Summary:

Name: sample.bin
Nicknames: Pincav
Analysis findings: Evasive / persistent malware

Hash values:
Sample1.exe
MD5: 38F151B5164D18158BE1D6E3493A897D
SHA256: 516935769CE832AE4E31E38AE0764009F90B55208710B29987F9289BD4FAFC3D

Sample2_unpacked.dll
MD5: 516935769CE832AE4E31E38AE0764009F90B55208710B29987F9289BD4FAFC3D
SHA256: AF085618094A6D1FB3E6D533B2B57D4A9C2DCB731F7824E1D7C21E5FFF8844C8

Environment: Windows 7 Professional, Service Pack 1. 32-bit OS.

Tools:

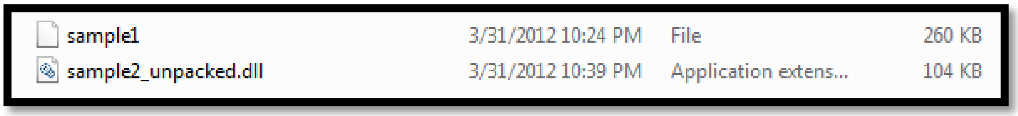
- IDA Pro Free v5.0
- QuickHash-Windows v2.8.0
- PEid v0.65
- Process Explorer v16.20
- Process Monitor v3.32
- Regshot v1.8.3

NOTE: Sample ran with administrative privileges

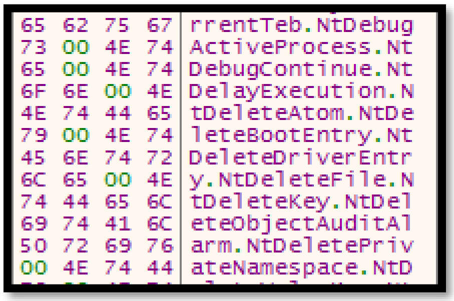
Analysis:

What is the purpose of each malware sample?

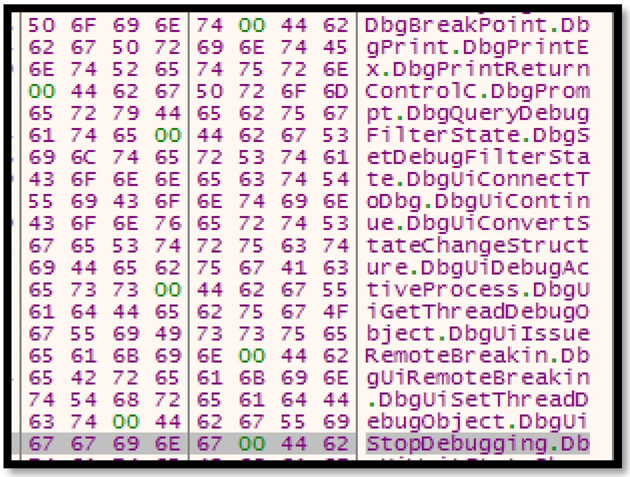
- To better answer this question readers should understand what a DLL is. DLL stands for Dynamic Linked library. These are linked libraries that are saved to memory which other programs can utilize (Microsoft, 2022). These libraries are 'common' libraries that can used by numerous programs which encourages code reuse and efficient memory usage. The advantage of having an already unpacked DLL sample allows us to better analyze the sample using a debugger or static analysis tool. Using a debugger, we can break down the DLL's to see exactly how each one functions and what is affects. This was particularly helpful to have as this sample was found to be de-bugger aware.



sample1	3/31/2012 10:24 PM	File	260 KB
sample2_unpacked.dll	3/31/2012 10:39 PM	Application extens...	104 KB



