons alors en déduire que ces valeurs sont stockés dans les slack space.

Valeurs en Base64

En analysant son contenu avec la commande `strings -a -el executable.8628_mem.exe`, il y une partie qui nous intéresse :

```
JyM9IiM9ISAqPSEiKzx5emBnPXlg
ISElPSEjIj0iJCA9JCE8e3x8d2R6fXg8RXJ9cHxmZXZhPGV6YHpnPGBmfn52YT17Z35/
ISMjPSInJz0mIz0hIiE8YXZgcn5jf3Z3PHF2dXxhdj17Z35/
ISEiPSIlJj0iJSc9JiU8Y2Z/f3FycHg8anZge3plcns9Z2tn
JyY9IiYiPSIrID0iJyo8eHJge3ZhPEt2YXxren10PXlg
ISEkPSEhPSImJD0mPGZgcjxgfHBwdmE9e2d+fw==
ISIrPSIiID0iJCs9JCI8ZHZyZ3t2YT17Z35/
IiQrPSsmPSIqIj0mITx2dmNhfH48YHZldmF2PXtnfn8=
IiMiPSogPSohPSIlJDx8YGd2fWB8YWo8ZHpweHZ3PHJ/fw==
IiA9KyE9IiYiPSEiJjx/fGNjdmE8d2Z+cXF2f389Z2tn
KiM9ICY9ISAnPSEnKzx8YXp2fWc8cXJ4dmFgPXtnfn8=
IiMmPSQiPSEgJD0iJTxgZmN2YTxwf3JxcXZhYD1rfn8=
ICE9ISc9IScrPSIkKzx0fHx3emB7PHV2f398ZD17Z35/
IiMrPSIiJj0qKj0rJTxqdn9/fGR7cn5+dmE8YXZjfGFnYA==
IiErPSIgJz0iJCU9IiElPHZremBnYDxDcmByd3Z9cj13fHA=
JSQ9KyY9IScrPSEmPGRyenVgPHlmfXw9e2d+fw==
ISMhPSIqPSIqJD0iICI8fHBnfHF2YTxgcmdmYX09eWA=
IiYqPSEgIT0iIyE9IiYrPGNyYGB6fXQ8YHxmYXB2PHt8fnY9cmBj
IiUlPSEjPSYgPSEiKjxRZmFxcn14PF9mcHp3PHlycHhgPXtnfn8=
IiclPSshPSQkPSYqPGAqIioiIjx4f2B5ciIiPHV6f2d2YT1na2c=
ISAgPSIkIT0iKyM9IiUgPGFyfXB7dmE8ZHp9d3xkYD1je2M=
ISYhPSEhKj0iKiA9ISEkPGBnfHB4PGdhcnd6fXQ9e2d+fw==
ISIhPScgPSInIz0iJiE8YGN8YWc8e3Jje3JpcmF3PXlg
IisnPScrPSInID0iIiQ8ZHJ6Z3p9dD1yYGM=
ISYgPSEhJz0iJCI9ICo8fXZkYDxwfH5+PWN7Yw==
Jic9IiAqPSIrIz0iICs8YXxyYWA8ZXJhenxmdzxyf2d2YX1yZ3p8fT17Z35/
IiUrPSsnPSIqKz0hJys8YHZ2fTxqdn9jPXtnfn8=
IismPSEhKj0iJiQ9IiUrPHx9YXZwanZ7fDxldmFxcn89Z2tn
ISEqPSY9ISAgPSEjJDxndnpqajxgY39meD1yYGM=
IionPSEjKj0rKj0nIjxgdmF6fGA8e3J/cXZhPXtnfn8=
```

Cela représente du texte au format "base 64", si nous le décodons, nous aurons des valeurs illisibles. Cependant dans la donnée du laboratoire, il y a le terme "jackal's c2 list is just base64 and xor" qui revient, ce qui veut dire qu'il y a encore une étape de "déchiffrement" sur nos textes. Donc il faut

trouver encore une clé qui permet de déchiffrer nos textes. Pour cela, il faut utiliser l'outil "CyberChef", voici le lien avec nos étapes d'analyse : CyberChef Analysis.

