Basic Dynamic Analysis Lab Solution and Guide TMPprovider038.dll

SHORT ANSWERS

Any interesting observations from basic static analysis?

The sample appears to be packed and/or obfuscated using a tool called *VMProtect*.

What do you observe this program doing through dynamic analysis?

The malware connects to `fauxnet.mandiant.com` and `flossme.mandiant.com` over port 80 using HTTP `POST` requests.

The malware copies itself to `%TEMP%\TMProvider038.dll` and sets itself for persistence at `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\TmProvider`.

The malware also writes a GUID value to

`"HKCU\Software\Microsoft\Internet Explorer\InternetRegistry\fertger"`.

The malware creates a file `qln.dbx` in the current user's Temp directory with an unknown constant "`044`"

hide01.ir

List any potential host-based indicators of this malware.

The registry value `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\TmProvider`

The registry value `"HKCU\Software\Microsoft\Internet Explorer\InternetRegistry\fertger"`

The copy location of `%TEMP%\TMProvider038.dll`

Creates a file `qln.dbx` in the user's Temp directory

List any potential network-based indicators of this malware.

`fauxnet.mandiant.com:80 /wp08/wp-includes/dtcla.php`

`flossme.mandiant.com:80 /geo/productid.php`

DETAILED ANALYSIS

Any interesting observations from basic static analysis?

Open the file in "*CFF Explorer*". There are suspicious anomalies that suggest packing. There are sections named `.FLARE0` and `.FLARE1`. Five of the sections have "*Raw Size"* of `0` which suggests decoding/decompression at runtime.

Name	Virtual Size	Virtual Address	Raw Size
Byte[8]	Dword	Dword	Dword
.text	000136A7	00001000	00000000
.rdata	00007208	00015000	00000000
.data	00001330	0001D000	00000000
.gfids	000000A8	0001F000	00000000
.FLARE0	000C76CD	00020000	00000000
.FLARE1	001AA5B0	000E8000	001AA600
.reloc	000007F4	00293000	00000800
.rsrc	000001D5	00294000	00000200

hide01.ir

Figure 1: Section headers indicate possible packing

Open the file in *PEiD* and *DIE* to investigate further. Both tools have multiple detection techniques. For *PEiD*, try the "*Hardcore Scan"*. Unfortunately, no packer is detected with *PEiD*. *DIE* has additional rules you can select. Instead of using the "*Detect It Easy*" ruleset, try the "*Nauz File Detector*". When choosing this option, *DIE* successfully detects the *VMProtect* packer.

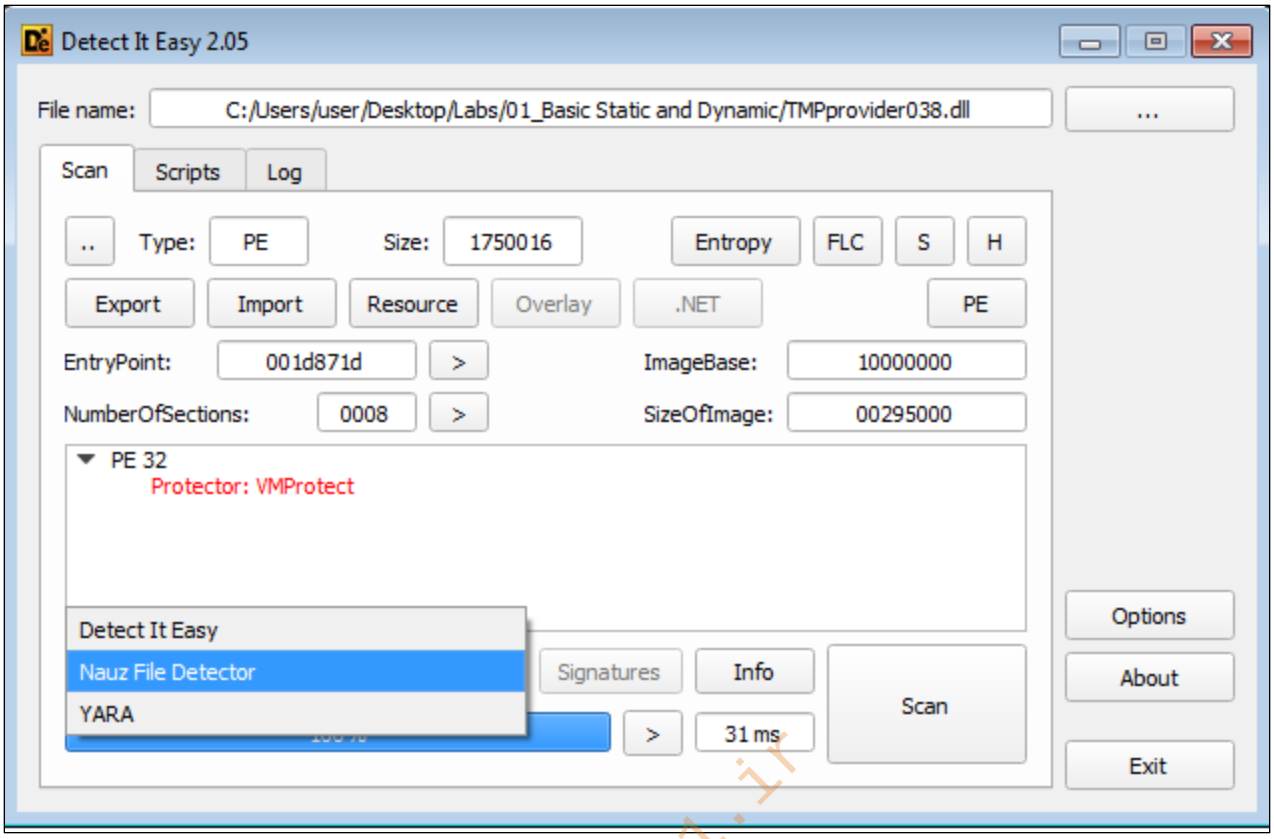

Figure 2: DIE "Nauz File Detector" indicates VMProtect

hide01.ir

Run *FLOSS* ("`floss input_filename -o output_filename`"). There are not many strings. A few strings do stand out and confirm the *DIE* result:

```
VMProtect Software1
VMProtect Software CA
```

Dynamic analysis will be needed to analyze this obfuscated sample, since there is no tool available to easily deal with *VMProtect*.

What do you observe this program doing through dynamic analysis?

Prepare your dynamic analysis environment.

1. Open "*Process Monitor*" to capture Windows events during the malware execution. Stop capture and clear the output, then prepare filters. It is recommended to show *Operations* that include `ProcessCreate`, `RegSetValue`, `WriteFile`, and `SetDispositionInformationFile`. There are many approaches to filtering the events captured by "*Process Monitor*", so experiment with different workflows and use what works for you.

