
AMM - Laboratoire 7

Analyse du malware Jackal

Maxime Chantemargue, Ruben Pereira Lopes, Charles
Matrand



5 mai 2024

Table des matières

Résumé	2
Chronologie d'infection	2
Analyse	3
Analyse de l'exécutable	3
Liste des processus	3
Dump exécutable	3
Slack space	4
Valeurs en Base64	4
Clés de registre	5
Privilèges	6
Fonctionnalités	6
Mutex	7
Analyse réseau	7
Requête sur le serveur	8
Analyse de firefox	8
Historique web	10
Conversation	12
Adresses emails	13
Mitigation et blocage	13
Conclusion	14

Résumé

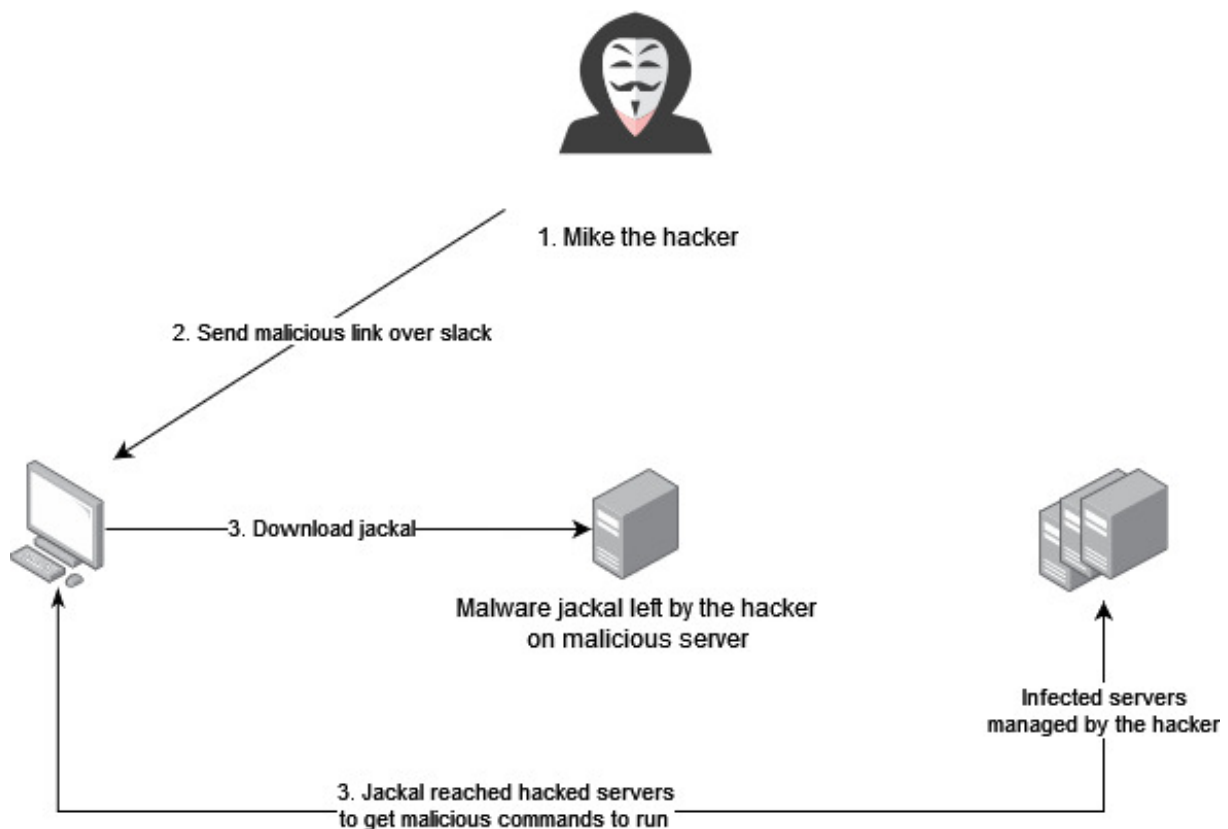
Un échantillon du malware Jackal a été analysé, provenant d'une image mémoire et d'une capture des communications entre l'appareil affecté et un serveur externe. L'objectif de cette analyse était de comprendre les fonctionnements et les méthodes d'infection du malware.