Avec cet outil, la clé retrouvée est "0x13", qui teste toutes les possibilités de clés à une taille donnée et affiche le contenu du texte déchiffré (0x13 étant la plus reconnaissable). Le texte déchiffré est la liste des C2 suivante :

```
40.10.239.218/jist.js
226.201.173.72/hoodwink/Vancouver/visit/summer.html
200.144.50.212/resampled/before.html
221.165.164.56/pullback/yeshivah.txt
45.151.183.149/kasher/Xeroxing.js
227.22.157.5/usa/soccer.html
218.113.178.71/weather.html
178.85.191.52/eeprom/severe.html
101.93.92.167/ostensory/wicked/all
13.82.151.215/lopper/dumbbell.txt
90.35.234.248/orient/bakers.html
105.71.237.16/super/clabbers.xml
32.24.248.178/goodish/fellow.html
108.115.99.86/yellowhammer/reports
**128.134.176.126/exists/Pasadena.doc** => revient dans le netscan
67.85.248.25/waifs/juno.html
202.19.197.131/october/saturn.js
159.232.102.158/passing/source/home.asp
166.20.53.219/Burbank/Lucid/jacks.html
146.82.77.59/s91911/klsja11/filter.txt
233.172.180.163/rancher/windows.php
252.229.193.227/stock/trading.html
212.43.140.152/sport/haphazard.js
184.48.143.117/waiting.asp
253.224.171.39/news/comm.php
54.139.180.138/roars/varioud/alternation.html
168.84.198.248/seen/yelp.html
185.229.157.168/onrecyeho/verbal.txt
229.5.233.207/teiyy/spluk.asp
194.209.89.41/serios/halber.html
```

Le but de les chiffrer est d'éviter que quelqu'un ne retrouve les IPs des C2 (dans ce cas c'est un simple XOR, donc facilement déchiffrable). Mais probablement pour cacher leurs contenus en cas d'analyse d'antivirus par exemple, pour éviter d'être détectées.

Clés de registre

Dans le résultat précédent, nous retrouvons une clé de registre `Software\Microsoft\Windows Player`. Si nous allons regarder son contenu avec `vol.py -f data.lime --profile=Win10x64_17134 printkey -K "\SOFTWARE\MICROSOFT\WINDOWS PLAYER"`:

```
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \??\C:\Users\Analyst\ntuser.dat
Key name: Windows Player (S)
Last updated: 2019-08-28 17:32:55 UTC+0000

Subkeys:

Values:
REG_SZ        DB1L            : (S) IionPSEjKj0rKj0nIjxgdmF6fGA8e3J/cXZhPXtnfn8=
REG_SZ        WN33            : (S) ISEqPSY9ISAgPSEjJDxndnpqajxgY39meD1yYGM=
REG_SZ        4H2N            : (S) IismPSEhKj0iJiQ9IiUrPHx9YXZwanZ7fDxldmFxcn89Z2tn
REG_SZ        MRRU            : (S) IiUrPSsnPSIqKz0hJys8YHZ2fTxqdn9jPXtnfn8=
REG_SZ        HNFY            : (S) Jic9IiAqPSIrIz0iICs8YXxyYWA8ZXJhenxmdzxyf2d2YX1yZ3p8fT17Z35/
REG_SZ        IEUH            : (S) ISYgPSEhJz0iJCI9ICo8fXZkYDxwfH5+PWN7Yw==
REG_SZ        1AUR            : (S) IisnPScrPSInID0iIiQ8ZHJ6Z3p9dD1yYGM=
REG_SZ        47SG            : (S) ISIhPScgPSInIz0iJiE8YGN8YWc8e3Jje3JpcmF3PXlg
REG_SZ        FAU1            : (S) ISYhPSEhKj0iKiA9ISEkPGBnfHB4PGdhcnd6fXQ9e2d+fw==
REG_SZ        2LHL            : (S) ISAgPSIkIT0iKyM9IiUgPGFyfXB7dmE8ZHp9d3xkYD1je2M=
REG_SZ        5WYY            : (S) IiclPSshPSQkPSYqPGAqIioiIjx4f2B5ciIiPHV6f2d2YT1na2c=
REG_SZ        KYKG            : (S) IiUlPSEjPSYgPSEiKjxRZmFxcn14PF9mcHp3PHlycHhgPXtnfn8=
REG_SZ        Q810            : (S) IiYqPSEgIT0iIyE9IiYrPGNyYGB6fXQ8YHxmYXB2PHt8fnY9cmBj
REG_SZ        M65P            : (S) ISMhPSIqPSIqJD0iICI8fHBnfHF2YTxgcmdmYX09eWA=
REG_SZ        0GF9            : (S) JSQ9KyY9IScrPSEmPGRyenVgPHlmfXw9e2d+fw==
REG_SZ        IYCD            : (S) IiErPSIgJz0iJCU9IiElPHZremBnYDxDcmByd3Z9cj13fHA=
REG_SZ        78OI            : (S) IiMrPSIiJj0qKj0rJTxqdn9/fGR7cn5+dmE8YXZjfGFnYA==
REG_SZ        YBTI            : (S) ICE9ISc9IScrPSIkKzx0fHx3emB7PHV2f398ZD17Z35/
REG_SZ        THRG            : (S) IiMmPSQiPSEgJD0iJTxgZmN2YTxwf3JxcXZhYD1rfn8=
REG_SZ        PXDT            : (S) KiM9ICY9ISAnPSEnKzx8YXp2fWc8cXJ4dmFgPXtnfn8=
REG_SZ        FYNO            : (S) IiA9KyE9IiYiPSEiJjx/fGNjdmE8d2Z+cXF2f389Z2tn
REG_SZ        IWT5            : (S) IiMiPSogPSohPSIlJDx8YGd2fWB8YWo8ZHpweHZ3PHJ/fw==
REG_SZ        6NLE            : (S) IiQrPSsmPSIqIj0mITx2dmNhfH48YHZldmF2PXtnfn8=
REG_SZ        XOJV            : (S) ISIrPSIiID0iJCs9JCI8ZHZyZ3t2YT17Z35/
REG_SZ        XP8X            : (S) ISEkPSEhPSImJD0mPGZgcjxgfHBwdmE9e2d+fw==
REG_SZ        3EDQ            : (S) JyY9IiYiPSIrID0iJyo8eHJge3ZhPEt2YXxren10PXlg
REG_SZ        ONON            : (S) ISEiPSIlJj0iJSc9JiU8Y2Z/f3FycHg8anZge3plcns9Z2tn
REG_SZ        ZXU6            : (S) ISMjPSInJz0mIz0hIiE8YXZgcn5jf3Z3PHF2dXxhdj17Z35/
REG_SZ        A3D7            : (S) ISElPSEjIj0iJCA9JCE8e3x8d2R6fXg8RXJ9cHxmZXZhPGV6YHpnPGBmfn52YT17Z35/
REG_SZ        FRRM            : (S) JyM9IiM9ISAqPSEiKzx5emBnPXlg
```

