

# LSE - Recrutement 2020-2021 - Phase II

Thomas Berlioz, thomas.berlioz, thomas.berlioz@epita.fr

January 2021

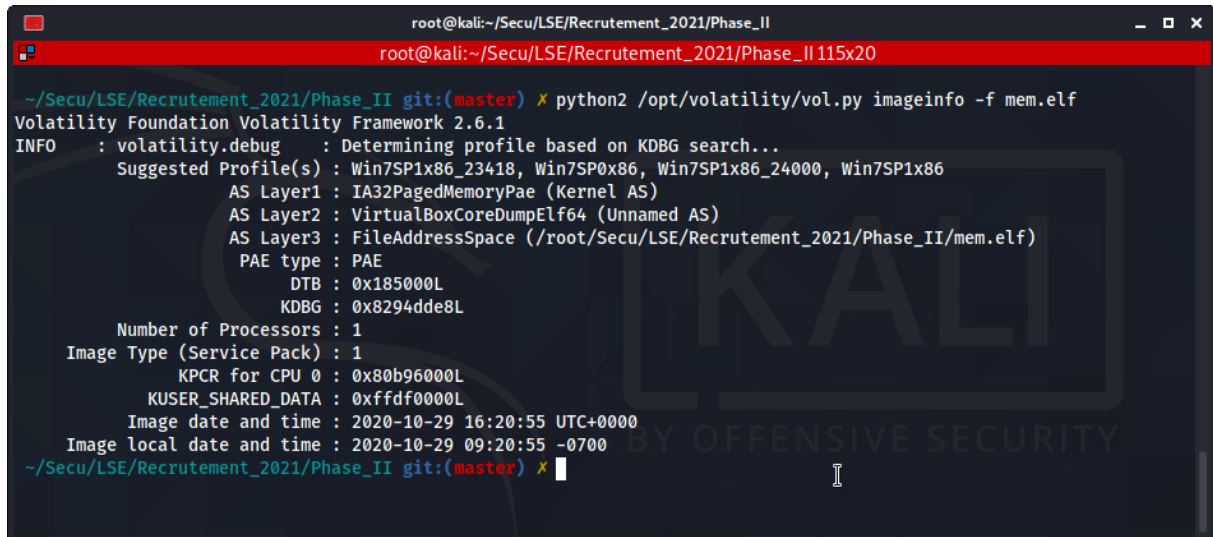
## Table des matières

<b>1</b>	<b>Identification du système d'exploitation</b>	<b>2</b>
<b>2</b>	<b>Détection de programmes suspects</b>	<b>3</b>
<b>3</b>	<b>Recherche d'une connexion</b>	<b>6</b>
<b>4</b>	<b>Analyse du <i>malware</i></b>	<b>8</b>
<b>5</b>	<b>Etat de l'attaque</b>	<b>12</b>
<b>6</b>	<b>Identification de l'attaquant</b>	<b>16</b>

# 1 Identification du système d'exploitation

*Quel est le système d'exploitation avec lequel a été fait le dump de ram ?*

Afin d'analyser ce *dump*, on utilise la version Python 2 de volatility. La commande `imageinfo` permet d'obtenir le système d'exploitation du *dump*, ainsi que le profil précis à utiliser pendant l'analyse.



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 115x20

~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ python2 /opt/volatility/vol.py imageinfo -f mem.elf
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
      AS Layer1 : IA32PagedMemoryPae (Kernel AS)
      AS Layer2 : VirtualBoxCoreDumpElf64 (Unnamed AS)
      AS Layer3 : FileAddressSpace (/root/Secu/LSE/Recrutement_2021/Phase_II/mem.elf)
      PAE type : PAE
      DTB : 0x185000L
      KDBG : 0x8294dde8L
      Number of Processors : 1
      Image Type (Service Pack) : 1
      KPCR for CPU 0 : 0x80b96000L
      KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2020-10-29 16:20:55 UTC+0000
      Image local date and time : 2020-10-29 09:20:55 -0700
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

FIGURE 1 – `python2 /opt/volatility/vol.py imageinfo -f mem.elf`

Le système d'exploitation est donc Windows 7, architecture 32 *bits* et on suppose ici que c'est le pack SP1 car il n'a pas de différence dans les commandes existantes des deux profils. Il est d'ailleurs judicieux de faire dès maintenant un alias pour enchaîner plus facilement les commandes.

---

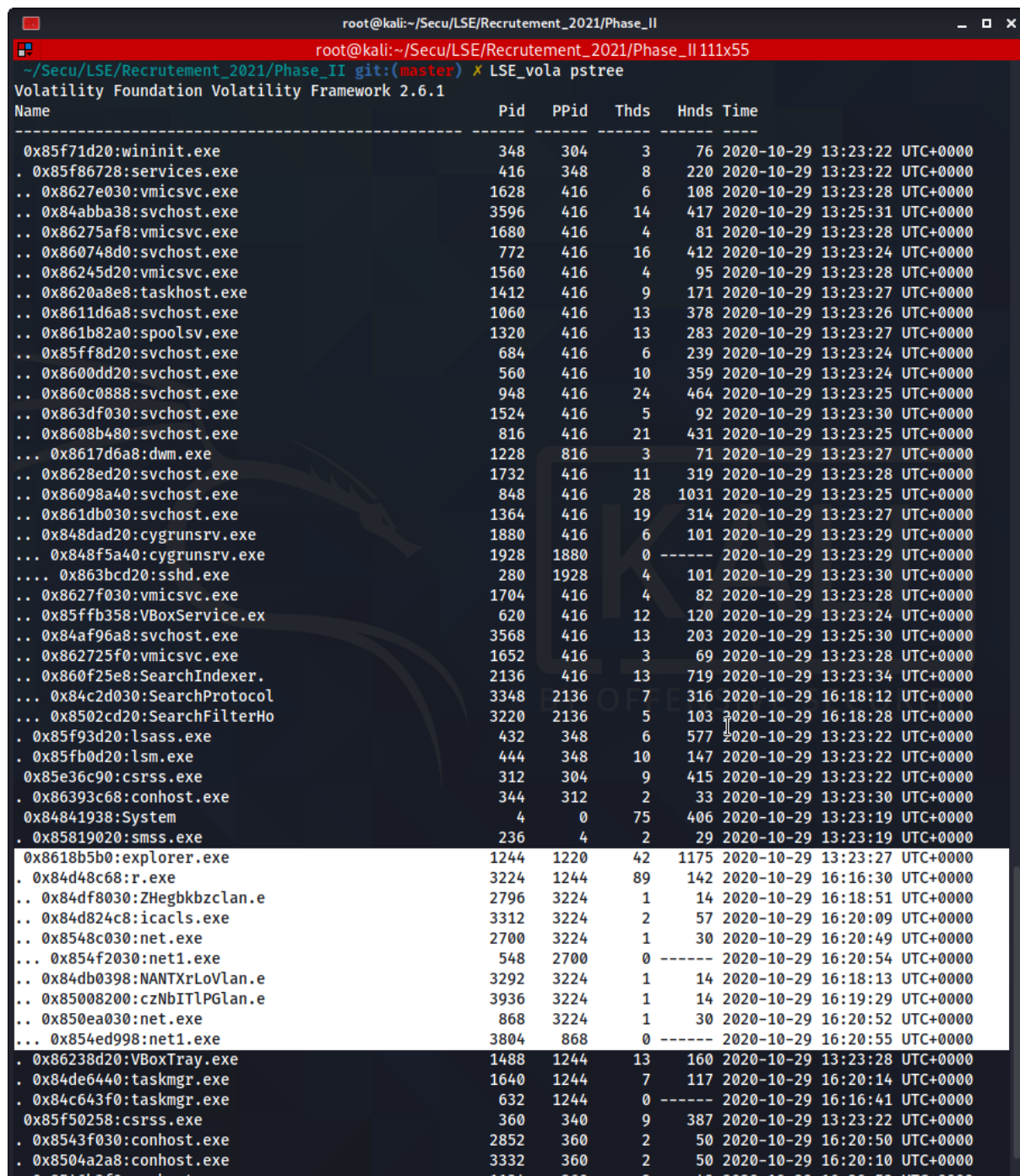
```
$ alias LSE_vola='python2 /opt/volatility/vol.py --profile=Win7SP1x86 -f mem.elf'
```

---

## 2 Détection de programmes suspects

*Citer le(s) process qui vous paraissent malveillant(s) ?*

On commence par lister les *process* avec `pstree`, afin d'avoir l'arborescence parent - enfant.