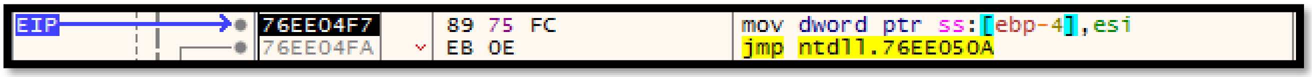
```
65 62 75 67  rrentTeb.NtDebug
73 00 4E 74  ActiveProcess.Nt
65 00 4E 74  DebugContinue.Nt
6F 6E 00 4E  DelayExecution.N
4E 74 44 65  tDeleteAtom.NtDe
79 00 4E 74  leteBootEntry.Nt
45 6E 74 72  DeleteDriverEntr
6C 65 00 4E  y.NtDeleteFile.N
74 44 65 6C  tDeleteKey.NtDel
69 74 41 6C  eteObjectAuditAl
50 72 69 76  arm.NtDeletePriv
00 4E 74 44  ateNamespace.NtD
```



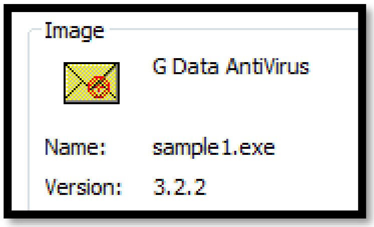
```
50 6F 69 6E  74 00 44 62  DbgBreakPoint.Db
62 67 50 72  69 6E 74 45  gPrint.DbgPrintE
6E 74 52 65  74 75 72 6E  x.DbgPrintReturn
00 44 62 67  50 72 6F 6D  ControlC.DbgProm
65 72 79 44  65 62 75 67  pt.DbgQueryDebug
61 74 65 00  44 62 67 53  FilterState.DbgS
69 6C 74 65  72 53 74 61  etDebugFilterSta
43 6F 6E 6E  65 63 74 54  te.DbgUiConnectT
55 69 43 6F  6E 74 69 6E  oDbg.DbgUiContin
43 6F 6E 76  65 72 74 53  ue.DbgUiConvertS
67 65 53 74  72 75 63 74  tateChangeStruct
69 44 65 62  75 67 41 63  ure.DbgUiDebugAc
65 73 73 00  44 62 67 55  tiveProcess.DbgU
61 64 44 65  62 75 67 4F  iGetThreadDebugO
67 55 69 49  73 73 75 65  bject.DbgUiIssue
65 61 6B 69  6E 00 44 62  RemoteBreakin.Db
65 42 72 65  61 6B 69 6E  gUiRemoteBreakin
74 54 68 72  65 61 64 44  .DbgUiSetThreadD
63 74 00 44  62 67 55 69  ebugObject.DbgUi
67 67 69 6E  67 00 44 62  StopDebugging.Db
```

What persistence mechanism does this malware use? What files are involved in this?

- This sample masquerades as the file 'G Data Antivirus' and uses multiple methods of persistence including trojanized system binaries (Sikorski & Honig, 2012). We can see multiple jumps to 'ntdll' a likely sign that it has been maliciously modified.
- Additionally, a windows batch file named '115097' appears when the program is executed as admin. This file launches a shell as seen below. This shell is tied to some of the program's abilities when ran with admin privileges that are covered later, but it's important to note that the program is able to create shells.



```
EIP  76EE04F7   89 75 FC   mov dword ptr ss:[ebp-4],esi
     76EE04FA   EB 0E      jmp ntdll.76EE050A
```