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.

```
remnux@remnux:/mnt/hgfs/share/amm/Jackal/Jackal/memory$ vol.py -f data.lime --profile=Win10x64_17134 pslist | grep jackal.exe
0xfffffe50cc9644580 jackal.exe 8628 2004 2 0 1 1 2019-08-28 17:32:54 UTC+0000
```

```
remnux@remnux:/mnt/hgfs/share/amm/Jackal/Jackal/memory$ vol.py -f data.lime --profile=Win10x64_17134 pslist | grep " 2004 "
0xfffffe50ccb09e580 firefox.exe 2004 7540 62 0 1 0 2019-08-28 15:50:41 UTC+0000
```

Dump exécutable

Maintenant que nous avons son PID, nous pouvons dumper l'exécutable avec `vol.py` (`http://vol.py/`) `procdump -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 -e1 executable.8628_mem.exe`, il y a une partie qui nous intéresse :

```

1  JyM9IiM9ISaQpSEiKzx5emBnPXLg
2  ISElPSEjIj0iJCA9JCE8e3x8d2R6fXg8RXJ9cHxmZXZhPGV6YHpnPGBmfn52YT17Z35/
3  ISMjPSInJz0mIz0hIiE8YXZgcN5jf3Z3PHF2dXxhdj17Z35/
4  ISEiPSIlJj0iJSc9JiU8Y2Z/f3FycHg8anZge3plcns9Z2tn
5  JyY9IiYiPSIrID0iJyo8eHJge3ZhPEt2YXxren10PXLg
6  ISEkPSEhPSImJD0mPGZgcjxgfHBwdmE9e2d+fw==
7  ISIrPSIiID0iJCs9JCI8ZHzyZ3t2YT17Z35/
8  IiQrPSsmPSIqIj0mITx2dmNhFH48YHZldmF2PXtnfn8=
9  IiMiPSogPSohPSIlJDx8YGd2fWB8YWo8ZHpwH3PHJ/fw==
10 IiA9KyE9IiYiPSEiJjx/fGNjdmE8d2Z+cXF2f389Z2tn
11 KiM9ICY9ISAnPSEnKzx8YXp2fWc8cXJ4dmFgPXtnfn8=
12 IiMmPSQiPSEgJD0iJTxcgZmN2YTxf3JxcXZhYD1rfn8=
13 ICE9ISc9IScrPSIkKzx0fHx3emB7PHV2f398ZD17Z35/
14 IiMrPSIiJj0qKj0rJTxdn9/fGR7cn5+dmE8YXZjFGFnYA==
15 IiErPSIgJz0iJCU9IiElPHZremBnYDxDcmByd3Z9cj13fHA=
16 JSQ9KyY9IScrPSEmPGRyenVgPHlmfXw9e2d+fw==
17 ISMhPSIqPSIqJD0iICI8fHBnfHF2YTxcgcmdmYX09eWA=
18 IiYqPSEgIT0iIyE9IiYrPGNyYGB6fXQ8YHxmYXB2PHt8fnY9cmBj
19 IiUlPSEjPSYgPSEiKjxRZmFxcn14PF9mcHp3PHlyCHhgPXtnfn8=
20 IiclPSshPSQkPSYqPGAqIioiIjx4f2B5ciIiPHV6f2d2YT1na2c=
21 ISAgPSIkIT0iKyM9IiUgPGFyfXB7dmE8ZHp9d3xkYD1je2M=
22 ISYhPSEhKj0iKiA9ISEkPGBnfHB4PGdhcnd6fXQ9e2d+fw==
23 ISIhPScgPSInIz0iJiE8YGN8YWc8e3Jje3JpcmF3PXLg
24 IisnPSscrPSInID0iIiQ8ZHJ6Z3p9dD1yYGM=
25 ISYgPSEhJz0iJCI9ICo8fXZkYDxfH5+PWN7Yw==
26 Jic9IiAqPSIrIz0iICs8YXxyYWA8ZXJhenxmdzxyf2d2YX1yZ3p8fT17Z35/
27 IiUrPSsnPSIqKz0hJys8YHZ2fTxqdn9jPXtnfn8=
28 IismPSEhKj0iJiQ9IiUrPHx9YXZwanZ7fDxldmFxcn89Z2tn
29 ISEqPSY9ISAgPSEjJDxndnpqajxgY39meD1yYGM=
30 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 :