Le même contenu que dans l'exécutable. Pour la persistance, il stocke ces valeurs dans des clés de registres.

Privilèges

`SeDebugPrivilege` est aussi retrouvable dans le contenu de l'exécutable. Il permet de modifier les processus d'autres utilisateurs, ce qui peut entraîner une élévation de privilèges par exemple. Il est exploité par des outils comme Mimikatz pour extraire les mots de passe du processus LSASS, qui gère l'authentification sur les systèmes Windows.

Fonctionnalités

Au niveau des fonctionnalités du malware, les `dll` sont une source d'informations intéressantes. Dans le résultat du `strings -a` de l'exécutable sans slack space, les dll suivantes sont retrouvables :

- `KERNEL32.DLL`
- `ADVAPI32.dll`
- `ole32.dll`
- `urlmon.dll`
- `WININET.dll`

- `WS2_32.dll`

Le malware utilise les 3 dernières dll pour le côté réseau, communication. Ensuite, `ADVAPI32`, qui fournit une liste de fonctions qui permettent aux applications de communiquer avec les services du système. Il y a d'autres fonctionnalités suspectes retrouvables :

- Manipulation de clés de registres : `RegCloseKey`, `RegEnumValueW`, `RegQueryInfoKeyW`, `RegCreateKeyExW`
- `AdjustTokenPrivileges` : permet de modifier les privilèges d'un token de sécurité
- `LookupPrivilegeValueW` : permet de rechercher la valeur numérique d'un privilège spécifique
- `URLDownloadToFileA` : pour télécharger un fichier à partir d'une URL et l'enregistrer localement

Mutex

La commande `mutantscan` nous permet d'analyser les mutex appelés par les différents processus :

```
1 Offset(P)              #Ptr     #Hnd Signal Thread
     CID Name
2 ------------------ -------- -------- ------ ------------------
     --------- ----
3 [...]
4 **0x0000e50cc83d7ec0        2        1      0 0xffffe50cc9ff1700
     8628:5876 __Dassara__**
5 [...]
6 0x0000e50ccaa013f0    32768        1      1 0x0000000000000000
               SM0:8628:64:WilError_01
7 0x0000e50ccad25550    32768        1      1 0x0000000000000000
               SM0:8628:168:WilStaging_02
```

Il nous indique qu'il y a un lien avec un thread lancé par jackal, un mutex du nom de `__Dassara__`, retrouvable aussi dans le contenu de l'exécutable.

Analyse réseau

Avec `netscan`, permettant de scanner des artefacts réseau, nous avons les lignes intéressantes suivantes (vol.py -f data.lime –profile=Win10x64_17134 netscan):

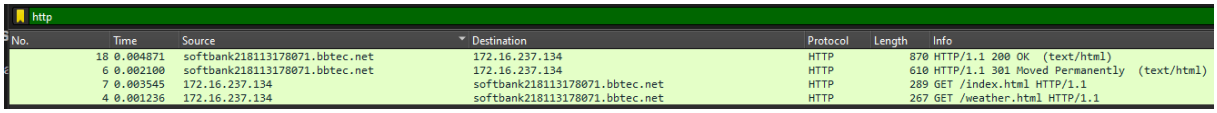
```
1 **[...]
2 0xe50cc8a79180     TCPv4    0.0.0.0:9090                    0.0.0.0:0
                LISTENING        8628     jackal.exe     2019-08-28
     18:41:54 UTC+0000**
3 [...]
4 **0xe50cca1d4010     TCPv4    192.168.231.131:52349
     128.134.176.126:80   CLOSED           -1
     6490-05-12 05:38:17 UTC+0000**
5 [...]
```

La première ligne fait référence au owner `jackal.exe` qui écoute sur le port 9090 de toutes les IP (0.0.0.0).