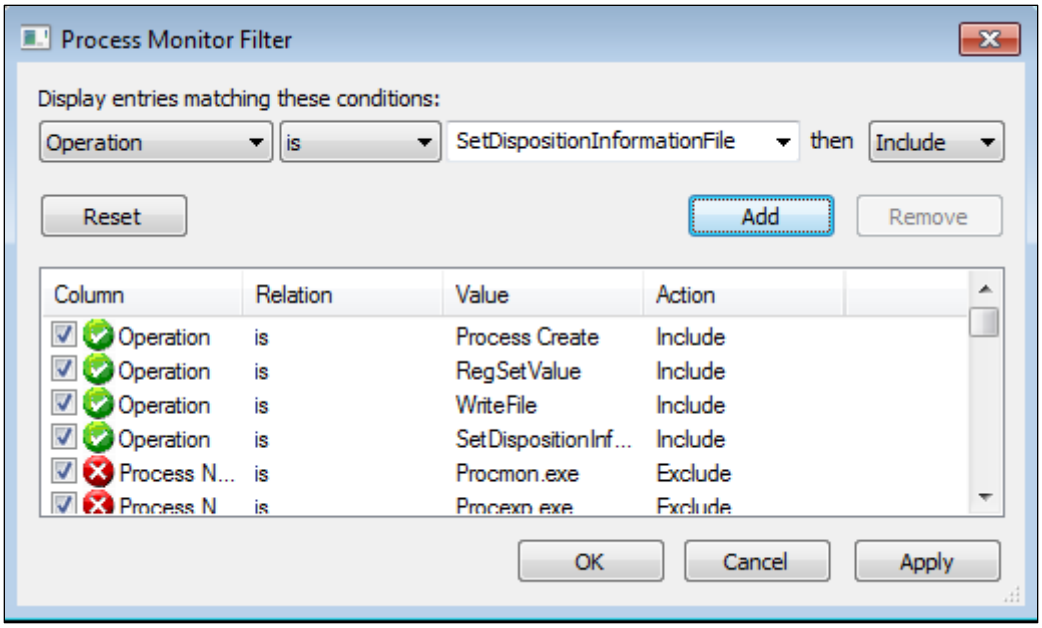

Figure 3: Filter by Operation using "Process Monitor"

2. Open "*Process Explorer*" to observe running processes.

3. Open *FakeNet-NG* to simulate a network connection and capture network behavior. *FakeNet-NG* requires a connected network interface, so make sure you have a network interface set to "*Host Only*". This is configured by default within FLARE VM.

4. Open an Administrator Command Prompt and prepare to run the malware via the command line. This sample is a DLL so use the Windows utility `rundll32.exe`. Open "*CFF Explorer*" and examine the "*Export Directory*" so `rundll32.exe` can run an exported DLL function. The DLL has one export, `RunDllEntry`.

hide01.ir

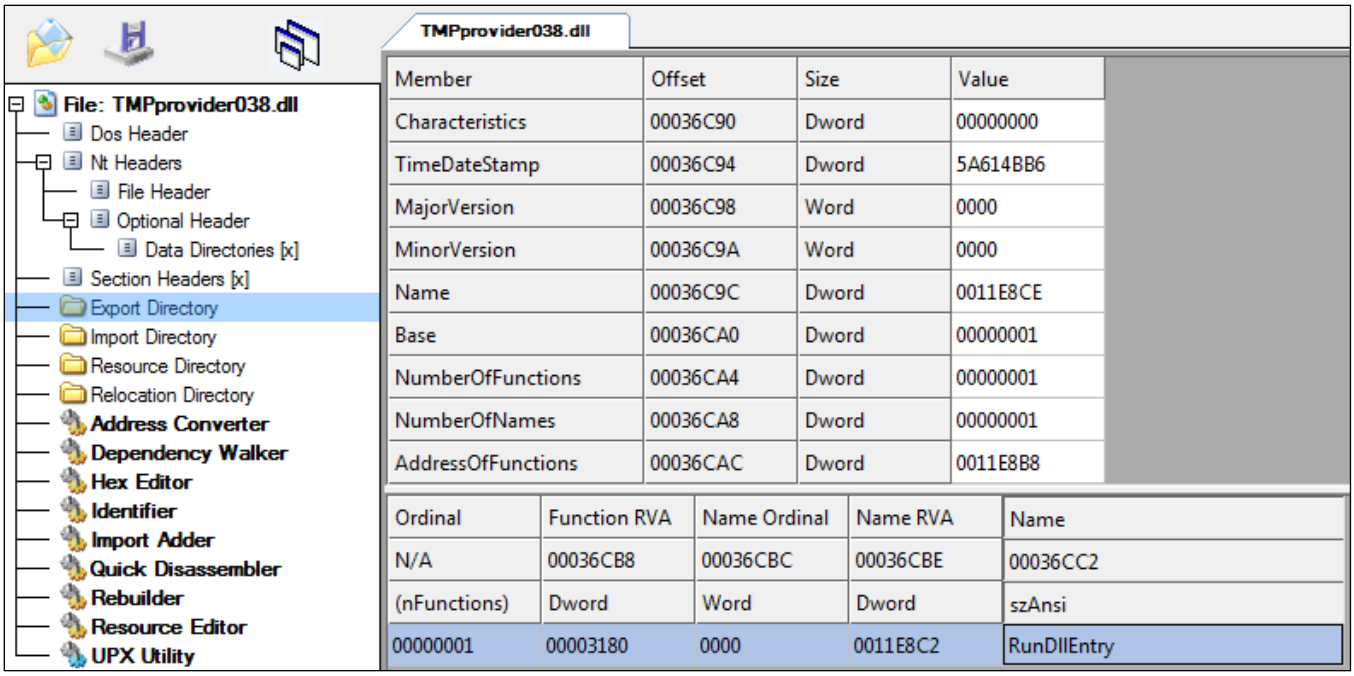

Figure 4: "Export Directory" shows a single export: RunDllEntry

Prepare the text to run the program on the command line, but do not run it until a VM snapshot has been taken.