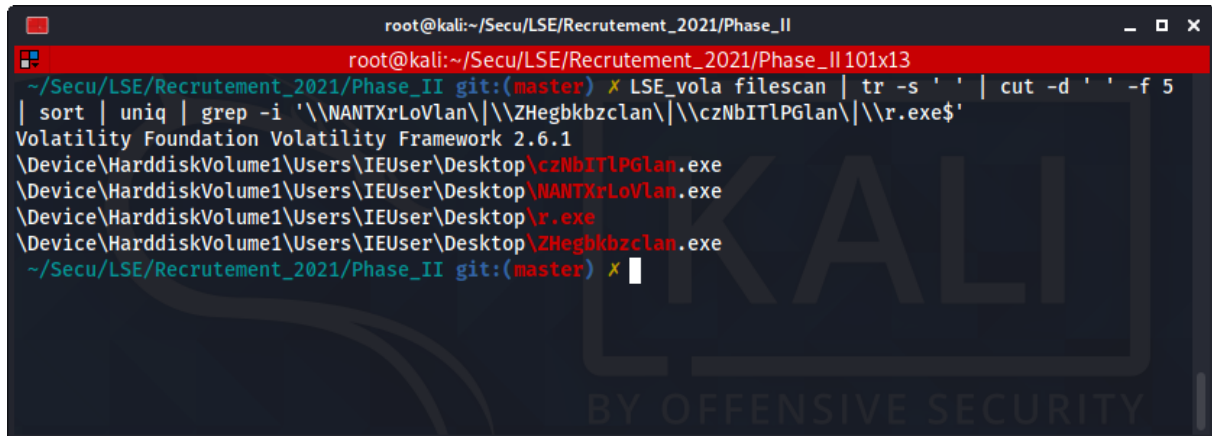
Name	Pid	PPid	Thds	Hnds	Time
0x85f71d20:wininit.exe	348	304	3	76	2020-10-29 13:23:22 UTC+0000
.. 0x85f86728:services.exe	416	348	8	220	2020-10-29 13:23:22 UTC+0000
.. 0x8627e030:vmicsvc.exe	1628	416	6	108	2020-10-29 13:23:28 UTC+0000
.. 0x84abba38:svchost.exe	3596	416	14	417	2020-10-29 13:25:31 UTC+0000
.. 0x86275af8:vmicsvc.exe	1680	416	4	81	2020-10-29 13:23:28 UTC+0000
.. 0x860748d0:svchost.exe	772	416	16	412	2020-10-29 13:23:24 UTC+0000
.. 0x86245d20:vmicsvc.exe	1560	416	4	95	2020-10-29 13:23:28 UTC+0000
.. 0x8620a8e8:taskhost.exe	1412	416	9	171	2020-10-29 13:23:27 UTC+0000
.. 0x8611d6a8:svchost.exe	1060	416	13	378	2020-10-29 13:23:26 UTC+0000
.. 0x861b82a0:spoolsv.exe	1320	416	13	283	2020-10-29 13:23:27 UTC+0000
.. 0x85ff8d20:svchost.exe	684	416	6	239	2020-10-29 13:23:24 UTC+0000
.. 0x8600dd20:svchost.exe	560	416	10	359	2020-10-29 13:23:24 UTC+0000
.. 0x860c0888:svchost.exe	948	416	24	464	2020-10-29 13:23:25 UTC+0000
.. 0x863df030:svchost.exe	1524	416	5	92	2020-10-29 13:23:30 UTC+0000
.. 0x8608b480:svchost.exe	816	416	21	431	2020-10-29 13:23:25 UTC+0000
... 0x8617d6a8:dwm.exe	1228	816	3	71	2020-10-29 13:23:27 UTC+0000
.. 0x8628ed20:svchost.exe	1732	416	11	319	2020-10-29 13:23:28 UTC+0000
.. 0x86098a40:svchost.exe	848	416	28	1031	2020-10-29 13:23:25 UTC+0000
.. 0x861db030:svchost.exe	1364	416	19	314	2020-10-29 13:23:24 UTC+0000
.. 0x848dad20:cygrunsrv.exe	1880	416	6	101	2020-10-29 13:23:29 UTC+0000
... 0x848f5a40:cygrunsrv.exe	1928	1880	0	-----	2020-10-29 13:23:29 UTC+0000
.... 0x863bcd20:sshd.exe	280	1928	4	101	2020-10-29 13:23:30 UTC+0000
.. 0x8627f030:vmicsvc.exe	1704	416	4	82	2020-10-29 13:23:28 UTC+0000
.. 0x85ffb358:VBoxService.exe	620	416	12	120	2020-10-29 13:23:24 UTC+0000
.. 0x84af96a8:svchost.exe	3568	416	13	203	2020-10-29 13:25:30 UTC+0000
.. 0x862725f0:vmicsvc.exe	1652	416	3	69	2020-10-29 13:23:28 UTC+0000
.. 0x860f25e8:SearchIndexer.exe	2136	416	13	719	2020-10-29 13:23:34 UTC+0000
... 0x84c2d030:SearchProtocolHost.exe	3348	2136	7	316	2020-10-29 16:18:12 UTC+0000
... 0x8502cd20:SearchFilterHost.exe	3220	2136	5	103	2020-10-29 16:18:28 UTC+0000
.. 0x85f93d20:lsass.exe	432	348	6	577	2020-10-29 13:23:22 UTC+0000
.. 0x85fb0d20:lsm.exe	444	348	10	147	2020-10-29 13:23:22 UTC+0000
.. 0x85e36c90:csrss.exe	312	304	9	415	2020-10-29 13:23:22 UTC+0000
.. 0x86393c68:conhost.exe	344	312	2	33	2020-10-29 13:23:30 UTC+0000
.. 0x84841938:System	4	0	75	406	2020-10-29 13:23:19 UTC+0000
.. 0x85819020:smss.exe	236	4	2	29	2020-10-29 13:23:19 UTC+0000
.. 0x8618b5b0:explorer.exe	1244	1220	42	1175	2020-10-29 13:23:27 UTC+0000
.. 0x84d48c68:r.exe	3224	1244	89	142	2020-10-29 16:16:30 UTC+0000
.. 0x84df8030:Zhegkbzclan.exe	2796	3224	1	14	2020-10-29 16:18:51 UTC+0000
.. 0x84d824c8:icaccls.exe	3312	3224	2	57	2020-10-29 16:20:09 UTC+0000
.. 0x8548c030:net.exe	2700	3224	1	30	2020-10-29 16:20:49 UTC+0000
... 0x854f2030:net1.exe	548	2700	0	-----	2020-10-29 16:20:54 UTC+0000
.. 0x84db0398:NANTXrLoVlan.exe	3292	3224	1	14	2020-10-29 16:18:13 UTC+0000
.. 0x85008200:czNbITLPGlan.exe	3936	3224	1	14	2020-10-29 16:19:29 UTC+0000
.. 0x850ea030:net.exe	868	3224	1	30	2020-10-29 16:20:52 UTC+0000
... 0x854ed998:net1.exe	3804	868	0	-----	2020-10-29 16:20:55 UTC+0000
.. 0x86238d20:VBoxTray.exe	1488	1244	13	160	2020-10-29 13:23:28 UTC+0000
.. 0x84de6440:taskmgr.exe	1640	1244	7	117	2020-10-29 16:20:14 UTC+0000
.. 0x84c643f0:taskmgr.exe	632	1244	0	-----	2020-10-29 16:16:41 UTC+0000
.. 0x85f50258:csrss.exe	360	340	9	387	2020-10-29 13:23:22 UTC+0000
.. 0x8543f030:conhost.exe	2852	360	2	50	2020-10-29 16:20:10 UTC+0000
.. 0x8504a2a8:conhost.exe	3332	360	2	50	2020-10-29 16:20:10 UTC+0000

FIGURE 2 – pstree

Au milieu des *process* habituels de Windows comme les *Service Host Processes* `svchost.exe` ou le *Desktop Window Manager* `dwm.exe`, on en remarque facilement avec des noms suspects :

ZHegbkbzclan.exe, NANTXrLoVlan.exe et czNbITlPGlan.exe. Plus alarmant, ils semblent interagir avec `icaccls.exe`, utilisé pour modifier les permissions sur des fichiers systèmes NTFS, ainsi qu'avec `net.exe` et `net1.exe` qui contrôlent le protocole IPv6 ainsi que plusieurs autres fonctionnalités LAN.

Il serait intéressant de voir où sont ces programmes dans le système.



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 101x13
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola filescan | tr -s ' ' | cut -d ' ' -f 5
| sort | uniq | grep -i '\\NANTXrLoVlan\\|\\ZHegbkbzclan\\|\\czNbITlPGlan\\|\\r.exe$'
Volatility Foundation Volatility Framework 2.6.1
\\Device\\HarddiskVolume1\\Users\\IEUser\\Desktop\\czNbITlPGlan.exe
\\Device\\HarddiskVolume1\\Users\\IEUser\\Desktop\\NANTXrLoVlan.exe
\\Device\\HarddiskVolume1\\Users\\IEUser\\Desktop\\r.exe
\\Device\\HarddiskVolume1\\Users\\IEUser\\Desktop\\ZHegbkbzclan.exe
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

FIGURE 3 – filescan

Ils se situent sur le bureau de l'utilisateur, avec le programme parent. On peut considérer cela suffisant pour que ce soit suspect dans une première analyse afin de trouver des programmes malveillants.