```
1 40.10.239.218/jist.js
2 226.201.173.72/hoodwink/Vancouver/visit/summer.html
3 200.144.50.212/resampled/before.html
4 221.165.164.56/pullback/yeshivah.txt
5 45.151.183.149/kasher/Xeroxing.js
6 227.22.157.5/usa/soccer.html
7 218.113.178.71/weather.html
8 178.85.191.52/eprom/severe.html
9 101.93.92.167/ostensory/wicked/all
10 13.82.151.215/lopper/dumbbell.txt
11 90.35.234.248/orient/bakers.html
12 105.71.237.16/super/clabbers.xml
13 32.24.248.178/goodish/fellow.html
14 108.115.99.86/yellowhammer/reports
15 **128.134.176.126/exists/Pasadena.doc** => revient dans le netscan
16 67.85.248.25/waifs/juno.html
17 202.19.197.131/october/saturn.js
18 159.232.102.158/passing/source/home.asp
19 166.20.53.219/Burbank/Lucid/jacks.html
20 146.82.77.59/s91911/kljsa11/filter.txt
21 233.172.180.163/rancher/windows.php
22 252.229.193.227/stock/trading.html
23 212.43.140.152/sport/haphazard.js
24 184.48.143.117/waiting.asp
25 253.224.171.39/news/comm.php
26 54.139.180.138/roars/varioud/alternation.html
27 168.84.198.248/seen/yelp.html
28 185.229.157.168/onrecyeho/verbal.txt
29 229.5.233.207/teiy/spluk.asp
30 194.209.89.41/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) ISMhPSIqPSIqJD0iICI8fHBnfHF2YTxcgcmdmYX09eWA=
REG_SZ 0GF9 : (S) JSQ9KyY9IScrPSEmPGRyenVgPHlmfXw9e2d+fw==
REG_SZ IYCD : (S) IiErPSIgJz0iJCU9IiElPHZremBnYDxDcmByd3Z9cj13fHA=
REG_SZ 780I : (S) IiMrPSiIj0qKj0rJTxqdn9/fGR7cn5+dmE8YXZjfgFnYA==
REG_SZ YBTI : (S) ICE9ISc9IScrPSIkKzx0fHx3emB7PHV2f398ZD17Z35/
REG_SZ THRG : (S) IiMmPSQiPSEgJD0iJTxgZmN2YTxf3JxcXZhYD1rfn8=
REG_SZ PXDT : (S) KiM9ICY9ISAnPSEnKzx8YXp2fWc8cXJ4dmFgPXtnfn8=
REG_SZ FYNO : (S) IiA9KyE9IiYiPSEiJjx/fGNjdmE8d2Z+cXF2f389Z2tn
REG_SZ IWT5 : (S) IiMiPSogPSohPSILJDx8YGD2fWB8YWo8ZHpwH3PHJ/fw==
REG_SZ 6NLE : (S) IiQrPSsmPSIqIj0mITx2dmNhF48YHZldmF2PXtnfn8=
REG_SZ X0JV : (S) ISIRPSiID0iJCs9JCI8ZHyZ3t2YT17Z35/
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) JyM9IiM9ISAgPSEiKzx5emBnPXlg

```

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
2	CID Name				
3	-----	-----	-----	-----	-----
4	[...]				
5	**0x0000e50cc83d7ec0	2	1	0	0xfffffe50cc9ff1700
6	8628:5876 __Dassara__**				
7	[...]				
8	0x0000e50ccaa013f0	32768	1	1	0x0000000000000000
9	SM0:8628:64:WilError_01				
10	0x0000e50ccad25550	32768	1	1	0x0000000000000000
11	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	
3	LISTENING		8628	jackal.exe	2019-08-28
4	18:41:54 UTC+0000**				
5	[...]				
6	**0xe50cca1d4010	TCPv4	192.168.231.131:52349		
7	128.134.176.126:80	CLOSED	-1		
8	6490-05-12 05:38:17 UTC+0000**				
9	[...]				

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 :

No.	Time	Source	Destination	Protocol	Length	Info
18	0.004871	softbank218113178071.bbtec.net	172.16.237.134	HTTP	870	HTTP/1.1 200 OK (text/html)
6	0.002100	softbank218113178071.bbtec.net	172.16.237.134	HTTP	610	HTTP/1.1 301 Moved Permanently (text/html)
7	0.003545	172.16.237.134	softbank218113178071.bbtec.net	HTTP	289	GET /index.html HTTP/1.1
4	0.001236	172.16.237.134	softbank218113178071.bbtec.net	HTTP	267	GET /weather.html HTTP/1.1

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 :

The image shows a Wireshark packet capture of an HTTP response. The packet list on the left shows a packet from 172.16.237.134 to softbank218113178071.bbtec.net. The packet details pane shows the HTTP response structure, including the status code 200 OK. The packet bytes pane shows the raw data, and the text pane shows the decoded HTML content. The HTML content includes a base64-encoded comment: `<!-- Ceci est 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.