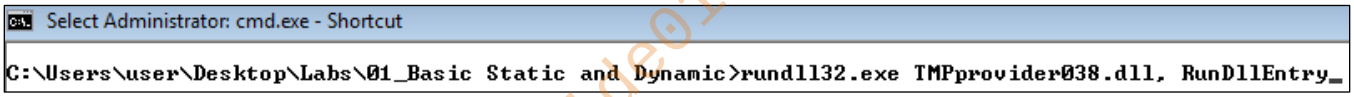

Figure 5: Prepare to run the malware on the command line

5. Take a VM snapshot so the computer state can be restored after analysis is complete.

hide01.ir

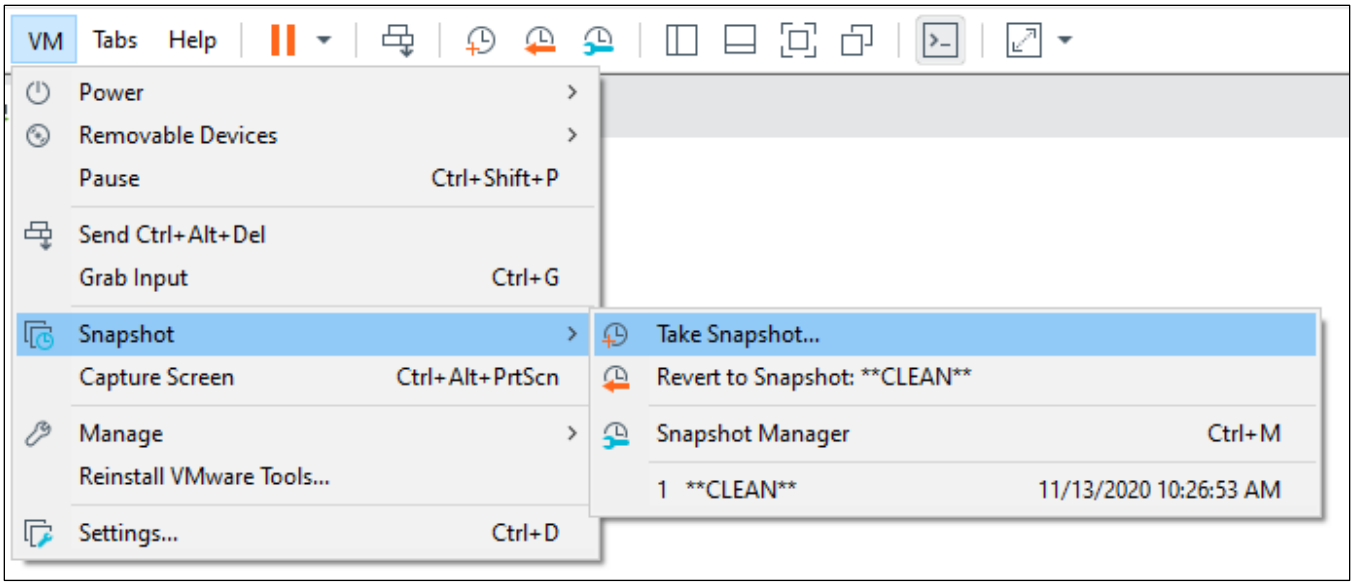

Figure 6: Take a snapshot before running the malware

6. Turn on capturing within "*Process Monitor*" and press Enter on the Command Prompt to run the malware.

Analyze the captured data.

First wait a few seconds for the malware to execute. *FakeNet-NG* continually produces output, so it is advisable to close it shortly after malware behavior is captured, but before the output is filled with unrelated data. Click on the *FakeNet-NG* window and press `CTRL-C` to exit the program. You may need to press it twice. It is recommended to allow *FakeNet-NG* to complete the closing process and exit cleanly rather than forcing it closed – this way *FakeNet-NG* is more likely to restore changes it has made to Windows networking behavior.

Stop the capture in "*Process Monitor*".

FakeNet-NG output can be examined in the output window or by using the packet analysis tool *WireShark*. Sometimes Windows can produce a lot of network traffic and the output window can become filled – in those cases *WireShark* may be better. *FakeNet-NG* produces a packet capture file (`.pcap`) that contains traffic captured by *FakeNet-NG* (both before and after the packet is modified by *FakeNet-NG*). By default, the file is saved in the directory from which *FakeNet-NG* is run. FLARE VM may be configured to save the file in the directory `C:\Users\user\Desktop\fakenet_logs` if *FakeNet-NG* is run from the taskbar. *FakeNet-NG* is also configured to save a file for each HTTP request that is served by the application. Here is an example of what the directory listing can contain:

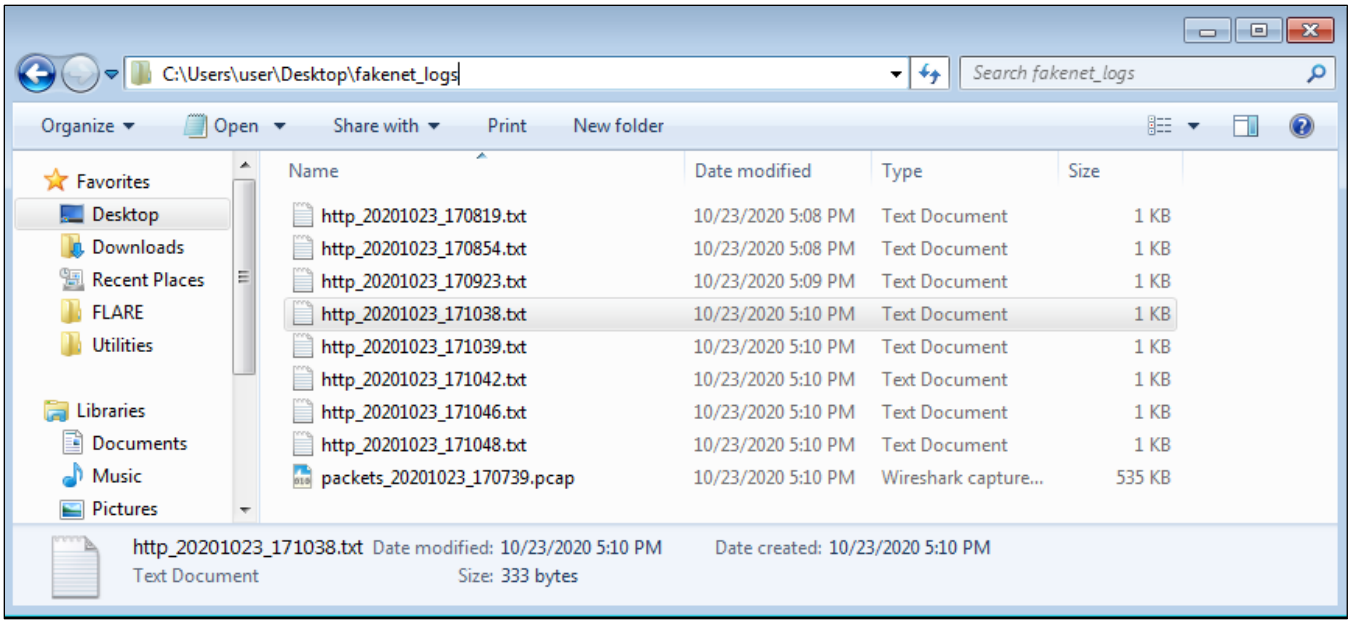

Figure 7: Fakenet-NG creates .txt files for captured HTTP requests

Try opening each of the `.txt` files to see if any HTTP traffic was captured. Most of them will likely be related to benign Windows activity, such as Online Certificate Status Protocol (OCSP) requests. These can be ignored as they are not generated by the malware.