On essaye quand même de voir dès maintenant si on trouve pas l'origine de ces programmes suspects dans l'historique internet avec un téléchargement louche par exemple, mais sans succès. On ne trouve rien de très intéressant entre un *debugger* et un site de *softwares*.

```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 94x33
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola iehistory | grep Location
Volatility Foundation Volatility Framework 2.6.1
Location: https://www.jam-software.de/customers/includes/functions.js
Location: https://www.jam-software.de/fonts/stylesheetIE.css
Location: https://www.jam-software.de/web/js/modernizr-2.7.min.js
Location: https://www.jam-software.de/css/reset_v4.css
Location: https://www.jam-software.de/css/screen_v4.css
Location: https://www.jam-software.de/css/shop_v4.css
Location: https://www.jam-software.de/img/icons/TreeSize-Icon-16.png
Location: https://www.jam-software.de/img/icons/UltraSearch-Icon-16.png
Location: https://www.jam-software.de/web/js/jquery.tools.min.js
Location: https://www.jam-software.de/img/icons/TreeSizeFree-Icon-16.png
Location: https://www.jam-software.de/img/icons/SmartPOP2Exchange-Icon-16.png
Location: https://www.jam-software.de/img/icons/SpamAssassin-Icon-16.png
Location: https://www.jam-software.de/img/icons/SpamAssassinInaBox-Icon-16.png
Location: https://www.jam-software.de/img/icons/VirtualTreeview-Icon-16.png
Location: https://www.jam-software.de/img/icons/ShellBrowserComponents-Icon-16.png
Location: https://www.jam-software.de/img/icons/SEPA-Transfer-Icon-16.png
Location: https://www.jam-software.de/img/icons/SmartSerialMailFree-Icon-16.png
Location: https://www.jam-software.de/img/icons/HeavyLoad-Icon-16.png
Location: https://www.jam-software
Location: Cookie:ieuser@www.jam-software.de/
Location: Cookie:ieuser@www.jam-software.de/
Location: Visited: IEUser@https://www.microsoft.com
Location: Visited: IEUser@file:///C:/Users/IEUser/Desktop/OSEF0B7.tmp
Location: Visited: IEUser@https://dev.modern.ie/welcome/Win7/IE8/20150916
Location: Visited: IEUser@file:///C:/Users/IEUser/Desktop/DbgChild.Beta.10.zip
Location: Visited: IEUser@https://dev.modern.ie/welcome/Win7/IE8/20150916
Location: Visited: IEUser@https://dev.modern.ie/welcome/Win7/IE8/20150916
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ htop
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

FIGURE 4 – iehistory

### 3 Recherche d'une connexion

*Y a-t-il une connexion avec l'extérieur ?*

On commence par utiliser `netscan` pour trouver des connexions. C'est malheureusement la seule commande existante sur ce profil avec `volatility` pour du réseau.

```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 152x47
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola netscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P) Proto Local Address Foreign Address State Pid Owner Created
0x7e2b2f50 UDPv4 127.0.0.1:65318 *:* 3568 svchost.exe 2020-10-29 13:27:35 UTC+0000
0x7e225208 TCPv4 0.0.0.0:49156 0.0.0.0:0 LISTENING 432 lsass.exe
0x7e225208 TCPv6 :::49156 :::0 LISTENING 432 lsass.exe
0x7e5d58f0 UDPv4 0.0.0.0:0 *:* 1524 svchost.exe 2020-10-29 13:23:31 UTC+0000
0x7e5d58f0 UDPv6 :::0 *:* 1524 svchost.exe 2020-10-29 13:23:31 UTC+0000
0x7e5eb48 UDPv4 0.0.0.0:0 *:* 1524 svchost.exe 2020-10-29 13:23:31 UTC+0000
0x7e6177b8 UDPv4 0.0.0.0:53970 *:* 1060 svchost.exe 2020-10-29 13:24:11 UTC+0000
0x7e64fc70 UDPv6 :::1:1900 *:* 3568 svchost.exe 2020-10-29 13:27:35 UTC+0000
0x7e65c9c0 UDPv4 127.0.0.1:1900 *:* 3568 svchost.exe 2020-10-29 13:27:35 UTC+0000
0x7eb89b78 UDPv4 0.0.0.0:0 *:* 1060 svchost.exe 2020-10-29 15:21:18 UTC+0000
0x7eb89b78 UDPv6 :::0 *:* 1060 svchost.exe 2020-10-29 15:21:18 UTC+0000
0x7eb95938 UDPv4 0.0.0.0:3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7e49ab68 TCPv4 0.0.0.0:22 0.0.0.0:0 LISTENING 280 sshd.exe
0x7e581490 TCPv4 0.0.0.0:49156 0.0.0.0:0 LISTENING 432 lsass.exe
0x7e584190 TCPv4 0.0.0.0:49155 0.0.0.0:0 LISTENING 416 services.exe
0x7e584190 TCPv6 :::49155 :::0 LISTENING 416 services.exe
0x7e594958 TCPv4 0.0.0.0:445 0.0.0.0:0 LISTENING 4 System
0x7e594958 TCPv6 :::445 :::0 LISTENING 4 System
0x7e5a83c0 TCPv4 0.0.0.0:22 0.0.0.0:0 LISTENING 280 sshd.exe
0x7e5a83c0 TCPv6 :::22 :::0 LISTENING 280 sshd.exe
0x7e64ff58 TCPv4 0.0.0.0:49152 0.0.0.0:0 LISTENING 348 wininit.exe
0x7e651d90 TCPv4 0.0.0.0:135 0.0.0.0:0 LISTENING 684 svchost.exe
0x7e651d90 TCPv6 :::135 :::0 LISTENING 684 svchost.exe
0x7e6523d0 TCPv4 0.0.0.0:49152 0.0.0.0:0 LISTENING 348 wininit.exe
0x7e6523d0 TCPv6 :::49152 :::0 LISTENING 348 wininit.exe
0x7e654190 TCPv4 0.0.0.0:135 0.0.0.0:0 LISTENING 684 svchost.exe
0x7e687460 TCPv4 0.0.0.0:49153 0.0.0.0:0 LISTENING 772 svchost.exe
0x7e68a490 TCPv4 0.0.0.0:49153 0.0.0.0:0 LISTENING 772 svchost.exe
0x7e68a490 TCPv6 :::49153 :::0 LISTENING 772 svchost.exe
0x7e7b1910 TCPv4 0.0.0.0:49154 0.0.0.0:0 LISTENING 848 svchost.exe
0x7e7b1910 TCPv6 :::49154 :::0 LISTENING 848 svchost.exe
0x7e7b5298 TCPv4 0.0.0.0:49154 0.0.0.0:0 LISTENING 848 svchost.exe
0x7e7ccb38 TCPv4 0.0.0.0:49155 0.0.0.0:0 LISTENING 416 services.exe
0x7fc1eb60 UDPv4 0.0.0.0:65320 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fc1eb60 UDPv6 :::65320 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd05b30 UDPv4 0.0.0.0:3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd0e918 UDPv4 0.0.0.0:3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd0e918 UDPv6 :::3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd294a0 UDPv4 0.0.0.0:65319 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd40f50 UDPv4 0.0.0.0:3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd40f50 UDPv6 :::3702 *:* 948 svchost.exe 2020-10-29 13:27:36 UTC+0000
0x7fd6b500 UDPv6 :::1:65317 *:* 3568 svchost.exe 2020-10-29 13:27:35 UTC+0000
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

FIGURE 5 – netscan

On ne voit aucune connexion établie, simplement des *sockets* en écoute et aucun des *owners* ne semble suspect. Néanmoins, la manière dont la question est posée pousse à chercher plus loin, et on trouve rapidement le *plugin* `ndispkts` qui pourrait être utile pour trouver des artefacts de connexions passées.