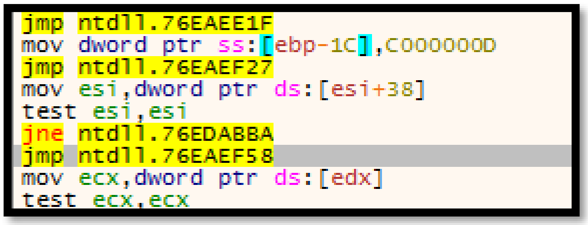


Image

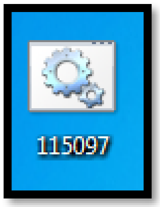
G Data AntiVirus

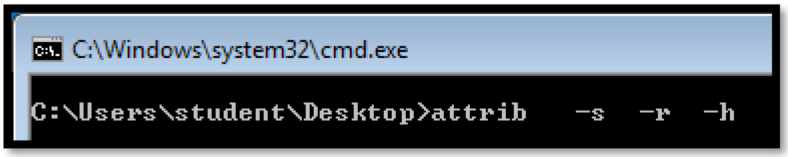
Name: sample1.exe

Version: 3.2.2



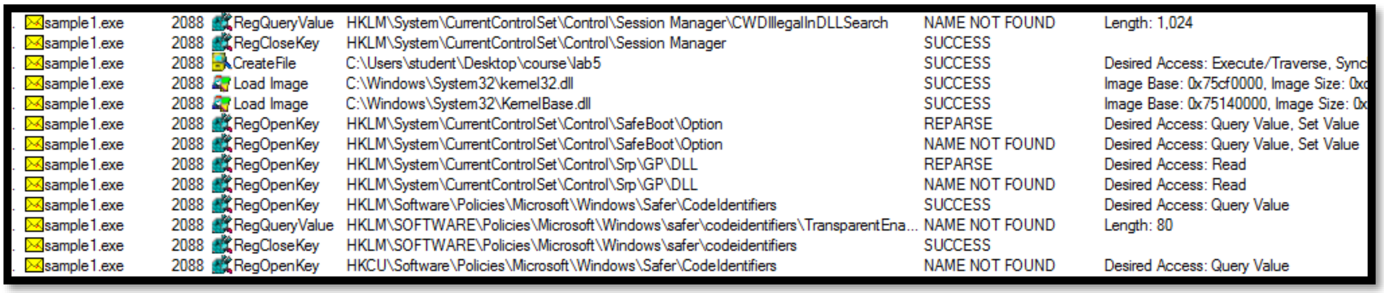
```
jmp ntdll.76EAEE1F
mov dword ptr ss:[ebp-1C],C000000D
jmp ntdll.76EAEF27
mov esi,dword ptr ds:[esi+38]
test esi,esi
jne ntdll.76EDABBA
jmp ntdll.76EAEF58
mov ecx,dword ptr ds:[edx]
test ecx,ecx
```





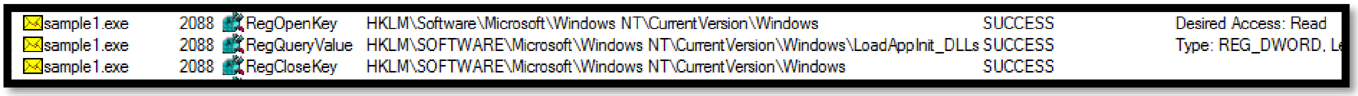
```
C:\Windows\system32\cmd.exe

C:\Users\student\Desktop>attrib   -s   -r   -h
```

- DLL load-Order hijacking. This method abuses vulnerable DLL's which then try to load other DLL's created by the attacker. Also known as DLL side-loading. Evidenced by the changes inside the System32 directory and many others.



```
sample1.exe  2088  RegQueryValue  HKLM\System\CurrentControlSet\Control\Session Manager\CWDIllegalInDLLSearch  NAME NOT FOUND  Length: 1,024
sample1.exe  2088  RegCloseKey    HKLM\System\CurrentControlSet\Control\Session Manager                        SUCCESS
sample1.exe  2088  CreateFile     C:\Users\student\Desktop\course\lab5                                         SUCCESS         Desired Access: Execute/Traverse, Sync
sample1.exe  2088  Load Image     C:\Windows\System32\kernel32.dll                                             SUCCESS         Image Base: 0x75cf0000, Image Size: 0x
sample1.exe  2088  Load Image     C:\Windows\System32\KernelBase.dll                                           SUCCESS         Image Base: 0x75140000, Image Size: 0x
sample1.exe  2088  RegOpenKey     HKLM\System\CurrentControlSet\Control\SafeBoot\Option                        REPARSE         Desired Access: Query Value, Set Value
sample1.exe  2088  RegOpenKey     HKLM\System\CurrentControlSet\Control\SafeBoot\Option                        NAME NOT FOUND  Desired Access: Query Value, Set Value
sample1.exe  2088  RegOpenKey     HKLM\System\CurrentControlSet\Control\Srp\GP\DLL                             REPARSE         Desired Access: Read
sample1.exe  2088  RegOpenKey     HKLM\System\CurrentControlSet\Control\Srp\GP\DLL                             NAME NOT FOUND  Desired Access: Read
sample1.exe  2088  RegOpenKey     HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers               SUCCESS         Desired Access: Query Value
sample1.exe  2088  RegQueryValue  HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers\TransparentEna...  NAME NOT FOUND  Length: 80
sample1.exe  2088  RegCloseKey    HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers               SUCCESS
sample1.exe  2088  RegOpenKey     HKCU\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers               NAME NOT FOUND  Desired Access: Query Value
```

