June 2021

Malware Analysis Report on

INFOSTEALER DEXTER

(32-bit Executable)

SHUBHAM CHOUBEY

OVERVIEW

Back in 2012-2013, we saw this malware that targeted POS (Point of Sale like checkout counters) systems to store payments card data. It caused damage to lots of big companies like Adobe, Google, Microsoft, Visa etc. It has various functionalities like changing registry values and add new values, keylogging, contacting a server to download additional malware

INITIAL TRIAGE

Before looking at the assembly code or running the executable, it is advised to go for a passive scan for potential malicious files.

Here are the few steps I took and their results :

Activity	Result
Filename	Win33.exe
md5hash	140d24af0c2b3a18529df12dfbc5f6de
sha256hash	4eabb1adc035f035e010c0d0d259c683e18193f509946652ed8aa7c5d92b6a92
Executable size on disk	68,096 bytes
Compile Time	0x521E23B1 (Wed Aug 28 09:22:09 2013)
32-bit Executable	Yes
Subsystem	GUI
# of AV Engines that deem it as a malware	59/70
Executable made in	Microsoft Visual C++ ~v.7.10 - 8 - Visual 2005
Packed	No

Table-1 : Initial Triage

The next step we took was to look at the Libraries, Imports and Exports(0) :

Libraries	# of Imports
wininet.dll	8
Urlmon.dll	1
Ws2_32.dll	3
Rpcrt4.dll	1
Kernel32.dll	75
User32.dll	10
Advapi32.dll	13
Shell32.dll	1

Ole32.dll	1
Shlwapi.dll	2

Table-2 : Libraries and Imports

Next step was to look at Strings for low hanging fruits, some important info we got were:

- 151.248.115.107
- /w1921831741862103104 1543/gateway.php
- ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
- SeDebugPrivilege
- Software\Microsoft\Windows\CurrentVersion\Policies\Associations
- Software\Microsoft\Windows\CurrentVersion\Run
- smss.exe
- csrss.exe
- lsass.exe
- wuauclt.exe
- iexplorer.exe
- wmiprvse.exe
- Windows 2000
- Windows XP
- Windows XP Professional x64
- Windows Server 2003
- Windows Home Server
- Windows Server 2003 R2
- Windows Vista
- Windows Server 2008
- Windows Server R2
- Windows 7
- 64 Bit
- 32 Bit
- Sun Java Secuity Plugin

It’s very likely that, this malware tries to contact to its C&C server and tries to download additional firepower(probably more malware and probably also acts as a Downloader)
I used Resource Hacker to look at the resources section and found out that it does contain a binary file as it’s binary header does show “4D 5A” which is “**MZ”**. I took that binary and saved it as a .bin file and begain looking at it through strings, didn’t find much except it checked for functions like IsDebuggerPresent, ADVAPI32.dll etc

md5sum – 2e3f5b165ab841aa3c59e264280cb2fc
sha256 - c06259278ac703b4ad46f3eaa0f3b4d95ad628f616692db1efb5f6cce5cc22d9

It tries to do Privilege Escalatation using **SeDebugPrivilege** which basically is a tool for system-level debugging. It is given only to local administrators. This will be explored more in Static Analysis
Next we see some registry values and some exec files. We also see that this malware is set to be used on various older versions of Windows as well.

DYNAMIC ANALYSIS

This is the part where we run the malware in an isolated environment and observe what it is capable of doing. We will be looking for changes in registry, New processes started, Handles opened, dll files used etc, does it contact its C&C and more.