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II132x25
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✗ git clone git@github.com:bridgeythegeek/ndispktscan.git /opt
fatal: destination path '/opt' already exists and is not an empty directory.
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✗ git clone git@github.com:bridgeythegeek/ndispktscan.git /opt/ndispktscan
Cloning into '/opt/ndispktscan'...
Enter passphrase for key '/root/.ssh/id_rsa':
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 41 (delta 12), reused 41 (delta 12), pack-reused 0
Receiving objects: 100% (41/41), 15.67 KiB | 5.22 MiB/s, done.
Resolving deltas: 100% (12/12), done.
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✗ python2 /opt/volatility/vol.py --plugins=/opt/ndispktscan --profile=Win7SP0x86
-f mem.elf ndispktscan
Volatility Foundation Volatility Framework 2.6.1
Offset (V) Source MAC      Destination MAC  Prot Source IP      Destination IP
SPort DPort  Flags
-----
ERROR : volatility.debug : No MAC addresses found.
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✗
```

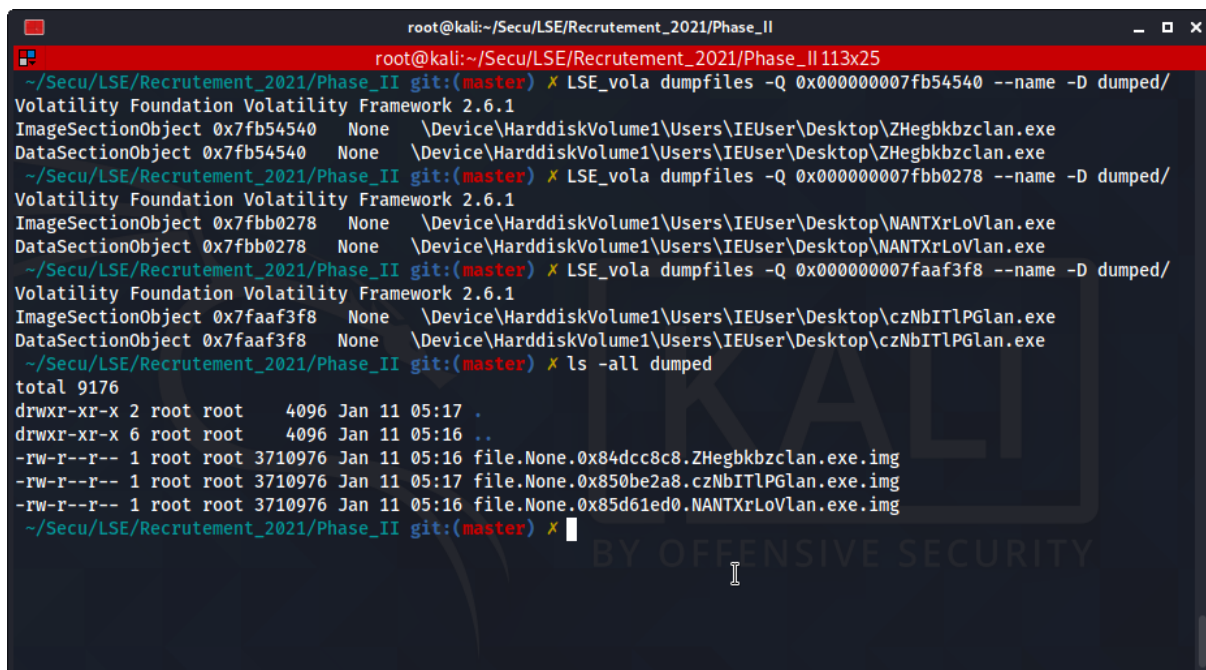
FIGURE 6 – ndispktscan

Sans documentation solide sur le *plugin*, on finit par passer à autre chose en considérant qu'il n'y pas de connexion avec l'extérieur. D'ailleurs, tous les *malwares* ne nécessitent pas une connexion avec extérieure, comme on le verra juste après, et il suffit souvent que la victime fasse partie d'un réseau local pour que le virus se répande.

## 4 Analyse du *malware*

*Quel est en fait le malware qui a infecté la machine ?*

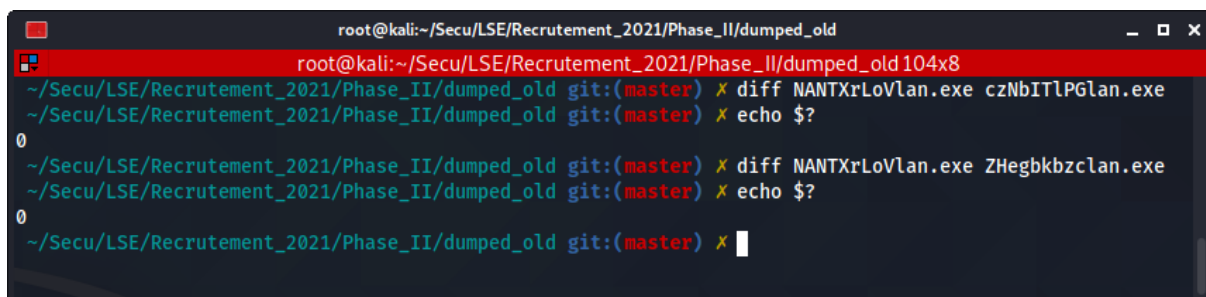
La première chose à faire est d'extraire les 3 fichiers suspects avec dumpfiles.



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 113x25
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola dumpfiles -Q 0x000000007fb54540 --name -D dumped/
Volatility Foundation Volatility Framework 2.6.1
ImageSectionObject 0x7fb54540 None \Device\HarddiskVolume1\Users\IEUser\Desktop\ZHegbkbzclan.exe
DataSectionObject 0x7fb54540 None \Device\HarddiskVolume1\Users\IEUser\Desktop\ZHegbkbzclan.exe
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola dumpfiles -Q 0x000000007fbb0278 --name -D dumped/
Volatility Foundation Volatility Framework 2.6.1
ImageSectionObject 0x7fbb0278 None \Device\HarddiskVolume1\Users\IEUser\Desktop\NANTXrLoVlan.exe
DataSectionObject 0x7fbb0278 None \Device\HarddiskVolume1\Users\IEUser\Desktop\NANTXrLoVlan.exe
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola dumpfiles -Q 0x000000007faaf3f8 --name -D dumped/
Volatility Foundation Volatility Framework 2.6.1
ImageSectionObject 0x7faaf3f8 None \Device\HarddiskVolume1\Users\IEUser\Desktop\czNbITLPglan.exe
DataSectionObject 0x7faaf3f8 None \Device\HarddiskVolume1\Users\IEUser\Desktop\czNbITLPglan.exe
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ ls -all dumped
total 9176
drwxr-xr-x 2 root root 4096 Jan 11 05:17 .
drwxr-xr-x 6 root root 4096 Jan 11 05:16 ..
-rw-r--r-- 1 root root 3710976 Jan 11 05:16 file.None.0x84dcc8c8.ZHegbkbzclan.exe.img
-rw-r--r-- 1 root root 3710976 Jan 11 05:17 file.None.0x850be2a8.czNbITLPglan.exe.img
-rw-r--r-- 1 root root 3710976 Jan 11 05:16 file.None.0x85d61ed0.NANTXrLoVlan.exe.img
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

FIGURE 7 – dumpfiles

Les trois font la même taille. On les renomme pour plus de lisibilité et on vérifie qu'il s'agit réellement de 3 programmes distincts.



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old 104x8
~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old git:(master) ✕ diff NANTXrLoVlan.exe czNbITLPglan.exe
~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old git:(master) ✕ echo $?
0
~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old git:(master) ✕ diff NANTXrLoVlan.exe ZHegbkbzclan.exe
~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old git:(master) ✕ echo $?
0
~/Secu/LSE/Recrutement_2021/Phase_II/dumped_old git:(master) ✕
```

FIGURE 8 – diff

Il s'agit en fait du même *malware* que VirusTotal identifie bien comme une menace.