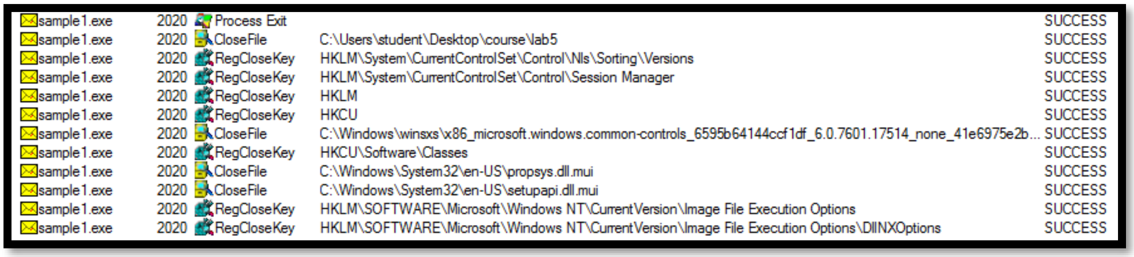
- This malware uses Windows API functions which create and edit Windows registry keys to maintain persistence after running. These changes remain in effect even after the computer is reset.



```
sample1.exe   2088 RegOpenKey    HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows                SUCCESS   Desired Access: Read
sample1.exe   2088 RegQueryValue HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\LoadAppInit_DLLs SUCCESS   Type: REG_DWORD, Le
sample1.exe   2088 RegCloseKey   HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows                SUCCESS
```

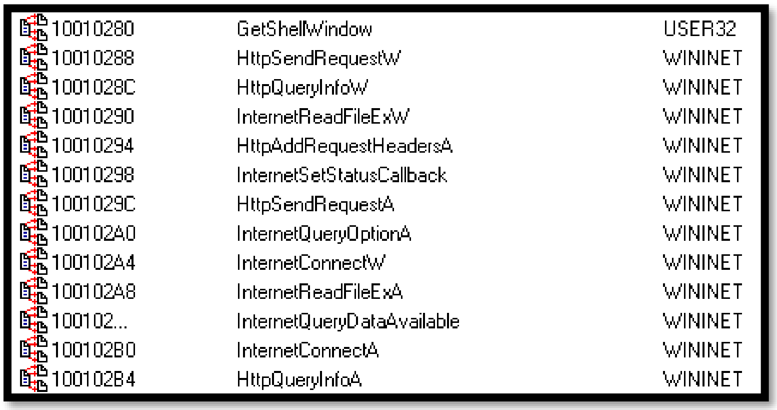
What capabilities does the malware possess? If there are commands associated with a CnC, what are they?

- As noted earlier, we can see that the malware can create shells, which can easily be combined with other network tools (netcat) to create listeners for CnC activity.
- This malware can also create and edit Windows registry keys. Therefore, it is persistent and will remain on systems even after restarting.