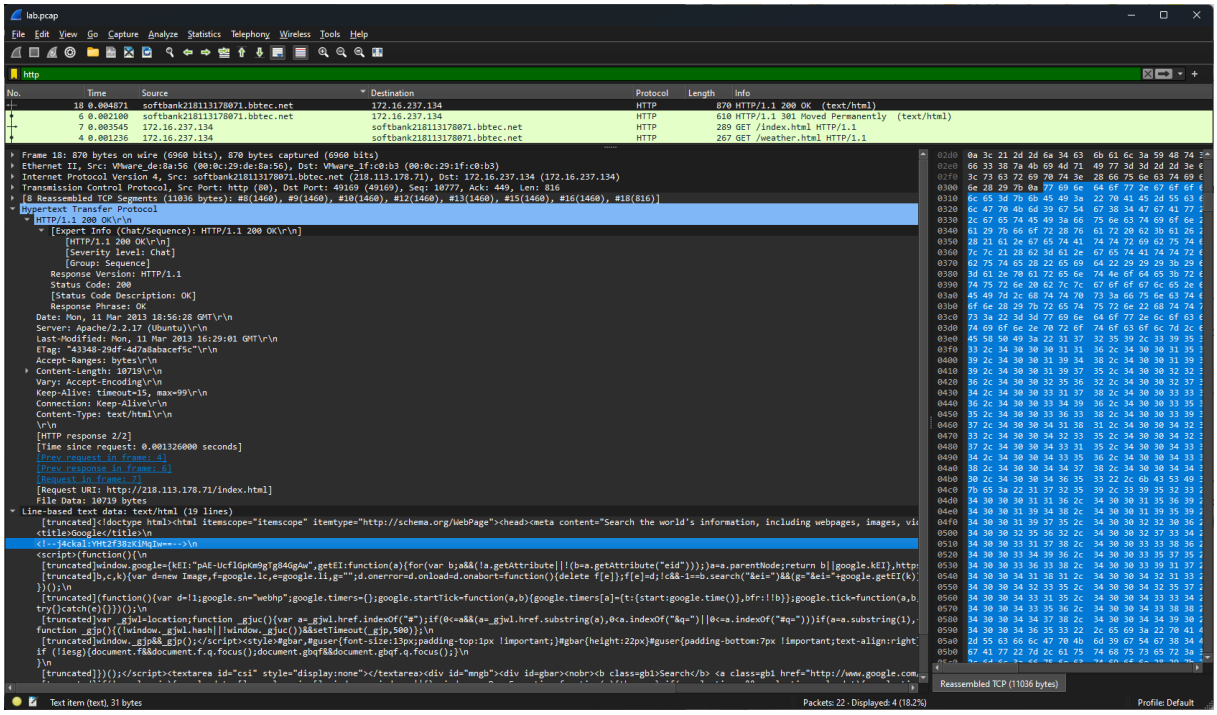
La deuxième ligne fait appel à l'une des C2 (128.134.176.126/exists/Pasadena.doc) retrouvées auparavant.

Requête sur le serveur

En analysant le pcap fourni, sur la base des données trouvées en dessus, nous filtrons directement sur la base du type de requête → http :



Nous remarquons plusieurs requêtes envoyées depuis jackal et en regardant une réponse serveur nous voyons une requête, à première vue légitime, mais contenant un commentaire en base64 :

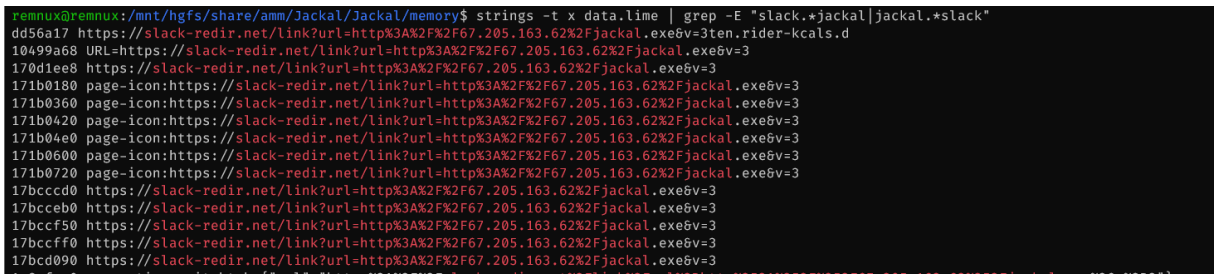


Cette chaîne base64 est chiffré avec la clé **0x13** et signifie : `shell 9090`. (ref. CyberChef Analysis 2) 9090 étant un numéro de port retrouvable dans le netscan au-dessus, écouté par Jackal.