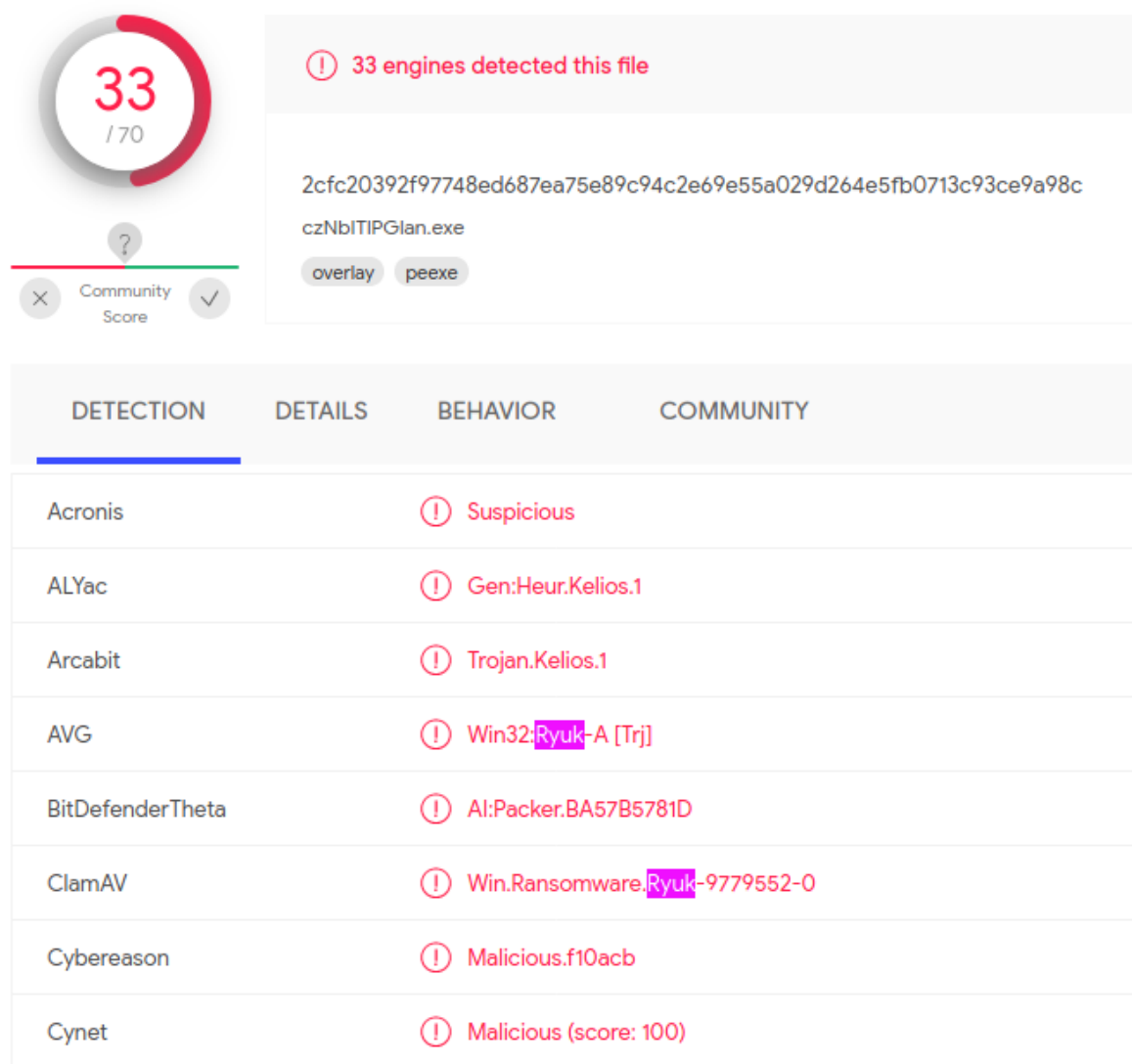


FIGURE 9 – VirusTotal sur le programme suspect

On voit une mention du *ransomware* Ryuk. Après recherches on tombe sur plusieurs articles dont un rapport de l'ANSSI qui explique comment fonctionne ce *malware*. Celui-ci résume également très bien l'attaque : RedCanary, et celui-ci s'occupe de *reverse* la version originale : SecurityLiterate. S'il est inutile de paraphraser tous les articles qui décrivent parfaitement son fonctionnement, il faut comprendre à quelle étape de l'attaque en était le logiciel au moment du *dump*.

On comprend rapidement que les trois *process* trouvés sont en fait des copies du *malware* original afin d'accélérer le chiffrement. Ces fichiers sont en effet identiques au logiciel parent à quelques sections près qui doivent servir à partager le chiffrement des fichiers et bien entendu à la démultiplication du parent. Il est d'ailleurs intéressant de noter que normalement, Ryuk est exécuté par un fichier *Ryuk.exe*, et non *r.exe*, qui est le nom d'un exécutable Windows à une majuscule près, responsable d'une interface graphique. On peut supposer qu'il s'agit d'une variante afin de passer sous les radars ou bien d'une simple coïncidence, puisque le *r* n'est pas en majuscule.

Le *ransomware* liste les fichiers auxquels il a accès et exécute certaines commandes afin, entre autres, d'être sûr d'avoir les permissions nécessaires et de ne pas être repéré par des systèmes de sécurité. C'est d'ailleurs en trouvant certaines de ces commandes pendant la recherche de DLL malveillantes qu'on confirme que Ryuk est passé à l'attaque :

```

root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X LSE_vola dlllist | grep -i command | tail -n 30 | grep -i 'command'
Volatility Foundation Volatility Framework 2.6.1
Command line : C:\Windows\System32\spoolsv.exe
Command line : C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork
Command line : "taskhost.exe"
Command line : "C:\Windows\System32\VBoxTray.exe"
Command line : C:\Windows\system32\vmicsvc.exe -feature Heartbeat
Command line : C:\Windows\system32\vmicsvc.exe -feature KvpExchange
Command line : C:\Windows\system32\vmicsvc.exe -feature Shutdown
Command line : C:\Windows\system32\vmicsvc.exe -feature TimeSync
Command line : C:\Windows\system32\vmicsvc.exe -feature VSS
Command line : C:\Windows\System32\svchost.exe -k utcsvc
Command line : "C:\Program Files\OpenSSH\bin\cygrunsrv.exe"
Command line : \??\C:\Windows\system32\conhost.exe "-285056441-276053921595353925-2049898454631478565-1375467437-47667982
81562157206
Command line : "C:\Program Files\OpenSSH\usr\sbin\sshd.exe"
Command line : C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted
Command line : C:\Windows\system32\SearchIndexer.exe /Embedding
Command line : C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation
Command line : C:\Windows\System32\svchost.exe -k secsvcs
Command line : "C:\Users\IEUser\Desktop\r.exe"
Command line : "C:\Windows\system32\SearchProtocolHost.exe" Global\UsGthrFltPipeMssGthrPipe8_Global\UsGthrCtrlFltPipeMss
GthrPipe8 1 -2147483646 "Software\Microsoft\Windows Search" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT; MS Search 4.0
Robot)" "C:\ProgramData\Microsoft\Search\Data\Temp\usgthrsvc" "DownLevelDaemon"
Command line : "C:\Users\IEUser\Desktop\NANTXrLoVlan.exe" 8 LAN
Command line : "C:\Windows\system32\SearchFilterHost.exe" 0 508 512 520 65536 516
Command line : "C:\Users\IEUser\Desktop\ZHegbkbzclan.exe" 8 LAN
Command line : "C:\Users\IEUser\Desktop\czNbITLPLan.exe" 8 LAN
Command line : icacls "C:\*" /grant Everyone:F /T /C /Q
Command line : \??\C:\Windows\system32\conhost.exe "902039631-1893777931281769700-136576645415903594911182552394-19340987
82750026892
Command line : "C:\Windows\system32\taskmgr.exe" /4
Command line : "C:\Windows\System32\net.exe" stop "vmickvpexchange" /y
Command line : \??\C:\Windows\system32\conhost.exe "713382535-346359522104223265-1953066377-1508232602-204626567518264816
8420243260
Command line : "C:\Windows\System32\net.exe" stop "samss" /y
Command line : \??\C:\Windows\system32\conhost.exe "1231024694-1963916851-6063226491671813406-237279456199244797398898317
41448566712
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X

```

FIGURE 10 – dlllist

Ryuk a par exemple arrêté le *Security Account Manager* **samss** pour désactiver les alarmes qui pourraient envoyer une alerte à un SIEM (*System Information and Event Manager*). Il a également donné les permissions d'accès et de modifications à tous les fichiers du disque C au groupe **Everyone** avec **icacls** comme on le soupçonnait dès le début afin d'être sûr de pouvoir chiffrer tous les fichiers. On retrouve également toutes les autres commandes dans les *strings* du *dump*.

On peut néanmoins faire une petite note à part sur la présence des fichiers **confax.exe** et **LBTServ.dll** présents sur le bureau également.

```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ LSE_vola filescan | grep -Ei 'confax|lbtserver'
Volatility Foundation Volatility Framework 2.6.1
0x000000007e260cb8 7 0 R--rwd \Device\HarddiskVolume1\Users\IEUser\Desktop\LBTserv.dll
0x000000007e5beec8 7 0 R--r-d \Device\HarddiskVolume1\Users\IEUser\Desktop\confax.exe
0x000000007fa613a0 7 0 R--r-d \Device\HarddiskVolume1\Users\IEUser\Desktop\LBTserv_unpack.dll
0x000000007fd37f38 7 0 R--rwd \Device\HarddiskVolume1\Users\IEUser\Desktop\LBTserv.dll
0x000000007fd95370 7 0 R--rwd \Device\HarddiskVolume1\Users\IEUser\Desktop\confax.exe
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕
```

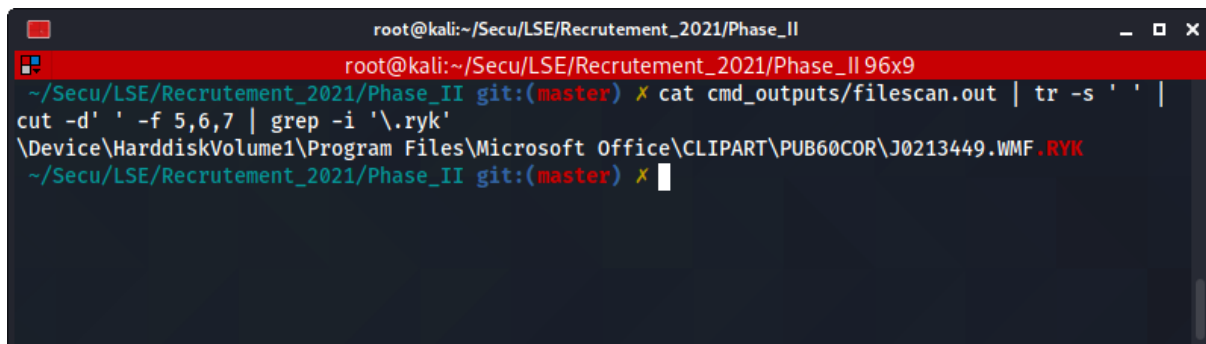
FIGURE 11 – filescan

En effet ils correspondent exactement aux charges utiles de cette attaque du groupe criminel APT17 sur le gouvernement du Kirghizistan et qui profitait du contexte de crise actuel lié au Covid-19. En outre comme on obtient des fichiers vides lors de l'extraction, et qu'ils n'apparaissent pas dans les *process* en cours, on écarte vite la piste toutefois curieuse.