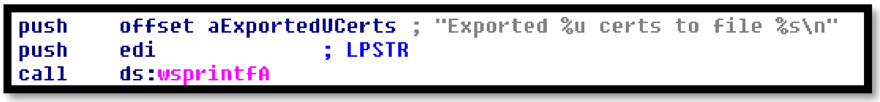
```
sample1.exe   2020 Process Exit                                                                                                   SUCCESS
sample1.exe   2020 CloseFile    C:\Users\student\Desktop\course\lab5                                                              SUCCESS
sample1.exe   2020 RegCloseKey  HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions                                        SUCCESS
sample1.exe   2020 RegCloseKey  HKLM\System\CurrentControlSet\Control\Session Manager                                             SUCCESS
sample1.exe   2020 RegCloseKey  HKLM                                                                                              SUCCESS
sample1.exe   2020 RegCloseKey  HKCU                                                                                              SUCCESS
sample1.exe   2020 CloseFile    C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_41e6975e2b... SUCCESS
sample1.exe   2020 RegCloseKey  HKCU\Software\Classes                                                                             SUCCESS
sample1.exe   2020 CloseFile    C:\Windows\System32\en-US\propsys.dll.mui                                                         SUCCESS
sample1.exe   2020 CloseFile    C:\Windows\System32\en-US\setupapi.dll.mui                                                        SUCCESS
sample1.exe   2020 RegCloseKey  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options                    SUCCESS
sample1.exe   2020 RegCloseKey  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DllNXOptions       SUCCESS
```

- It also attempts to access the internet through http requests, browsers, and shells. As we will see later in the report, this malware also contains functions to look for specific web browsers to establish outside connections, likely for CnC activities.

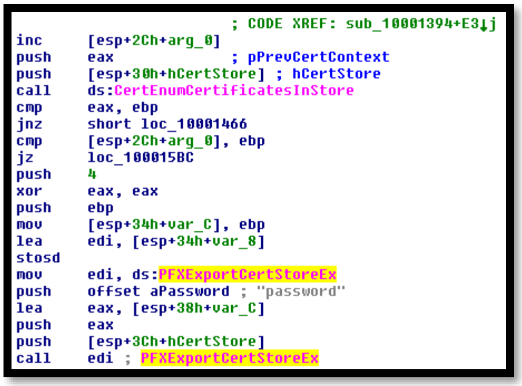


```
10010280   GetShellWindow             USER32
10010288   HttpSendRequestW           WININET
1001028C   HttpQueryInfoW             WININET
10010290   InternetReadFileExW        WININET
10010294   HttpAddRequestHeadersA     WININET
10010298   InternetSetStatusCallback  WININET
1001029C   HttpSendRequestA           WININET
100102A0   InternetQueryOptionA       WININET
100102A4   InternetConnectW           WININET
100102A8   InternetReadFileExA        WININET
100102...  InternetQueryDataAvailable WININET
100102B0   InternetConnectA           WININET
100102B4   HttpQueryInfoA             WININET
```

- Inside IDA Pro there is additional evidence that the malware can export private keys and read/write private certificates using Wincrypt calls (Microsoft, 2022). This could allow an attacker to have secure access and control to the users' computer resources.



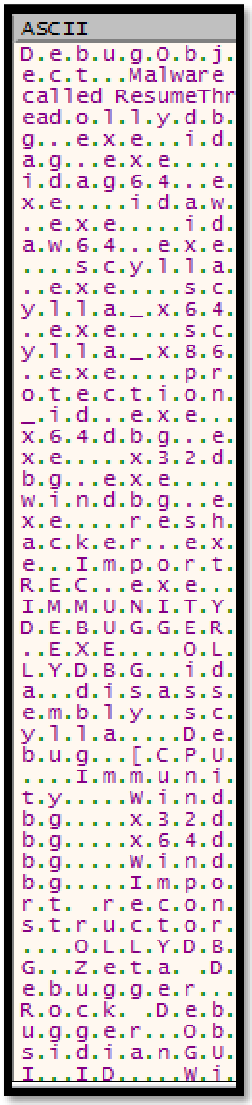
```
push    offset aExportedUCerts ; "Exported %u certs to file %s\n"
push    edi                 ; LPSTR
call    ds:wsprintfA
```