Analyse de firefox

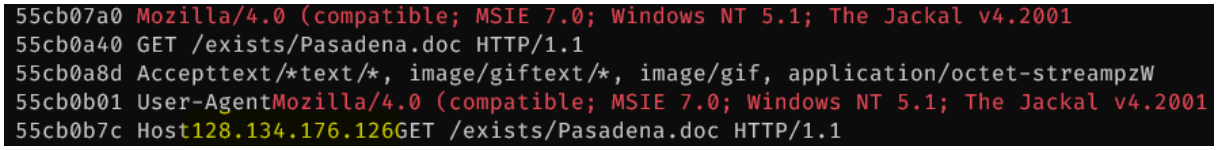
Après les analyses ci-dessus, le logiciel malveillant semble venir de `firefox.exe`. En analysant les données propres à `firefox.exe`, il est possible de récupérer un historique ainsi que des cookies.

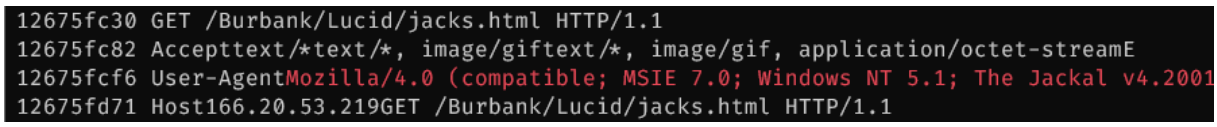
Il est possible de voir ci-dessus, des cookies propres à slack.com. Ce qui laisse penser que la victime s'est connectée sur cette plateforme de messagerie. Sachant ceci, en regardant plus en détail à l'aide de strings, il ressort clairement que `jackal.exe` a été téléchargé vient une conversation slack, car le serveur semble être un cdn lié au partage de fichier au sein d'une conversation slack.



Sachant que jackal.exe a été téléchargé par ce biais, en faisant un strings avec "jackal" et "Jackal", le résultat a affiché un usera-gent. En prenant dans le strings, exactement le user-agent de jackal, il est possible de relever les C2 ou serveurs utilisés par le hacker :

Voici les parties intéressantes :





```
--
145287f74 File
145288109 Accepttext/*text/*, image/giftext/*, image/gif, application/octet-stream
14528817d User-AgentMozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001&
1452881f8 Host229.5.233.207GET /teiyy/spluk.asp HTTP/1.1
145288252 Cache-Controlno-cache
145289144 1ndC
--
14eebe718 GET /seen/yelp.html HTTP/1.14vZ
14eebe760 Accepttext/*text/*, image/giftext/*, image/gif, application/octet-stream
14eebe7d4 User-AgentMozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001#wZ
14eebe84f Host168.84.198.248GET /seen/yelp.html HTTP/1.1}wZ
14eebe8a9 Cache-Controlno-cache/seen/yelp.html
14eebe8f9 Host67.205.163.62GET / HTTP/1.1
--
15d6900ef T$HH
15d69011d Accepttext/*text/*, image/giftext/*, image/gif, application/octet-stream
15d690191 User-AgentMozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001@!Z
15d69020c Host233.172.180.163GET /rancher/windows.php HTTP/1.1
15d69026c Cache-Controlno-cache
15d691100 T$ 3
```

Grâce à cela, nous en déduisons que `jackal.exe` fait ses propres requêtes avec son propre user-agent. Ce qui, soit dit en passant, n'est pas malin en termes de discrétion.

Plus bas, nous verrons la liste des C2 supposés du hacker, qui font bien référence à ces requêtes.

Historique web

Dans notre analyse, nous avons trouvé des résultats dans la liste de fichiers en mémoire avec vol.py`-f ../data.lime --profile=Win10x64_17134 filescan >> ../filedump.txt`. Avec cela, nous avons recherché des fichiers en lien avec Firefox premièrement avec un simple `cat filedump.txt | grep -i firefox` qui contenait une assez grande liste de fichier (surtout, cache).