## 5 Etat de l'attaque

*Qu'était-il en train de faire au moment où le dump a été fait ?*

En analysant les fichiers présents sur le système et accessibles avec `filescan`, on remarque qu'il n'y a qu'un seul fichier chiffré.

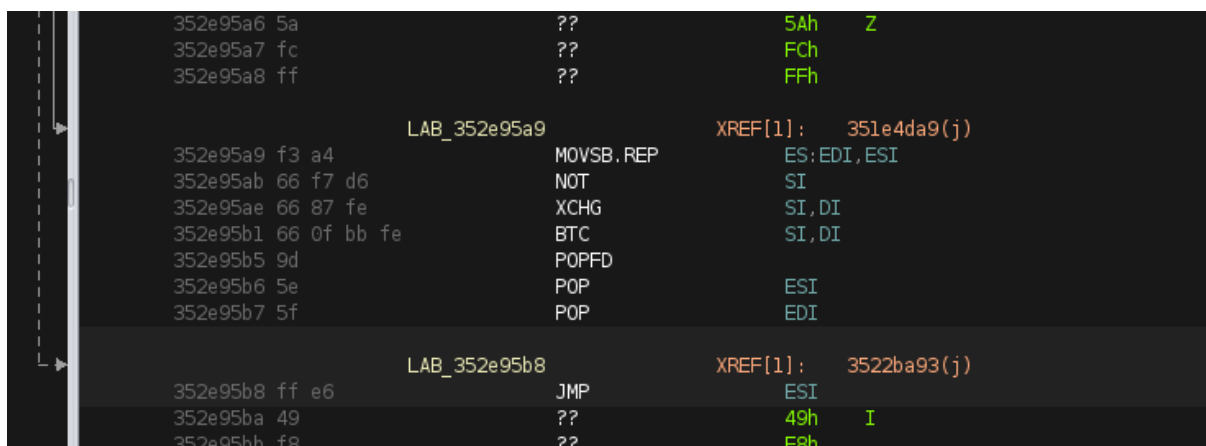


```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 96x9
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X cat cmd_outputs/filescan.out | tr -s ' ' |
cut -d' ' -f 5,6,7 | grep -i '\.ryk'
\\Device\\HarddiskVolume1\\Program Files\\Microsoft Office\\CLIPART\\PUB60COR\\J0213449.WMF.RYK
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X
```

FIGURE 12 – `filescan | grep`

Il semble impossible d'extraire ce fichier comme il arrive parfois lorsqu'un programme ferme le fichier qu'il utilisait par exemple (plus d'informations [ici](#)), mais savoir qu'il existe suffit pour comprendre que le *dump* s'est fait pendant le chiffrement de Ryuk.

Afin de trouver l'avancement de l'attaque et de trouver si celle-ci s'est arrêtée seule ou si le dump a été fait juste après le premier chiffrement, on extrait le *ransomware* `r.exe` pour l'analyser. Le *reverse* du logiciel en statique avec Ghidra et Binary Ninja est très compliqué à cause d'instructions comme `JMP ESI` qui nécessitent un *debug* dynamique pour suivre le flot d'exécution. De plus, il semble se modifier lui-même afin de cacher ses instructions et de rendre pénible son analyse.



```
352e95a6 5a      ??      5Ah     Z
352e95a7 fc      ??      FCh
352e95a8 ff      ??      FFh

LAB_352e95a9 XREF[1]: 351e4da9(j)
352e95a9 f3 a4    MOVSB, REP    ES:EDI,ESI
352e95ab 66 f7 d6  NOT      SI
352e95ae 66 87 fe  XCHG     SI,DI
352e95b1 66 0f bb fe BTC      SI,DI
352e95b5 9d      POPFD
352e95b6 5e      POP      ESI
352e95b7 5f      POP      EDI

LAB_352e95b8 XREF[1]: 3522ba93(j)
352e95b8 ff e6    JMP      ESI
352e95ba 49      ??      49h     I
352e95bb f8      ??      F8h
```

FIGURE 13 – ghidra

On décide donc de lancer le *ransomware* sur une machine virtuelle sous Windows 7 (32 bits) afin de pouvoir non seulement *debug* le programme pour comprendre pourquoi le chiffrement

s'est arrêté après avoir chiffré un seul fichier mais également pour obtenir l'*email* présent dans la note laissée avec les fichiers chiffrés.

On commence par lancer un serveur sur la machine virtuelle qui contient le *malware* :

---

```
$ python -m SimpleHTTPServer 4444
```

---

On peut ensuite aller chercher les fichiers intéressants depuis **Windows** à l'adresse du serveur avec l'IP de la machine hôte et un *port-forwarding* sur le port 4444 de la machine où le serveur a été lancé. Dès le téléchargement, le logiciel malveillant est détecté.

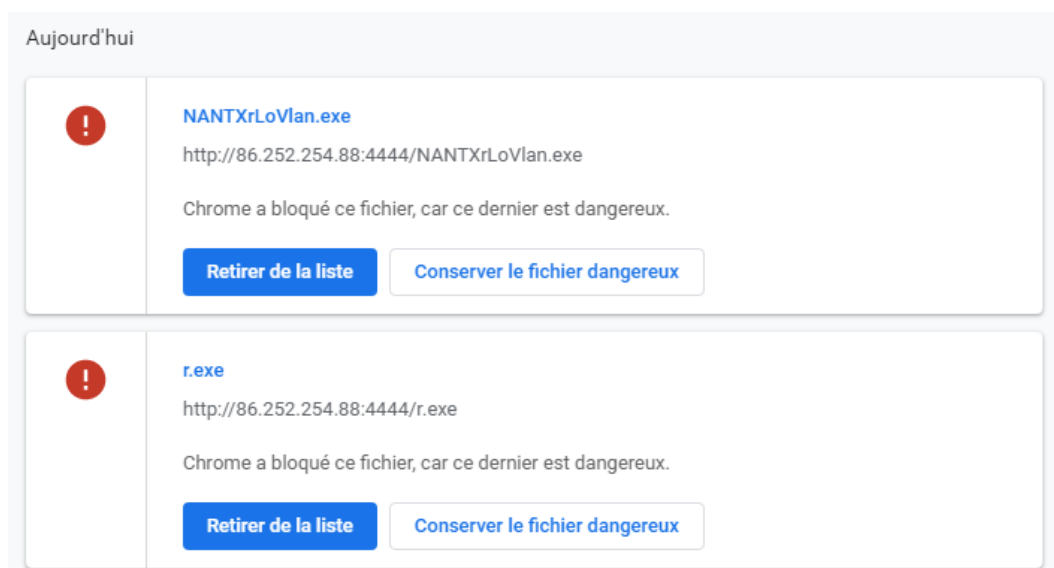


FIGURE 14 – détection par chrome

Il l'est également lors de l'exécution (ici sur **Windows 10** avec l'antivirus par défaut, celui de **Windows 7** étant moins visuel).