hide01.ir

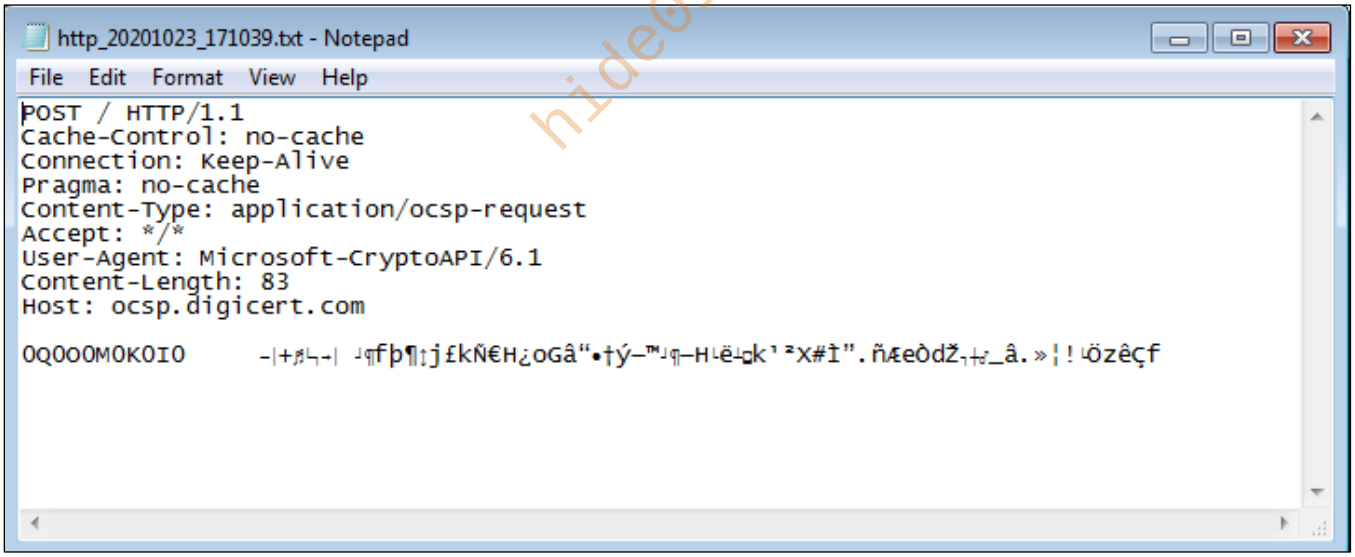

Figure 8: Windows activity such as OCSP creates noise in Fakenet-NG output

FakeNet-NG should have captured HTTP traffic from the malware to `flossme.mandiant.com`.

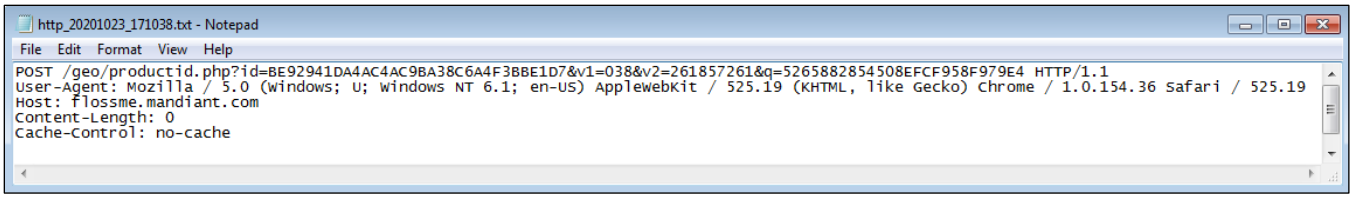

Figure 9: HTTP POST request to C2 as captured by Fakenet-NG

If you are unable to observe HTTP requests this way, open the `.pcap` file in *WireShark*. *WireShark* is a powerful tool and there are many ways to approach traffic analysis. One way to examine HTTP traffic is to navigate to *File* – "*Export Objects" – HTTP*. Here you can see the malware requests grouped together, and you can click them to see the packets in the main window.

hide01.ir

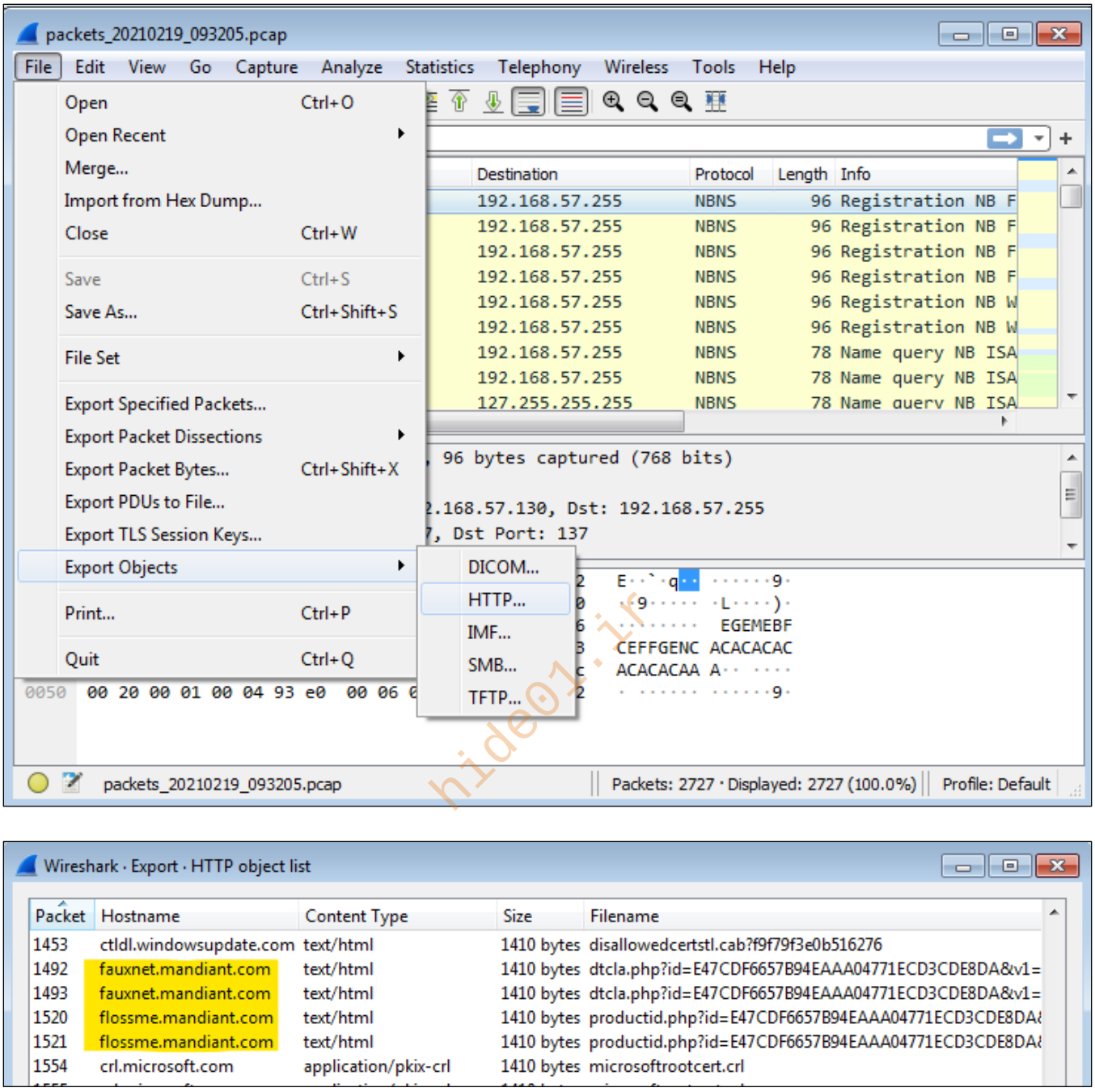

Figure 10: WireShark - export HTTP objects

Once you identify the HTTP packets, right click on one and choose *Follow – "TCP Stream"*. A text output of the requests is displayed.