Il serait intéressant de voir s'il y a l'historique des liens parcourus et de voir si le terme "jackal" revient ou non. Dans https://support.mozilla.org/en-US/questions/1176169#:~:text=Firefox stores your history and bookmarks together in,named places.sqlite which is in your profile folder, il nous indique que l'historique se trouve dans un fichier `places.sqlite` que nous retrouvons bien dans notre `grep` de tout à l'heure :

```
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\favicons.sqlite
1 R--rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\favicons.sqlite
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\places.sqlite
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\favicons.sqlite-wal
0 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Local\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\cache2\entries\337044
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\webappsstore.sqlite-shm
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvwl.default-release\webappsstore.sqlite-wal
0 RWD--- \Device\HarddiskVolume3\Program Files\Mozilla Firefox\omni.ja
```

Maintenant que nous avons cette information, il faut dump le fichier. Pour cela, nous avons utilisé la commande vol.py`-f data.lime --profile=Win10x64_17134 dumpfiles --regex "\.(sqlite)$" -D dump_files/ --name`. Nous avons choisi de sélectionner tous les ".sqlite", car possiblement il pourrait y avoir d'autres fichiers intéressants.

Pour observer le contenu du fichier "place.sqlite", nous avons utilisé l'outil "sqlitebrowser" qui permet de visualiser le contenu d'une base de données. Il faut auparavant convertir le fichier en format `sqlite` puis `sql` pour qu'il soit lisible). Voici ce que nous trouvons d'intéressant :

Table: moz_origins

	id	prefix	host	frecency
	Filter	Filter	Filter	Filter
1	1	https://	www.mozilla.org	545
2	2	https://	support.mozilla.org	280
3	3	https://	www.google.com	18000
4	4	https://	www.messenger.com	100
5	5	https://	www.facebook.com	100
6	6	https://	www.microsoft.com	325
7	7	https://	www.googleadservices.com	25
8	8	https://	pixel.everesttech.net	25
9	9	https://	www.adobe.com	240
10	10	https://	weechat.org	500
11	11	https://	www.wechat.com	100
12	12	http://	weixin.qq.com	25
13	13	https://	pc.weixin.qq.com	100
14	14	https://	slack.com	900
15	15	https://	downloads.slack-edge.com	0
16	16	https://	acmemarketing123-talk.slack.com	25
17	17	https://	app.slack.com	200
18	18	https://	slack-redir.net	100
19	19	http://	67.205.163.62	0
20	20	http://	www.google.com	2000

Premièrement, dans la table “moz_origins”, il y a une IP, ce qui est suspect pour un lien. Si nous regardons plus loin dans les tables :