```
driftt_aid=3d9f3584-c994-4e6d-b934-c07dfaaf67a0; Domain=slack.com; Path=/
__qca=Pe-169849723-1567008231563; Domain=slack.com; Path=/
visitor_id755253=58e58584; Domain=slack.com; Path=/
visitor_id755253-hash=84e87b2b53a62ecd2502efc774ce81b980ee2791fa2eb2acc9eddb29e628da4e6fd9a4de5ca1834e7b830647c171057aee352054c; Domain=slack.com; Path=/
driftt_sid=2432e1be-50ea-4b5d-9037-a0b98fc2ebcf; Domain=slack.com; Path=/
visitor_id755253=58e58584; Domain=campaign.slack.com; Path=/
visitor_id755253-hash=9d8482254415eedd91acd2500a05b25f1fce4a201e6dfb6b6681b7681c17febfc183d62f75dbab41efed1139f76ec033b574; Domain=campaign.slack.com; Path=/
DFTT_END_USER_PREV_BOOTSTRAPPED=true; Domain=slack.com; Path=/
optimizeUserId=oeu1567008228133ro.407690231928394; Domain=slack.com; Path=/
ctm='pgv':7968415188451285|'vt':'2499940619085003'|'vstr':'8421353183465750'|'intr':'1567008310679'|'v':1; Domain=adobe.com; Path=/
d=0KxPaX07a193Vh0P5XADT661Qr1VhCvVVR2QJdkc1K70RsYepVWZ1STBGamcWJFVTFL0QS9JMVFPk29NcoH4b120VJmTWgwMTBuzGYvby9NMXBXaVNM3U2N0ZqR0IvS1p2R0RBVzI4ZzhxWEdrMEwraGLUb1BRTKVEbVU3a
lc=156700827; Domain=slack.com; Path=/; Secure
_swnid=jh0syjmkh6vd; Domain=spiceworks.com; Path=/
_swaithn; Domain=spiceworks.com; Path=/
_fbp=fb.1.1567008229933.1301530395; Domain=slack.com; Path=/
b=7qxcbrxwncs596bv042mcfq1; Domain=slack.com; Path=/
x=7qxcbrxwncs596bv042mcfq1.1567008227; Domain=slack.com; Path=/
MS0=d26d95ec10144df98306369b3d87b12d; Domain=microsoft.com; Path=/
_ga=GA1.2.1427820679.1567008230; Domain=slack.com; Path=/
_gid=GA1.2.1940167992.1567008230; Domain=slack.com; Path=/
_gat_UA=56978219-1=1; Domain=slack.com; Path=/
```

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.

```
remnux@remnux:/mnt/hgfs/share/amm/Jackal/Jackal/memory$ strings -t x data.lime | grep -E "slack.*jackal|jackal.*slack"
dd56a17 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3ten.rider-kcals.d
10499a68 URL=https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
170d1ee8 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b0180 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b0360 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b0420 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b04e0 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b0600 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
171b0720 page-icon:https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
17bcdd0 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
17bcce0 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
17bccf0 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
17bccf0 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
17bcd090 https://slack-redir.net/link?url=http%3A%2F%2F67.205.163.62%2Fjackal.exe6v=3
```

Sachant que jackal.exe a été téléchargé par ce biais, en faisant un strings avec “jackal” et “Jackal”, le résultat a affiché un user-agent. 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 :

```
remnux@remnux:/mnt/hgfs/share/amm/Jackal/Jackal/memory$ strings -t x data.lime | grep -i -A 3 -B 2 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001"
55cb07a0 Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001
55cb0a40 GET /exists/Pasadena.doc HTTP/1.1
55cb0a8d Accepttext/*text/*, image/giftext/*, image/gif, application/octet-streampzw
55cb0b01 User-AgentMozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001
55cb0b7c Host128.134.176.126GET /exists/Pasadena.doc HTTP/1.1