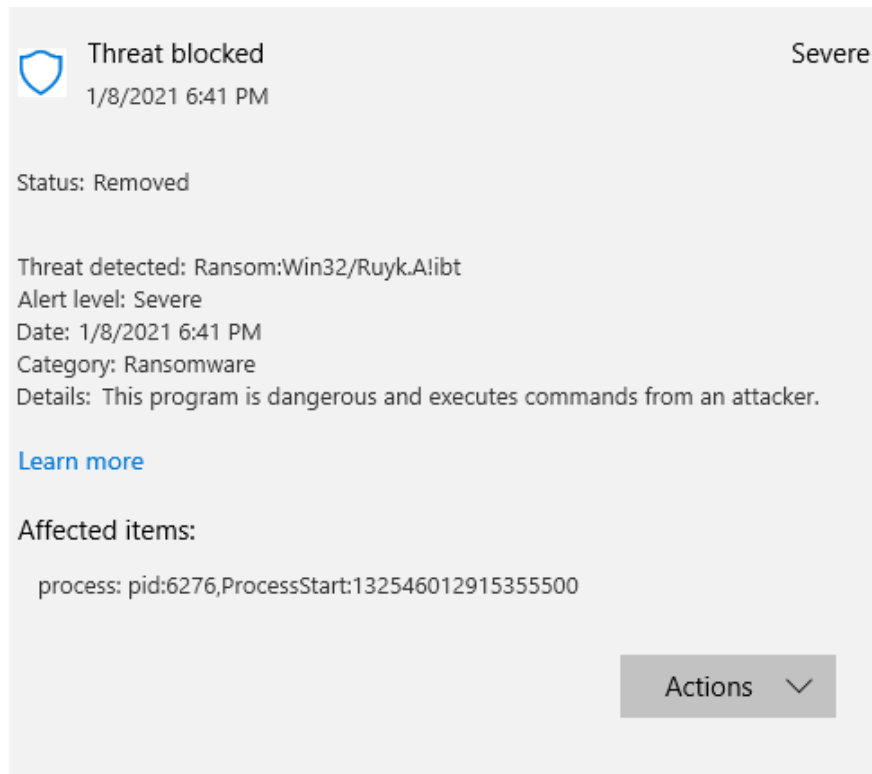


FIGURE 15 – détection par Windows

Néanmoins l'exécution du *malware* extrait semble échouer puisque rien ne se passe et que le *process* disparaît après quelques secondes. On essaye donc de *debug* le programme pour non seulement le faire fonctionner mais également pour avoir une piste sur la question suivante : *Ryuk* étant connu pour être un *ransomware* très rapide pour chiffrer grâce aux multiples copies créées (trouvées ici avant même l'original), comment et pourquoi s'est-il arrêté après n'avoir chiffré qu'un fichier ? Est-ce que le *dump* a simplement été fait à ce moment précis ou est-ce que le *ransomware* a été manuellement mis en pause, avec par exemple un des *debuggers* comme *Immunity Debugger* qu'on trouve sur le bureau.

Malheureusement, une *segfault* sur un problème d'accès semble être à l'origine du problème et même en tentant d'avoir le même environnement après avoir effectué les mêmes commandes que le *malware*, on ne réussit pas à le lancer, même si on peut facilement soupçonner un problème de permissions à cause d'un *grant* avec *icacis* mal simulé.



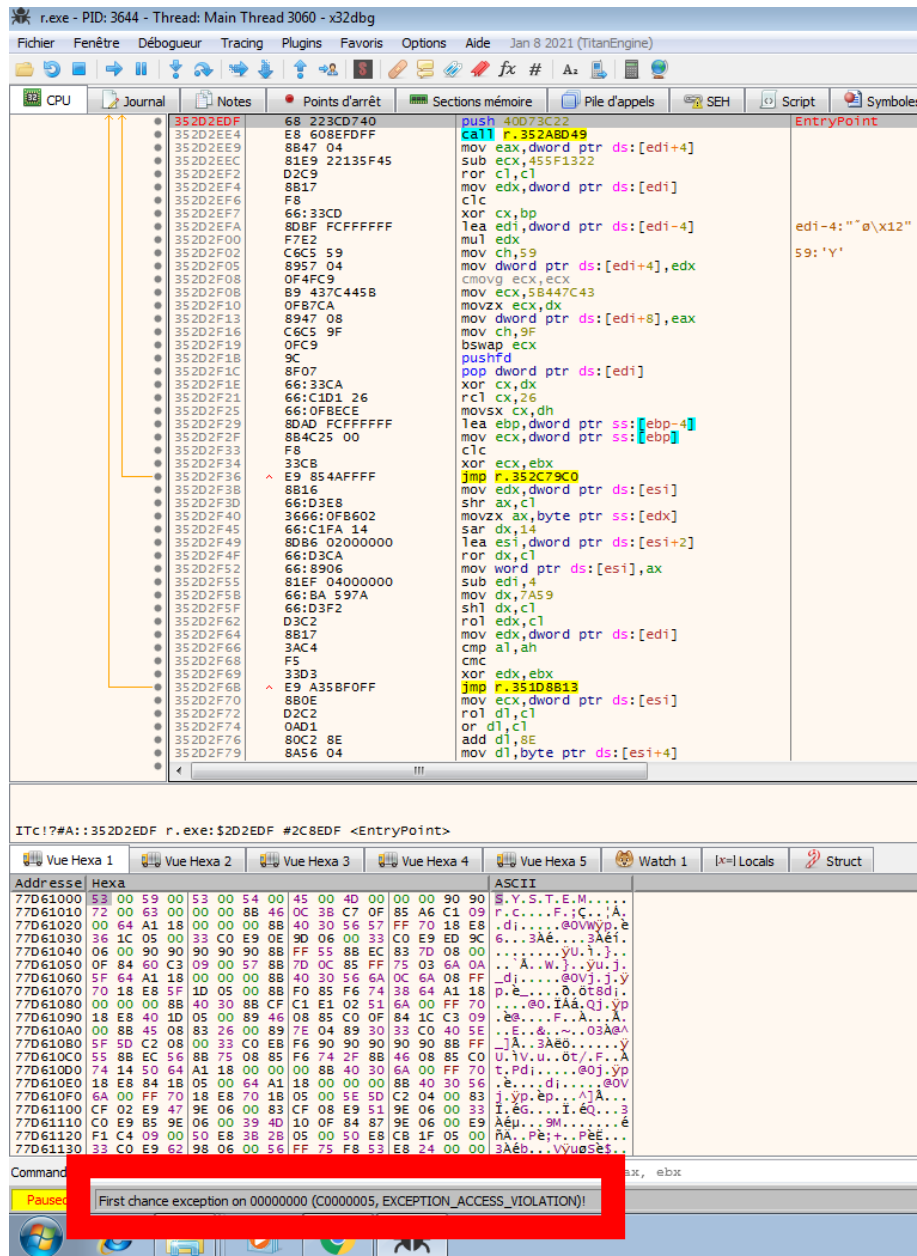


FIGURE 16 – *segfault* dans x32dbg

C'est avec cette question irrésolue qu'on peut simplement conclure que le *malware* était en train de chiffrer les fichiers du système lorsque le *dump* a été fait.

## 6 Identification de l'attaquant

*Quel est l'email de l'attaquant et à quoi sert-il ?*

Comme lancer le *ransomware* a échoué, il faut trouver un autre moyen d'obtenir la note. On commence par `strings | grep` sur le nom de la note `RyukReadMe` avec les *offsets*, afin de trouver l'adresse physique de la chaîne. On récupère en parallèle une cartographie mémoire du *dump* avec la commande `memmap` de *volatility*. On effectue ensuite plusieurs opérations sur l'*output* en suivant ce [lien](#) pour trouver le PID du *process* qui contient cette chaîne avec son adresse pour enfin extraire la mémoire de ce *process* et ainsi récupérer son état.

```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 114x53
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X strings -f -t x mem.elf > cmd_outputs/stringsWithOffsets.out
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X LSE_vola memmap > cmd_outputs/memmap.out 2>/dev/null
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X cat cmd_outputs/stringsWithOffsets.out | grep -i 'ryukreadme'
mem.elf: 1e4d4c94 RyukReadMe.html
mem.elf: 33767734 RyukReadMe.html
mem.elf: 3e54855e RyukReadMe.html
mem.elf: 3e5485d6 RyukReadMe.html
mem.elf: 3e54864e RyukReadMe.html
mem.elf: 3e5486c6 RyukReadMe.html
mem.elf: 3e54873e RyukReadMe.html
mem.elf: 3e5487b6 RyukReadMe.html
mem.elf: 3e54882e RyukReadMe.html
mem.elf: 3e5488a6 RyukReadMe.html
mem.elf: 3e54891e RyukReadMe.html
mem.elf: 3e548996 RyukReadMe.html
mem.elf: 3e548a0e RyukReadMe.html
mem.elf: 3e548b76 RyukReadMe.html
mem.elf: 3e548bee RyukReadMe.html
mem.elf: 3e548cde RyukReadMe.html
mem.elf: 3e548d56 RyukReadMe.html
mem.elf: 3e548dce RyukReadMe.html
mem.elf: 3e548e46 RyukReadMe.html
mem.elf: 3e548ebe RyukReadMe.html
mem.elf: 3e548f36 RyukReadMe.html
mem.elf: 3e548fae RyukReadMe.html
mem.elf: 3e549026 RyukReadMe.html
mem.elf: 3e54909e RyukReadMe.html
mem.elf: 3e549116 RyukReadMe.html
mem.elf: 3e54918e RyukReadMe.html
mem.elf: 3e549206 RyukReadMe.html
mem.elf: 3e54927e RyukReadMe.html
mem.elf: 3e5492f6 RyukReadMe.html
mem.elf: 3e54936e RyukReadMe.html
mem.elf: 574a9aa4 RyukReadMe.html
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X cd cmd_outputs
~/Secu/LSE/Recrutement_2021/Phase_II/cmd_outputs git:(master) X cat memmap.out | awk -F' ' '{print $2,$3,$4,$5}' >
memmap_without_V0.out
~/Secu/LSE/Recrutement_2021/Phase_II/cmd_outputs git:(master) X perl -lpe '$=hex' memmap_without_V0.out | paste -
d" " - memmap_without_V0.out | sort -n | cut -d" " -f 2- > memmap_without_V0_sorted.out
~/Secu/LSE/Recrutement_2021/Phase_II/cmd_outputs git:(master) X cat memmap_without_V0_sorted.out | grep '^0x574a'
0x574a7000 0x1000 0x5b6a000
0x574ab000 0x1000 0x38e5000
0x574af000 0x1000 0x4254000
~/Secu/LSE/Recrutement_2021/Phase_II/cmd_outputs git:(master) X cat memmap.out | grep 0x574a7000 -B 100000 | grep
pid
r.exe pid: 3224
~/Secu/LSE/Recrutement_2021/Phase_II/cmd_outputs git:(master) X ..
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X mkdir outdir
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X LSE_vola memdump -p 3224 --dump-dir=outdir
Volatility Foundation Volatility Framework 2.6.1
*****
Writing r.exe [ 3224] to 3224.dmp
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) X
```