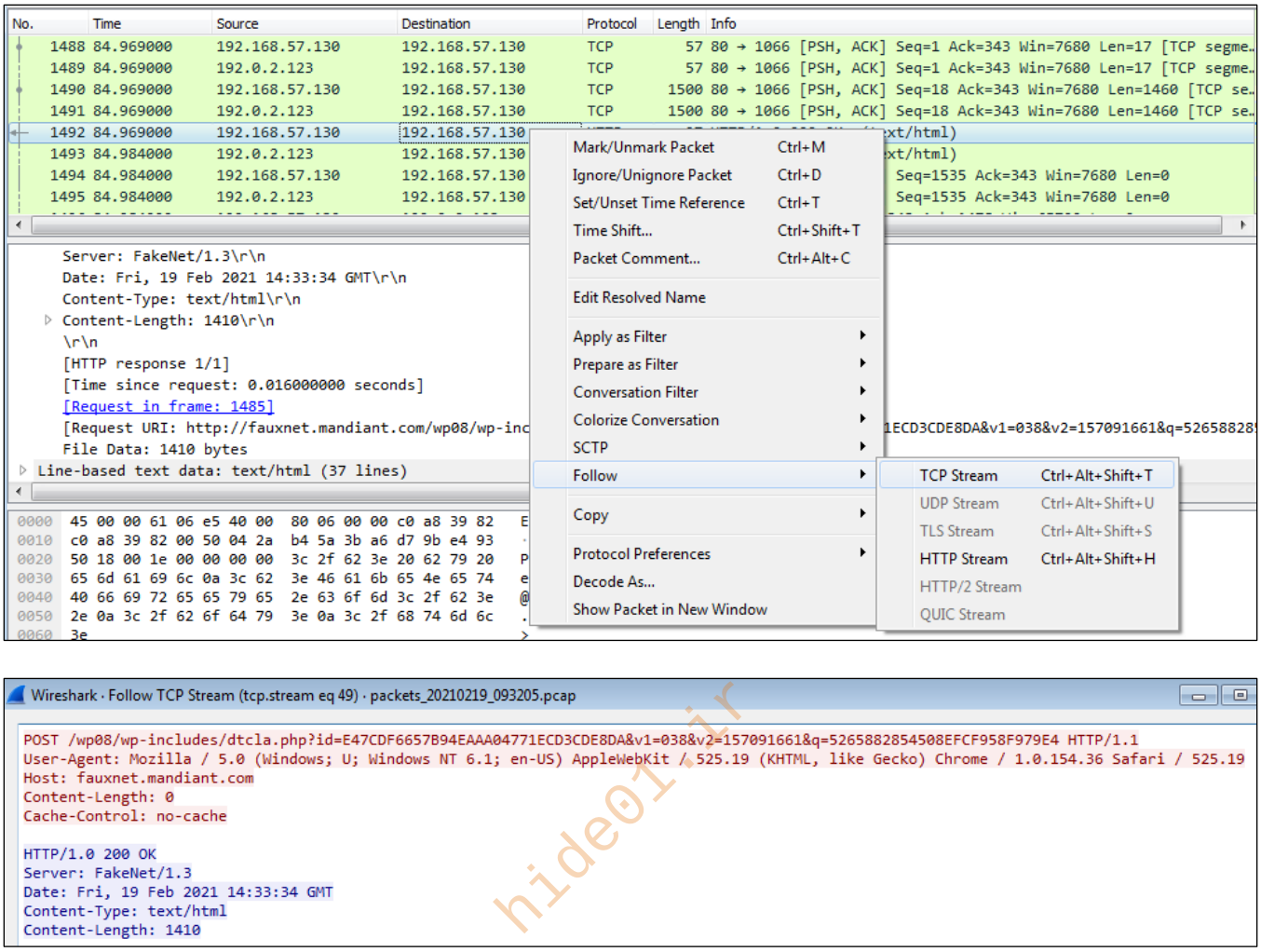

Figure 11: TCP stream displayed in WireShark

Observe that the malware makes two HTTP requests as follows:

```
POST /geo/productid.php?id=BE92941DA4AC4AC9BA38C6A4F3BBE1D7&v1=038&v2=261857261&q=5265882854508EFCF958F979E4 HTTP/1.1
User-Agent: Mozilla / 5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit / 525.19 (KHTML, like Gecko) Chrome / 1.0.154.36 Safari / 525.19
Host: flossme.mandiant.com
Content-Length: 0
Cache-Control: no-cache
```

```
POST /wp08/wp-includes/dtcla.php?id=BE92941DA4AC4AC9BA38C6A4F3BBE1D7&v1=038&v2=261857261&q=5265882854508EFCF958F979E4 HTTP/1.1
User-Agent: Mozilla / 5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit / 525.19 (KHTML, like Gecko) Chrome / 1.0.154.36 Safari / 525.19
Host: fauxnet.mandiant.com
Content-Length: 0
Cache-Control: no-cache
```

Now consider the "*Process Monitor*" output. Start by identifying the start of the process – you can ignore prior events. Although the malware file is named `TMPprovider038.dll`, the process of interest is `rundll32.exe`. `rundll32.exe` is responsible for loading the malicious DLL, which then runs within the context of the `rundll32.exe` process. Once you identify the start of the process, right click on it and choose "*Highlight 'rundll32.exe'*".

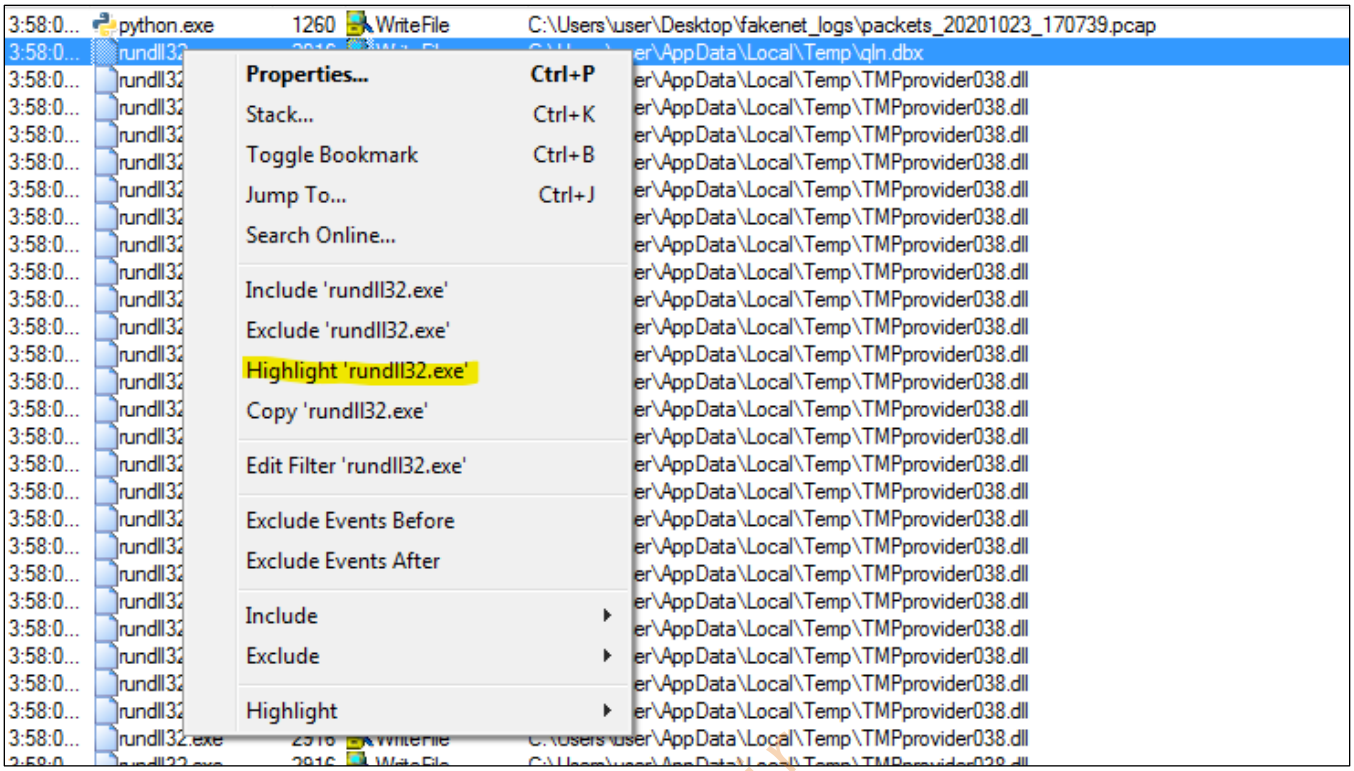

Figure 12: Highlight events produced by process named rundll32.exe

Now the output is limited to the filtered events, and only the process of interest is highlighted.