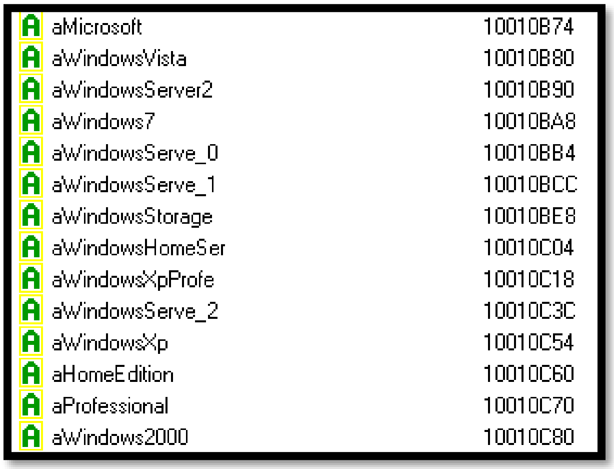
```
                                ; CODE XREF: sub_10001394+E3↓j
inc     [esp+2Ch+arg_0]
push    eax                     ; pPrevCertContext
push    [esp+30h+hCertStore] ; hCertStore
call    ds:CertEnumCertificatesInStore
cmp     eax, ebp
jnz     short loc_10001466
cmp     [esp+2Ch+arg_0], ebp
jz      loc_100015BC
push    4
xor     eax, eax
push    ebp
mov     [esp+34h+var_C], ebp
lea     edi, [esp+34h+var_8]
stosd
mov     edi, ds:PFXExportCertStoreEx
push    offset aPassword ; "password"
lea     eax, [esp+38h+var_C]
push    eax
push    [esp+3Ch+hCertStore]
call    edi ; PFXExportCertStoreEx
```

Are there any process specific checks in this malware, i.e. does it behave differently depending upon what the process is named? If so, what processes does it behave differently under? Generate a hypothesis on what the malware is doing.

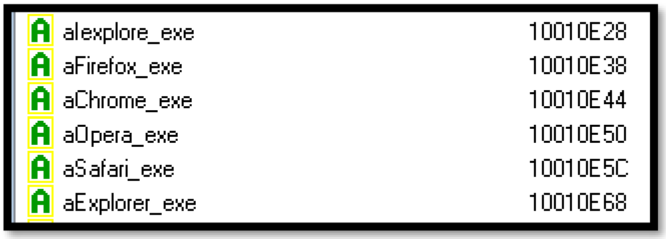
- Firstly, this malware is debugger aware, meaning it will act differently if it detects known debuggers including ollydbg, Immunity, Scylla, x32 or x64 and others, as seen inside the memory dumps below. This was also evidenced as the malware was extremely evasive when ran as sample exe with debugging tools.
- Additionally, we can see checks for different Windows Operating System environments. This is not surprising as most malware is designed for Windows environments (Drapkin, 2021).
- Shell creation also requires administrative privileges, as evidenced when the program is run. Therefore, this program does not fully function when ran as a normal user. This shell is likely used as a backdoor for adversaries to utilize.
- Lastly, we can see checks for specific web browsers, including internet explorer, chrome, and Firefox.



```
ASCII
D.e.b.u.g.O.b.j.
e.c.t...Malware
called ResumeThr
ead.o.l.l.y.d.b.
g...e.x.e...i.d.
a.g...e.x.e.....
i.d.a.g.6.4...e.
x.e.....i.d.a.w.
..e.x.e.....i.d.
a.w.6.4...e.x.e.
....s.c.y.l.l.a.
..e.x.e.....s.c.
y.l.l.a._.x.6.4.
..e.x.e.....s.c.
y.l.l.a._.x.8.6.
..e.x.e.....p.r.
o.t.e.c.t.i.o.n.
_.i.d...e.x.e...
x.6.4.d.b.g...e.
x.e.....x.3.2.d.
b.g...e.x.e.....
w.i.n.d.b.g...e.
x.e.....r.e.s.h.
a.c.k.e.r...e.x.
e...I.m.p.o.r.t.
R.E.C...e.x.e...
I.M.M.U.N.I.T.Y.
D.E.B.U.G.G.E.R.
..E.X.E.....O.L.
L.Y.D.B.G...i.d.
a...d.i.s.a.s.s.
e.m.b.l.y...s.c.
y.l.l.a.....D.e.
b.u.g...[.C.P.U.
....I.m.m.u.n.i.
t.y.....W.i.n.d.
b.g.....x.3.2.d.
b.g.....x.6.4.d.
b.g.....W.i.n.d.
b.g.....I.m.p.o.
r.t. .r.e.c.o.n.
s.t.r.u.c.t.o.r.
....O.L.L.Y.D.B.
G...Z.e.t.a. .D.
e.b.u.g.g.e.r...
R.o.c.k. .D.e.b.
u.g.g.e.r...O.b.
s.i.d.i.a.n.G.U.
I...I.D.....W.i.
```