12675fc30 GET /Burbank/Lucid/jacks.html HTTP/1.1
12675fc82 Accepttext/*text/*, image/giftext/*, image/gif, application/octet-streamE
12675fcf6 User-AgentMozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; The Jackal v4.2001
12675fd71 Host166.20.53.219GET /Burbank/Lucid/jacks.html HTTP/1.1
```

```
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 v4.20018
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] (<http://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\2uhkqvw1.default-release\favicon.sqlite
1 R--rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\favicon.sqlite
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\places.sqlite
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\favicon.sqlite-wal
0 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\cache2\entries\33704
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\websites.sqlite
1 RW-rw- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\websites.sqlite-shm
1 RWp- \Device\HarddiskVolume3\Users\Analyst\AppData\Roaming\Mozilla\Firefox\Profiles\2uhkqvw1.default-release\websites.sqlite-wal
```

Maintenant que nous avons cette information, il faut dump le fichier. Pour cela, nous avons utilisé la commande `v[ol.py] (http://ol.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	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.	2
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.	3
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.	4

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 crée 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 retrouvables) :

Table: moz_annos

	id	place_id	nno_attribute_i	content	flag
Filter	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"`

```
remnux@remnux:~/Downloads/Jackal/Jackal/memory$ strings data.lime | grep -i "hello "
hello my friends from avira! your topic "hard times for hackers" is a very stupid text for stupid lamers
// Websocket opened, hello sent, waiting for server reply (_handleHelloReply).
const STATE_WAITING_FOR_HELLO = 2;
*   ctx.addMessages('hello = Hello, { $name }!');
*   const hello = ctx.getMessage('hello');
*   const context = { phase: pushBroadcastService.PHASES_HELLO };
On hello 2 av programmers from india. you debug 'moonclicker' :openprinterapath=c:\windows\system32\cmd.exe /c whoami
hello 2 av programmers from india. you debug 'moonclicker' :openprinterapath=c:\windows\system32\cmd.exe /c whoami
[%u:%u] Timeout waiting for Hello after Reset
Hello World Unhandled !CServerPluginManagerCServerNetModuleCServerDataProcessPROTOCOLAL_TYPEIsMetroModeSetMetroDTNotificationShowMain!#HSTR:InstallerFil
windows xp amigo yo man friends hello go-gosoftwaresoftware\microsoft\windows\currentversion\uninstall\system alert popup\pipe\ipctestssoftware\microsoftwi
HELLO MyBaby!
CreateClientHello failed
xinchuserhello localhost220 ftpimage/jpegsoftware\microsoft\windows\shell\noroom\muicachepr#kernel32.dllbuilder.exehttp://stasmaster.hut2.ru/rcv.php#co
om %scontent-type: application/octet-stream; name=report.bincontent-disposition: attachment; filename=report.binrcpt to: victor@rusal.ru\svchost.dll\
hlost/st.phpsearch pagehttp://yandex.ruc:\khkhkhkuh
|/weblogs/rcv.phpHello cruel world!
```

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:

```
1e2594e0 ^((de\.firefoxextension12345@asdf\.pl)|(deex1@de\.com)|(esex1@ese\.com)|(estrellach@protonmail\.com)|(fifi312@protonmail\.com)|(finex
1@fin\.com)|(firefoxextension12345@asdf\.pl)|(firefoxextension1234@asdf\.pl)|(firefoxextension12345@asdf\.pl)|(firefoxextension123456@asdf\.pl)|
(frexxff1@frexxff1\.com)|(frexxff2@frexxff2\.com)|(frexxff3@frexxff3\.com)|(ind@niepodam\.pl)|(jacob4311@protonmail\.com)|(javonnu144@protonmail\.co
m)|(keellon33-ff@protonmail\.com)|(keellon33@protonmail\.com)|(masetoo4113@protonmail\.com)|(mikecosenti11@protonmail\.com)|(paigecho@protonma
il\.com)|(salooo12@protonmail\.com)|(swex1@swe\.com)|(swex2@swe\.com)|(swex3@swe\.com)|(willburpoor@protonmail\.com)|(williamhibburn@protonmai
l\.com))$
```

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 milliers 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.