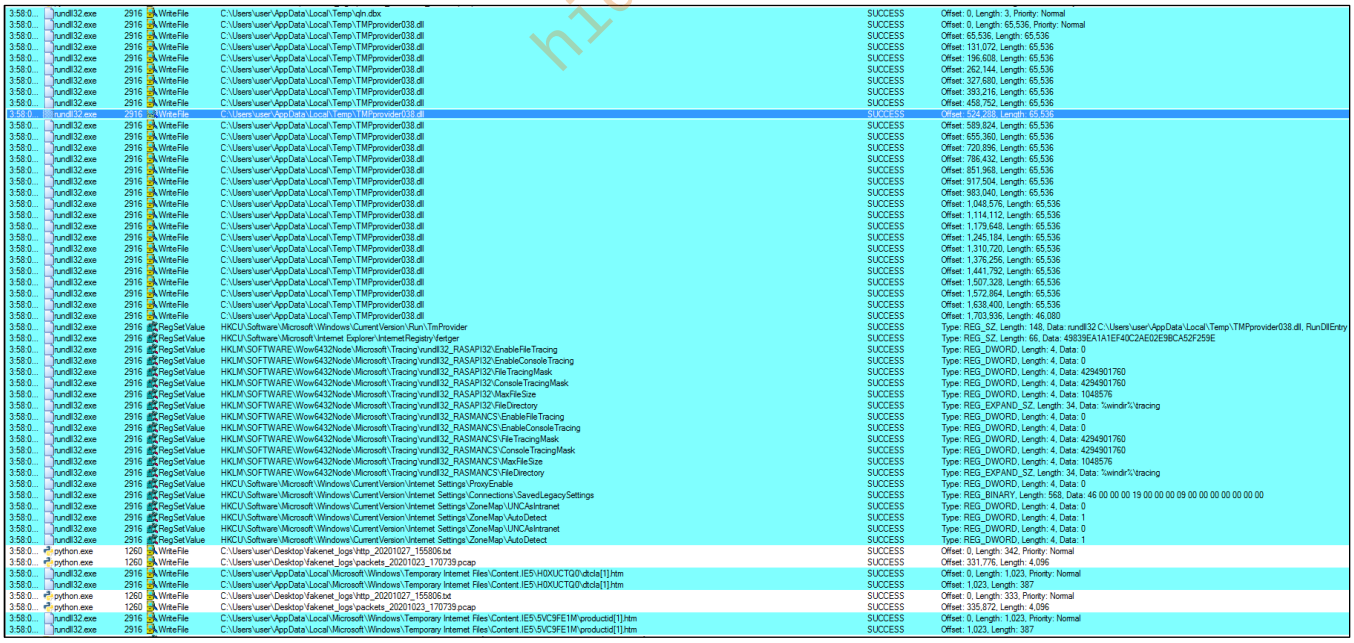

Figure 13: Highlighted output makes malware activity easier to recognize

The first interesting event is a `WriteFile` operation to `C:\Users\user\AppData\Local\Temp\qln.dbx`. Double click on the event to view more details.

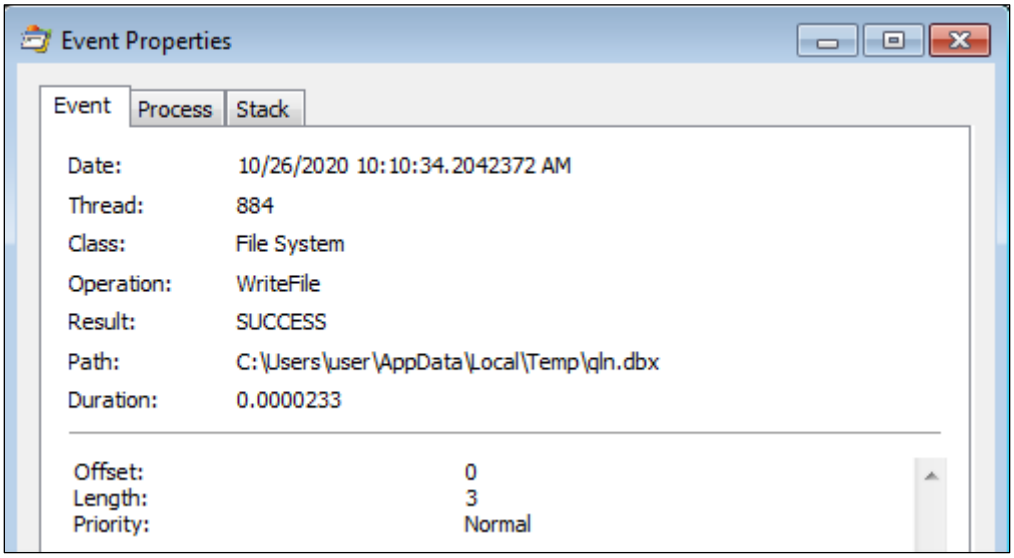

Figure 14: Event details for WriteFile operation

The length of the operation is only 3 bytes. Close this window, right click on the event, and choose "*Jump To…*." This takes you to the file location in "*File Explorer*".