FIGURE 17 – memmap, mumdumpp

Une fois le *process* récupéré, il est simple d'obtenir l'*email* de l'attaquant car tous les fichiers RyukReadMe.htm ont la même forme :

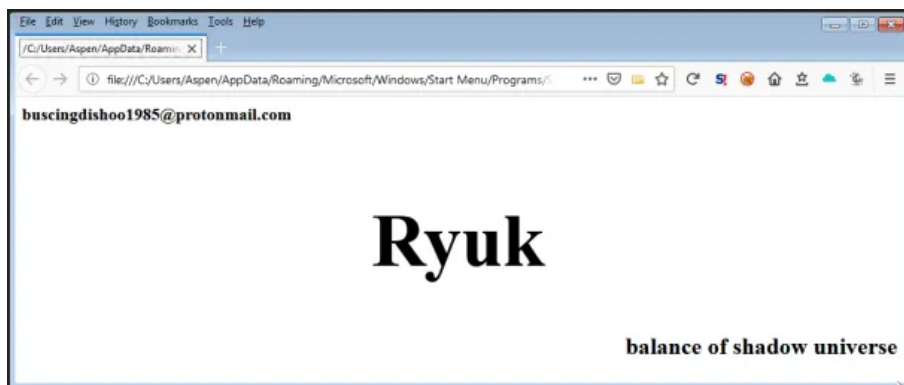


FIGURE 18 – RyukReadMe.htm - [source](#)

On `grep` la signature sur un `strings` de l'ensemble du *dump* et on vérifie qu'on trouve la même adresse dans le *dump* du *process* de Ryuk, les chaînes de caractères de l'exécutable extrait *r.exe* étant modifiées pour éviter une détection facile des systèmes de sécurité qui connaissent les signatures de nombreux logiciels malveillants.

```

root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 115x36

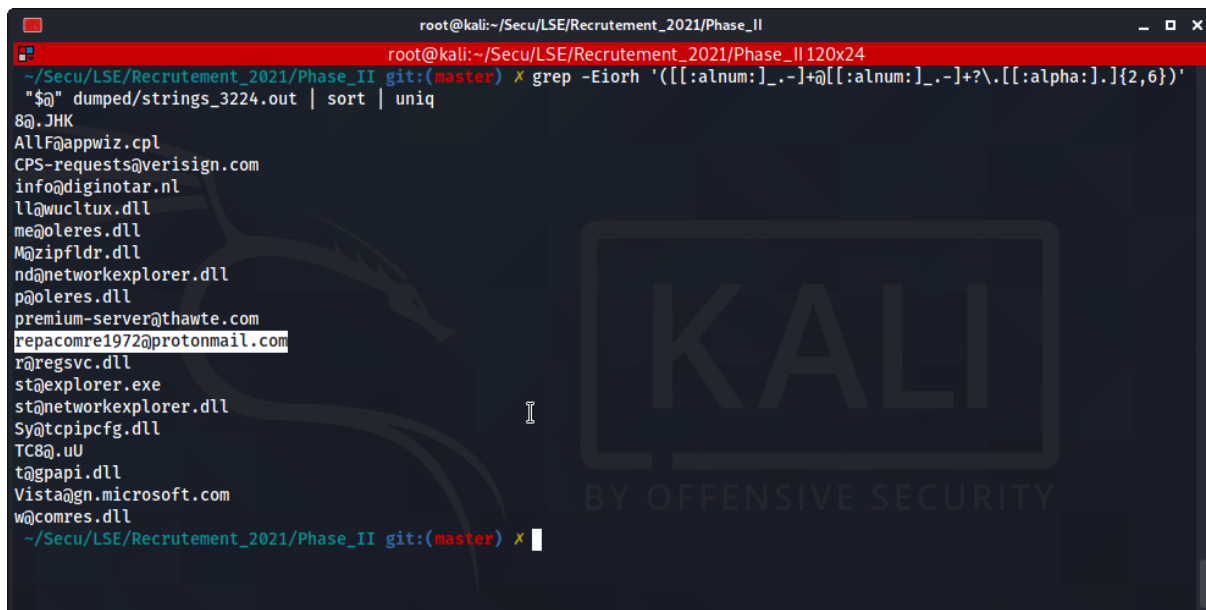
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ strings mem.elf | grep 'shadow universe' -C 2
\pard\ri-1782\sa200\sl276\slmult1\qr      \par
\par
\pard\ri-1782\sa200\sl240\slmult1\qr\fs34 balance of shadow universe\b0\f1\fs22\par
CryptEncrypt
\users\Public\
--
\pard\ri-1782\sa200\sl276\slmult1\qr      \par
\par
\pard\ri-1782\sa200\sl240\slmult1\qr\fs34 balance of shadow universe\b0\f1\fs22\par
CryptEncrypt
\users\Public\
--
repacomre1972@protonmail.com

balance of shadow universe
Ryuk
AbstractTableViewTextColor=#000000
--
\pard\ri-1782\sa200\sl276\slmult1\qr      \par
\par
\pard\ri-1782\sa200\sl240\slmult1\qr\fs34 balance of shadow universe\b0\f1\fs22\par
CryptEncrypt
\users\Public\
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕ strings dumped/3224.dmp | grep 'repacomre1972@protonmail.com'
<html><body><p style="font-weight:bold;font-size:127%;top:0;left:0;"> repacomre1972@protonmail.com
<br>
</p><p style="pos
ition:absolute;bottom:0;right:1%;font-weight:bold;font-size:171%">6#98;6#97;6#108;6#97;6#110;6#99;6#101;6#32;6#111;
6#102;6#32;6#115;6#104;6#97;6#100;6#111;6#119;6#32;6#117;6#110;6#105;6#118;6#101;6#114;6#115;6#101;</p><div style="
font-size: 551%;font-weight:bold;width:51%;height:51%;overflow:auto;margin:auto;position:absolute;top:36%;left:41%;
">6#82;6#121;6#117;6#107;</div></body></html>
repacomre1972@protonmail.com
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) ✕

```

FIGURE 19 – strings | grep

C'est d'ailleurs la seule adresse qui ressemble au modèle qu'on trouve dans Ryuk dans les articles au sein du *process* :



```
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II
root@kali:~/Secu/LSE/Recrutement_2021/Phase_II 120x24
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) x grep -Eiorh '([[:alnum:]]_[-]+@[[:alnum:]]_[-]+?\.[[:alpha:]]{2,6})'
"$@" dumped/strings_3224.out | sort | uniq
8@.JHK
AllF@appwiz.cpl
CPS-requests@verisign.com
info@diginotar.nl
ll@wucltux.dll
me@oleres.dll
M@zipfldr.dll
nd@networkexplorer.dll
p@oleres.dll
premium-server@thawte.com
repacomre1972@protonmail.com
r@regsvc.dll
st@explorer.exe
st@networkexplorer.dll
Sy@tcpipcfg.dll
TC8@.uU
t@gpapi.dll
Vista@gn.microsoft.com
w@comres.dll
~/Secu/LSE/Recrutement_2021/Phase_II git:(master) x
```

FIGURE 20 – grep

L'*email* de l'attaquant est donc `repacomre1972@protonmail.com`, et il sert à obtenir des informations sur comment payer la rançon en échange du logiciel pour déchiffrer les fichiers.

---

Ce fichier contient 9963 caractères.