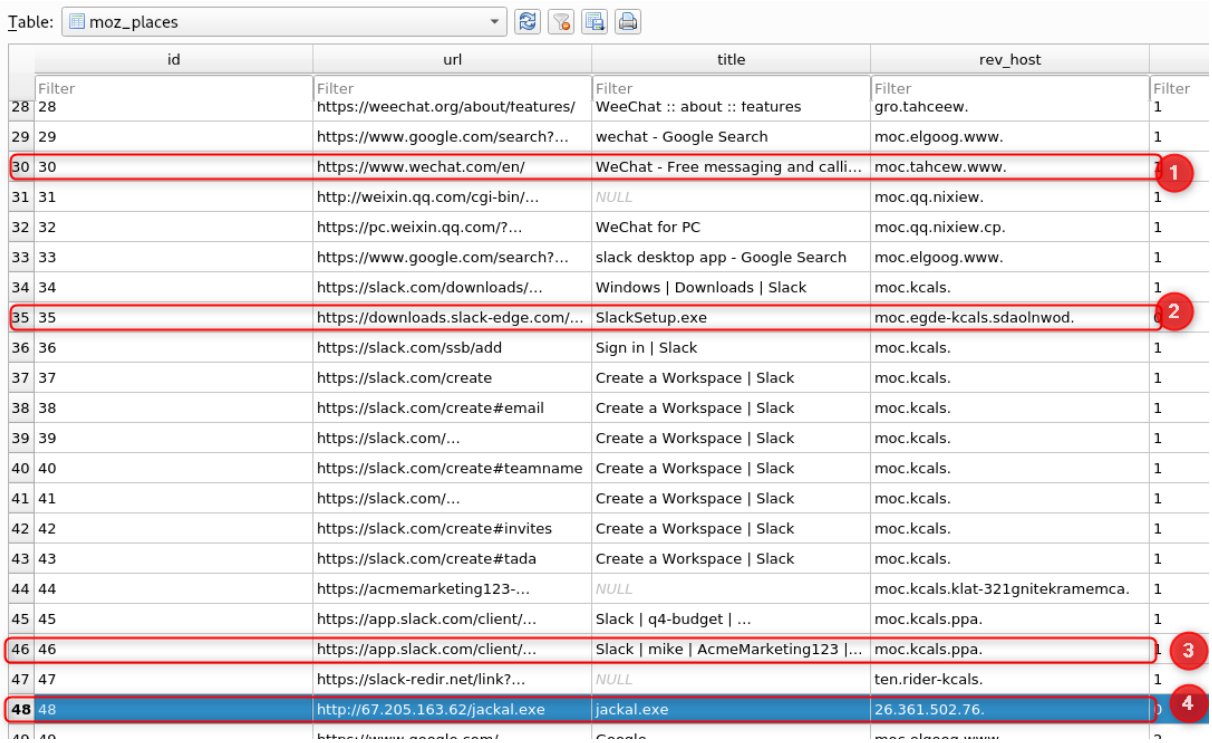


Table: moz_places

	id	url	title	rev_host	
	Filter	Filter	Filter	Filter	Filter
28	28	https://weechat.org/about/features/	WeeChat :: about :: features	gro.tahceew.	1
29	29	https://www.google.com/search?...	wechat - Google Search	moc.elgoog.www.	1
30	30	https://www.wechat.com/en/	WeChat - Free messaging and calli...	moc.tahcew.www.	1
31	31	http://weixin.qq.com/cgi-bin/...	NULL	moc.qq.nixiew.	1
32	32	https://pc.weixin.qq.com/?...	WeChat for PC	moc.qq.nixiew.cp.	1
33	33	https://www.google.com/search?...	slack desktop app - Google Search	moc.elgoog.www.	1
34	34	https://slack.com/downloads/...	Windows | Downloads | Slack	moc.kcals.	1
35	35	https://downloads.slack-edge.com/...	SlackSetup.exe	moc.egde-kcals.sdaolnwod.	0
36	36	https://slack.com/ssb/add	Sign in | Slack	moc.kcals.	1
37	37	https://slack.com/create	Create a Workspace | Slack	moc.kcals.	1
38	38	https://slack.com/create#email	Create a Workspace | Slack	moc.kcals.	1
39	39	https://slack.com/...	Create a Workspace | Slack	moc.kcals.	1
40	40	https://slack.com/create#teamname	Create a Workspace | Slack	moc.kcals.	1
41	41	https://slack.com/...	Create a Workspace | Slack	moc.kcals.	1
42	42	https://slack.com/create#invites	Create a Workspace | Slack	moc.kcals.	1
43	43	https://slack.com/create#tada	Create a Workspace | Slack	moc.kcals.	1
44	44	https://acmemarketing123-...	NULL	moc.kcals.klat-321gnitekramemca.	1
45	45	https://app.slack.com/client/...	Slack | q4-budget | ...	moc.kcals.ppa.	1
46	46	https://app.slack.com/client/...	Slack | mike | AcmeMarketing123 |...	moc.kcals.ppa.	1
47	47	https://slack-redir.net/link?...	NULL	ten.rider-kcals.	1
48	48	http://67.205.163.62/jackal.exe	jackal.exe	26.361.502.76.	0

Dans la table “moz_places” nous retrouvons plusieurs éléments intéressants. Dans un premier temps, la victime a utilisé “wechat” qui est une application de messagerie, donc nous pouvons supposer qu’une conversation a commencé par là entre elle et le hacker.

Ensuite, il télécharge `SlackSetup.exe`, donc une application qu’il n’avait pas auparavant. Après installation, nous remarquons qu’il créer un Workspace sur Slack et y invite son hacker. Nous y voyons même le nom “mike” apparaître au point 3. Et finalement l’accès à l’IP suspecte qui contient “jackal”, donc l’installation de notre malware et le début de nos problèmes.

Il existe aussi une table qui semble lister les téléchargements (SlackSetup.exe et jackal.exe retrouvable) :