hide01.ir

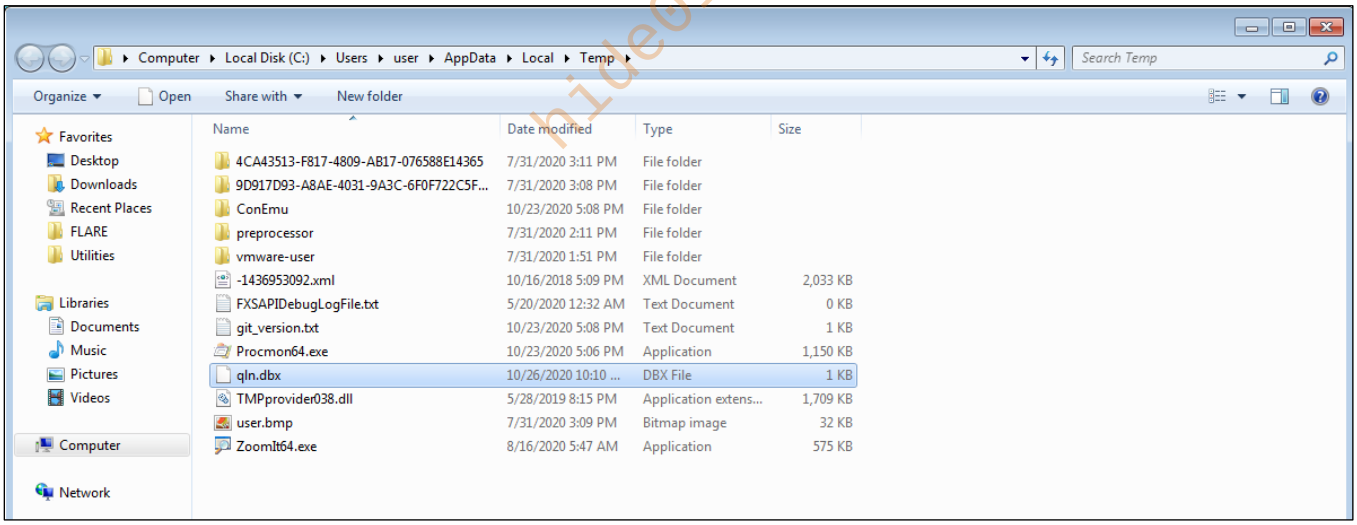

Figure 15: "File Explorer" displays dropped files

Open the file in "*010 Editor*" to view the contents. The text is `044`.

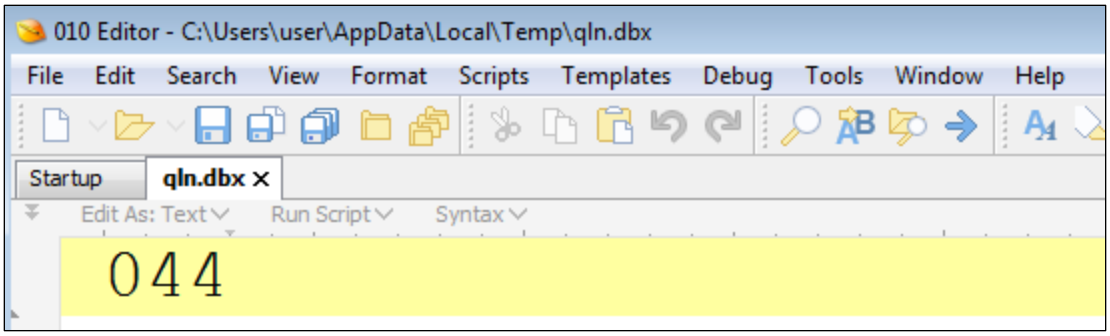

Figure 16: Examine file contents with "010 Editor"

At this point there is not enough information to understand the purpose of this file, but it can be recorded as a host-based indicator. Continue analyzing the "*Process Monitor*" output. Next there is a sequence of `WriteFile` operations to `C:\Users\user\AppData\Local\Temp\TMPprovider038.dll`. Right click and choose "*Jump To*…." to view the file. Often malware will copy itself into a common directory in order to blend in with Windows system files. `C:\Users\user\AppData\Local\Temp\` is likely the Windows Environment Variable `%TEMP%`. Compare the hash of the new file to the original in order to confirm this theory.

```
C:\Users\user\Desktop\Labs\01_Basic Static and Dynamic>sigcheck -h TMPprovider038.dll

Sigcheck v2.73 - File version and signature viewer
Copyright (C) 2004-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Users\user\Desktop\Labs\01_Basic Static and Dynamic\TMPprovider038.dll:
        Verified:       Unsigned
        Link date:      8:36 PM 1/18/2018
        Publisher:      n/a
        Company:        n/a
        Description:    n/a
        Product:        n/a
        Prod version:   n/a
        File version:   n/a
        MachineType:    32-bit
        MD5:    713111BF1249A567B9928C75901A5FB8
        SHA1:   68461EFB9E133567BFC77990A5F061BE03CF554E
        PESHA1: EB469A7DFC0291457F39407790C982C4BBF1FCA7
        PE256:  82777261C3CAAE9F9676F90DC7F2A1F02B8AAFF0B2A58CA4D175445AA6D62F2A
        SHA256: 98744AFC621B0A96034E6A4970110E6114A8405AA6DE5DF4268B8A022DEA4C5C
        IMP:    346E55297EA79E1C1F6919C96F942424

C:\Users\user\Desktop\Labs\01_Basic Static and Dynamic>sigcheck -h C:\Users\user\AppData\Local\Temp\TMPprovider038.dll

Sigcheck v2.73 - File version and signature viewer
Copyright (C) 2004-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\users\user\appdata\local\temp\TMPprovider038.dll:
        Verified:       Unsigned
        Link date:      8:36 PM 1/18/2018
        Publisher:      n/a
        Company:        n/a
        Description:    n/a
        Product:        n/a
        Prod version:   n/a
        File version:   n/a
        MachineType:    32-bit
        MD5:    713111BF1249A567B9928C75901A5FB8
        SHA1:   68461EFB9E133567BFC77990A5F061BE03CF554E
        PESHA1: EB469A7DFC0291457F39407790C982C4BBF1FCA7
        PE256:  82777261C3CAAE9F9676F90DC7F2A1F02B8AAFF0B2A58CA4D175445AA6D62F2A
        SHA256: 98744AFC621B0A96034E6A4970110E6114A8405AA6DE5DF4268B8A022DEA4C5C
        IMP:    346E55297EA79E1C1F6919C96F942424
```

Figure 17: sigcheck is used to compare file hashes and verify dropped file is identical to original sample

Now that it has been confirmed, continue to analyze the "*Process Monitor*" output. The next interesting event is a `RegSetValue` operation with path `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\TmProvider`.

Double click to see the details and observe that the data written to that registry value is "`rundll32 C:\Users\user\AppData\Local\Temp\TMPprovider038.dll, RunDllEntry`". The registry subkey `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` is used to register programs to run automatically on system start. In this case `rundll32` is used to launch the DLL export `RunDllEntry`, establishing persistence on the host. Note this as another host-based indicator.