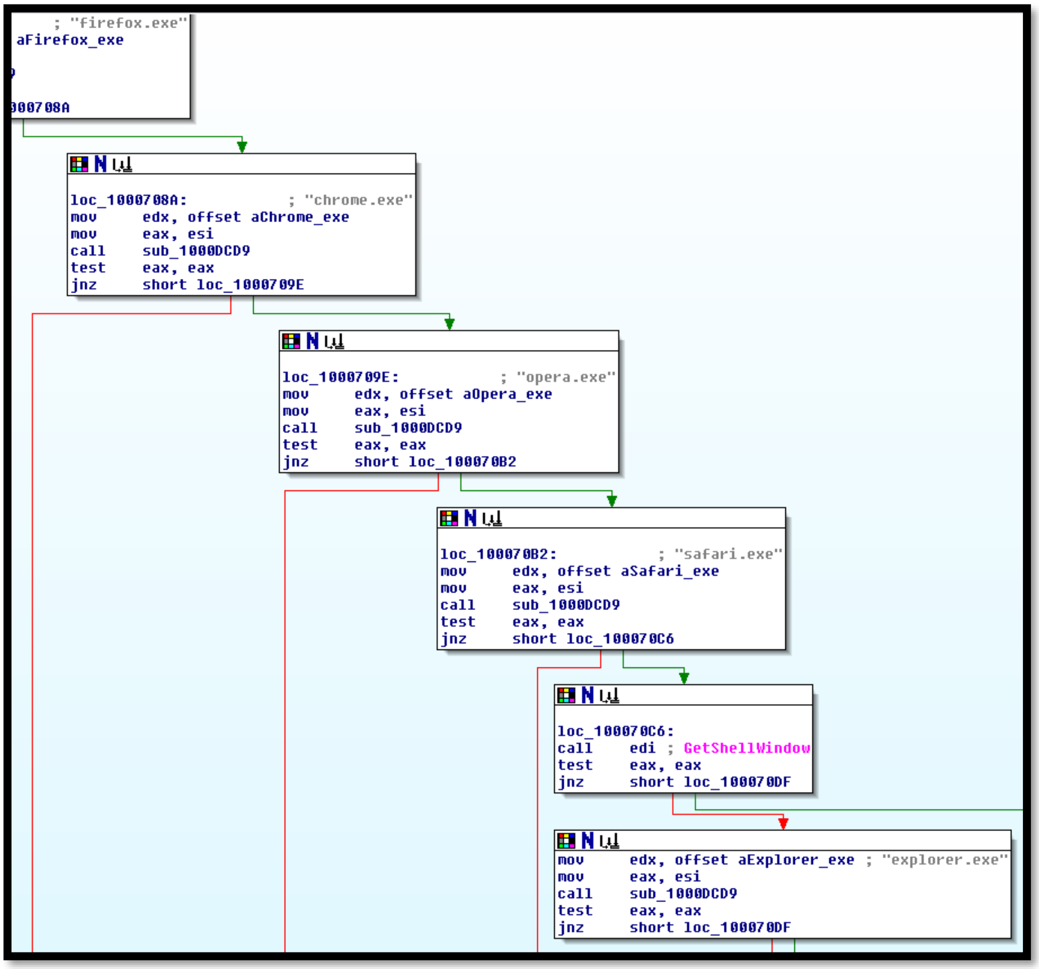


aMicrosoft	10010B74
aWindowsVista	10010B80
aWindowsServer2	10010B90
aWindows7	10010BA8
aWindowsServe_0	10010BB4
aWindowsServe_1	10010BCC
aWindowsStorage	10010BE8
aWindowsHomeSer	10010C04
aWindowsXpProfe	10010C18
aWindowsServe_2	10010C3C
aWindowsXp	10010C54
aHomeEdition	10010C60
aProfessional	10010C70
aWindows2000	10010C80



aIexplore_exe	10010E28
aFirefox_exe	10010E38
aChrome_exe	10010E44
aOpera_exe	10010E50
aSafari_exe	10010E5C
aExplorer_exe	10010E68