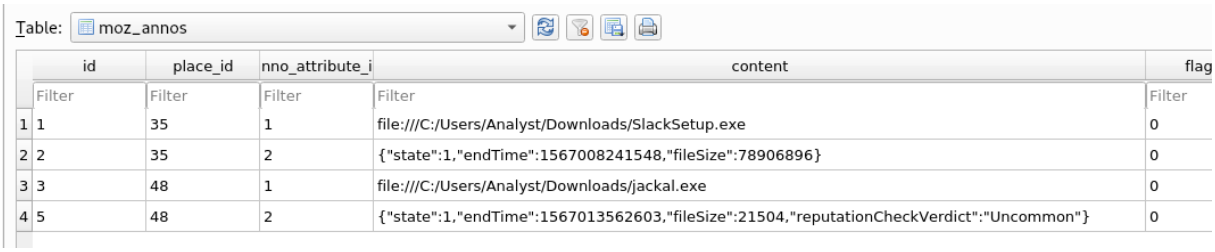


Table: moz_annos

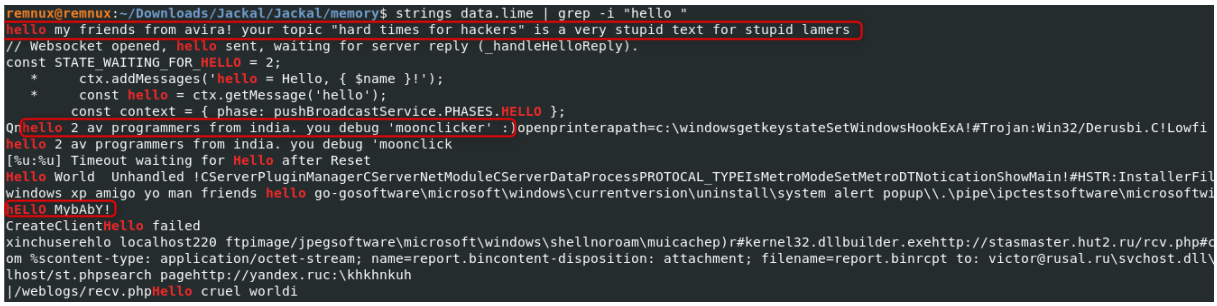
	id	place_id	nno_attribute_i	content	flag
	Filter	Filter	Filter	Filter	Filter
1	1	35	1	file:///C:/Users/Analyst/Downloads/SlackSetup.exe	0
2	2	35	2	{"state":1,"endTime":1567008241548,"fileSize":78906896}	0
3	3	48	1	file:///C:/Users/Analyst/Downloads/jackal.exe	0
4	5	48	2	{"state":1,"endTime":1567013562603,"fileSize":21504,"reputationCheckVerdict":"Uncommon"}	0

Conversation

Nous n’avons pas pu accéder à la conversation des 2 personnes (soi via Wechat, soi via Slack), il aurait été intéressant d’aller voir ce qui se trouve dans cette conversation `https://acmemarketing123-talk.slack.com/messages/CMVDCHQ7Q`, mais il aurait fallu avoir les informations d’authentification d’une des 2 personnes.

Cependant, pour aller plus loin, nous avons tout de même tenté de rechercher des mots communs pour un début de conversation comme “hi, hello, how are you, …”, etc… dans le dump mémoire

et nous avons étonnamment trouvé quelque chose avec `strings data.lime | grep -i "hello"`



Elle ne nous apporte pas tant d'éléments importants, mais nous pouvons remarquer des conversations potentielles, sans plus.

Adresses emails

Lors de l'analyse forensiques, nous avons effectué un strings sur "mike" et lors de l'affichage du résultat nous avons trouvé des adresses emails:

En appliquant le filtre suivant, nous pouvons trouver toutes les adresses emails dans le memory dump. Une des adresses intéressantes fait référence au fameux "mike". De plus, le domaine de l'adresse email est protonmail, tristement célèbre pour son utilisation dans le cadre d'affaires illégales de par son niveau de discrétion et sécurité.

```
1 strings data.lime | grep -E "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.(com|net
    |org|edu|gov|ru|ch|uk|de|jp|fr|au|us|ca|cn|es|mil|eu|info|io|biz)$"
```

Mitigation et blocage

Avec toutes ces analyses, nous sommes en mesure de détecter si jackal.exe venait à être installé sur une machine à nouveau :

- User-agent: `User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001`
- Toutes les IPs C2 liées à jackal trouvées plus haut.
- Détecter toutes les requêtes http comportant comme réponse ce commentaire html : `<!--j4ckal:YHt2f38zKiMqIw==-->` . Pour éviter une autre tentative, généraliser la règle à tout autre variant : `j4ckal:*`
- Bloquer tout programme utilisant la clé de registre : `Software\Microsoft\Windows Player`
- Bloquer tout logiciel, ou en tout cas, détecter et analyser profondément le comportement de tout logiciel utilisant le mutex Dassara.

- Finalement, former les employés à ne pas cliquer n'importe où et en cas de doute, appeler un supérieur/expert. Lors de l'analyse, un simple strings sur la chaîne "trojan" résulte de millier d'occurrences ce qui laisse penser que la personne propriétaire de cet ordinateur n'est manifestement pas du tout formée à utiliser un ordinateur de manière un minimum sécurisé.(Ces résultats n'ont pas été mis dans le rapport car ils n'étaient pas pertinents avec l'infection de `jackal.exe`)

Conclusion

`jackal.exe` est un logiciel malveillant basique, qui s'installe par le biais d'un utilisateur non sensibilisé au niveau minimum de sécurité à suivre lors de l'utilisation d'un outil informatique tel qu'un ordinateur.

De plus, une fois installé, celui-ci communique avec son propre user-agent, ce qui est très vite détectable par n'importe quel IDS ou AV, ce n'est pas très commun comme user-agent.

Par rapport au logiciel en lui-même, la technique d'obfuscation utilisée est très rudimentaire, la clé est très facilement trouvable en quelques secondes.

Le problème majeur est que lorsque celui-ci est installé sur la machine, il exécute des commandes vers un serveur distant pour savoir quelle commande exécuter par le hacker. Ainsi, la machine est contrôlable à distance par le hacker, ce qui est extrêmement critique.