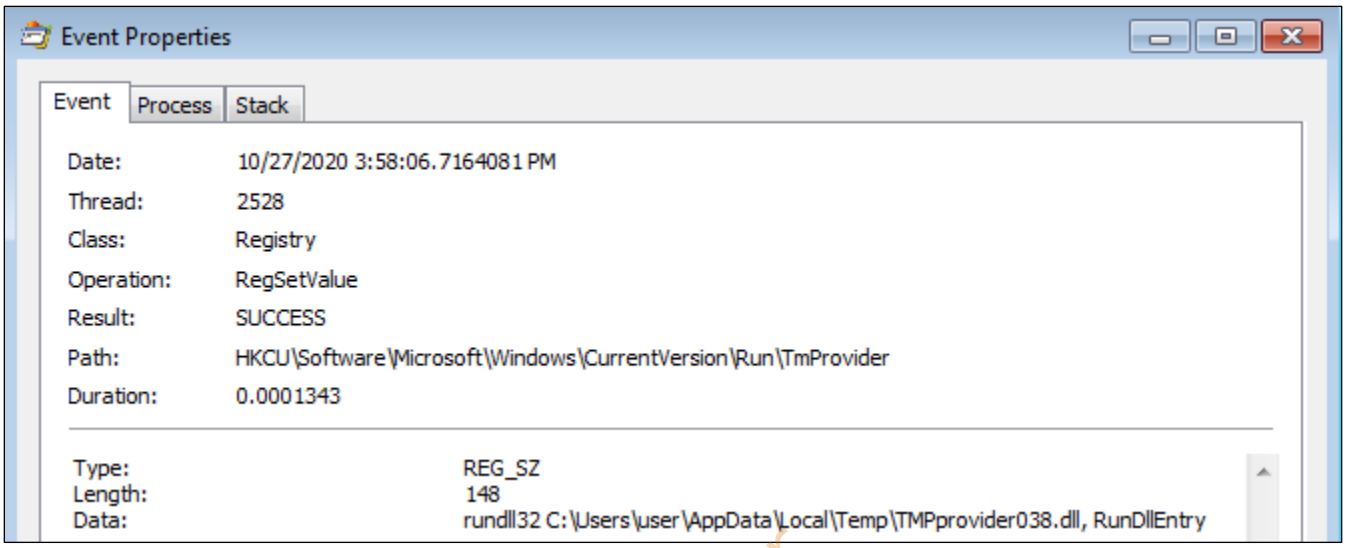

Figure 18: Persistence is achieved via registry

The next interesting event is another `RegSetValue` operation with value "`HKCU\Software\Microsoft\Internet Explorer\InternetRegistry\fertger`". Double click to see details and observe the data written is a hexadecimal string `49839EA1A1EF40C2AE02E9BCA52F259E`. At this point there is insufficient data to understand the purpose of this behavior, however this registry key should be documented as a potential host-based indicator.

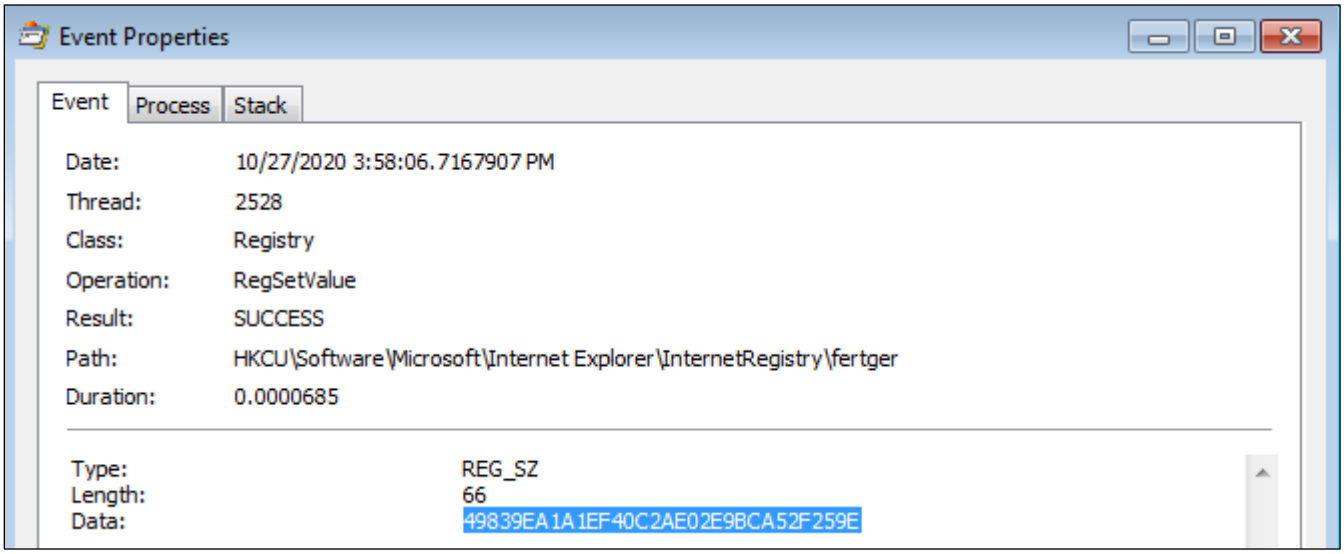

Figure 19: Unknown hexadecimal string written to fertger registry subkey

The remaining "*Process Monitor*" output is less relevant to our malware analysis tools. The `RegSetValue` operations to `HKLM\SOFTWARE\Wow6432Node\Microsoft\Tracing` and `"HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings"` are common behavior related to the Windows internet API. The `WriteFile` operations to `"C:\Users\user\AppData\Local\Microsoft\Windows\Temporary Internet Files"` are related to caching web requests. These can all be ignored.

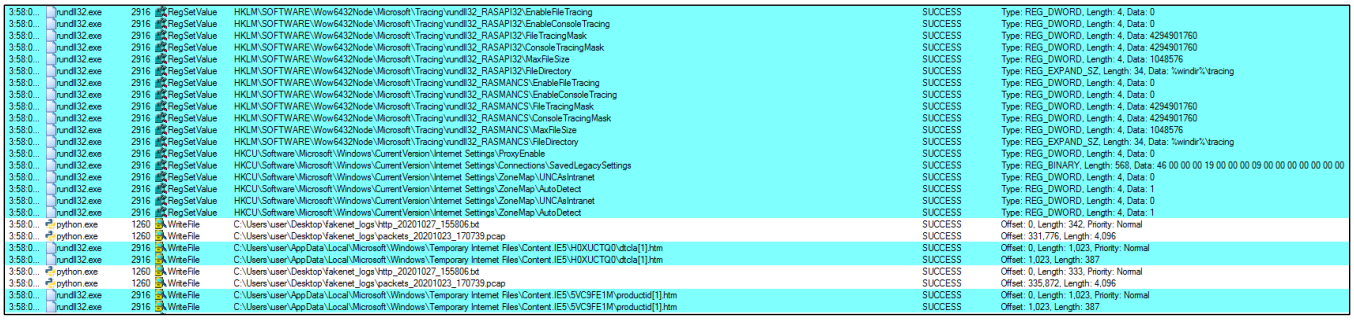

Figure 20: "Process Monitor" captures events related to caching web requests

List any potential host-based indicators of this malware.

Referring to the above analysis, the host-based indicators are:

The file `C:\Users\user\AppData\Local\Temp\qln.dbx` is created and populated with the string `044.`

The malware is copied to `C:\Users\user\AppData\Local\Temp\TMPprovider038.dll.`

The registry value `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\TmProvider` is set to `"C:\Users\user\AppData\Local\Temp\TMPprovider038.dll, RunDllEntry"`

`The registry value "HKCU\Software\Microsoft\Internet Explorer\InternetRegistry\fertger"` is set to `"49839EA1A1EF40C2AE02E9BCA52F259E"`

List any potential network-based indicators of this malware

Referring to the above analysis, the host-based indicators are:

Two variant HTTP POST requests are made.

1. `flossme.mandiant.com` on port 80. The query string is `/geo/productid.php?id=BE92941DA4AC4AC9BA38C6A4F3BBE1D7&v1=038&v2=261857261&q=5265882854508EFCF958F979E4`.

2. `fauxnet.mandiant.com` on port 80. The query string is `/wp08/wp-includes/dtcla.php?id=BE92941DA4AC4AC9BA38C6A4F3BBE1D7&v1=038&v2=261857261&q=5265882854508EFCF958F979E4`.

The HTTP User-Agent: `"Mozilla / 5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit / 525.19 (KHTML, like Gecko) Chrome / 1.0.154.36 Safari / 525.19"`

hide01.ir