(Continued…)



```
    ; "firefox.exe"
 aFirefox_exe

9

000708A
```

```
loc_1000708A:               ; "chrome.exe"
mov     edx, offset aChrome_exe
mov     eax, esi
call    sub_1000DCD9
test    eax, eax
jnz     short loc_1000709E
```

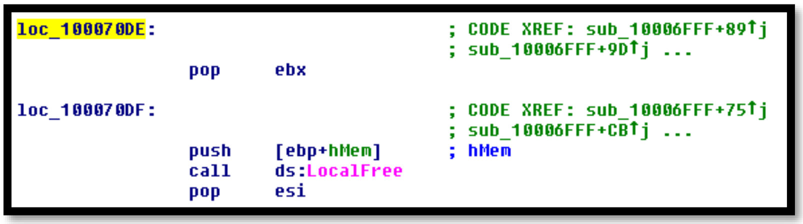
```
loc_1000709E:               ; "opera.exe"
mov     edx, offset aOpera_exe
mov     eax, esi
call    sub_1000DCD9
test    eax, eax
jnz     short loc_100070B2
```

```
loc_100070B2:               ; "safari.exe"
mov     edx, offset aSafari_exe
mov     eax, esi
call    sub_1000DCD9
test    eax, eax
jnz     short loc_100070C6
```

```
loc_100070C6:
call    edi ; GetShellWindow
test    eax, eax
jnz     short loc_100070DF
```

```
mov     edx, offset aExplorer_exe ; "explorer.exe"
mov     eax, esi
call    sub_1000DCD9
test    eax, eax
jnz     short loc_100070DF
```

- Inside IDA Pro we can see an overview of the different mechanisms that are triggered after the web browser is identified.
- It is likely that the malware spawns a shell to communicate with outside channels if a specific web browser is detected. After the browser checks, we can see the arrows leading to the ‘GetShellWindow’ (above) function, otherwise we are led to a ‘LocalFree’ function (below). LocalFree is a windows function that “frees the specified local memory object and invalidates its handle” (Microsoft, 2021). This is likely tied to freeing up resources for command-and-control type activity commonly found in trojans and other malware.



```
loc_100070DE:                              ; CODE XREF: sub_10006FFF+89↑j
                                           ; sub_10006FFF+9D↑j ...
                pop     ebx

loc_100070DF:                              ; CODE XREF: sub_10006FFF+75↑j
                                           ; sub_10006FFF+CB↑j ...
                push    [ebp+hMem]         ; hMem
                call    ds:LocalFree
                pop     esi
```

References

Drapkin, A. (2021, November 26). *Over 100 million pieces of malware were made for Windows Users in 2021*. Retrieved from Tech.co: https://tech.co/news/windows-users-malware

Microsoft. (2021, October 13). *LocalFree Function (winbase.h)*. Retrieved from Microsoft: https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-localfree

Microsoft. (2022, October 25). *PFXExportCertStoreEX function (wincrypt.h)*. Retrieved from Microsoft: https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-pfxexportcertstoreex

Microsoft. (2022, April 12). *What is a DLL*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The hands-on guide to disecting malicious software.* No Starch Press.

VirusTotal. (2022, October 01). *516935769ce832ae4e31e38ae0764009f90b55208710b29987f9289bd4fafc3d*. Retrieved from VirusTotal: https://www.virustotal.com/gui/file/516935769ce832ae4e31e38ae0764009f90b55208710b29987f9289bd4fafc3d/behavior