Environment

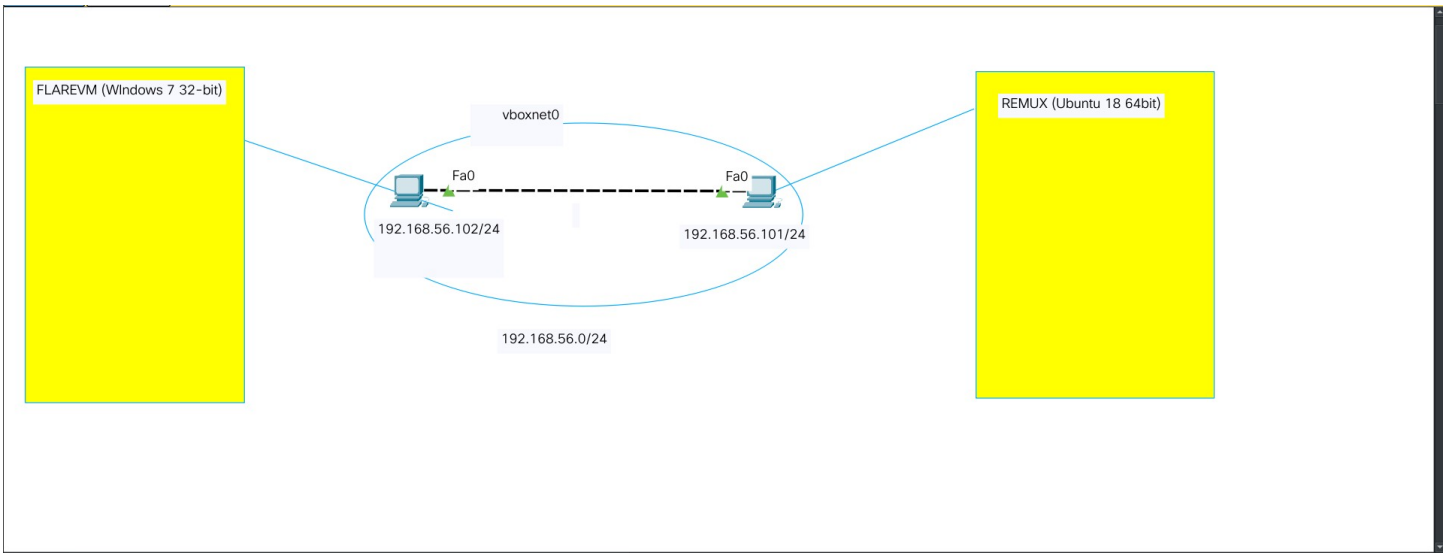

Figure -1 Virtual Machines running on Host connected via vboxnet0

FlareVM– Machine which I will be using to analyze the malware, Contains tools like regshot, procmon, procexp, pestudio, ida, immunity debugger
Remnux – Machine which will be used to act as a fake C&C and sniff packets, contains tools like Wireshark, fakenet, insetsim

The idea is if FlareVM tries to connect to any ip address or dns server, remnux would act as that and will comminicate with FlareVM and hence will get whatever application data being transmitted.

Tools Used:

Regshot Procmon Procexp InetSim Fakenet apimonitor

Registry Changes :

Using regshot we managed to compare the changes made between before and after running the malware.

Keys Added:

- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{AF5B2E1D-EC0B-43D8-B584-E045CA98C326}
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\HelperSolutions Software

Values Added:

- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\VRHfre\Qrfxgbc\Jva32.VasbFgrnyre.Qrkgre\jva33.rkr: 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF D0 20 50 E3 56 6A D7 01 00 00 00 00
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{AF5B2E1D-EC0B-43D8-B584-E045CA98C326}\WpadDecisionReason: 0x00000001
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{AF5B2E1D-EC0B-43D8-B584-E045CA98C326}\WpadDecisionTime: BC 08 05 E7 56 6A D7 01
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{AF5B2E1D-EC0B-43D8-B584-E045CA98C326}\WpadDecision: 0x00000000
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{AF5B2E1D-EC0B-43D8-B584-E045CA98C326}\WpadNetworkName: "Unidentified network"
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\0\1806: 0x00000000
- **HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Run\ Sun Java Security Plugin: "C:\Users\IEUser\AppData\Roaming\Java Security Plugin\javaplugin.exe"**
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\HelperSolutions Software\Digit: "578d2de3-38ae-41d9-8856-d5b6f3d8dcf6"
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\HelperSolutions Software\val1: "C:\Users\IEUser\Desktop\Win32.InfoStealer.Dexter\strokes.log"

- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\HelperSolutions Software\val2: "C:\Users\IEUser\Desktop\Win32.InfoStealer.Dexter\tmp.log"

Values Modified:

- HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastTime: 0x01D7699E76A2E458
- HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastTime: 0x01D76A56E9BAD50A
- HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\TransientValue: 40 13 0F D4 85 4F 22 40
- HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\TransientValue: DC 10 19 2F 2B AD 22 40
- HKLM\SYSTEM\ControlSet001\Control\Nsi\{eb004a03-9b1a-11d4-9123-0050047759bc}\24\ ffffffffffffffffffffffffffffff01: 01 00 00 00 4D 00 00 00 E5 11 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
- HKLM\SYSTEM\ControlSet001\Control\Nsi\{eb004a03-9b1a-11d4-9123-0050047759bc}\24\ ffffffffffffffffffffffffffffff01: 01 00 00 00 4D 00 00 00 E7 11 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
- HKLM\SYSTEM\CurrentControlSet\Control\Nsi\{eb004a03-9b1a-11d4-9123-0050047759bc}\24\ ffffffffffffffffffffffffffffff01: 01 00 00 00 4D 00 00 00 E5 11 00 00 FF FF FF FF FF FF FF FF FF FF FF FF

- HKLM\SYSTEM\CurrentControlSet\Control\Nsi\{eb004a03-9b1a-11d4-9123-0050047759bc}\24\ ffffffffffffffffffffffffffffff01: 01 00 00 00 4D 00 00 00 E7 11 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
- HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\ Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\HRZR_PGYFRFFVBA: 00 00 00 00 5B 0

Out of these values, the one in bold, “javaplugin.exe” seems to be a new file that gets created in face of a Java Security related software.It seems like it changes some values in the startup too
We also see that there were few files created:

$MALWARE_PATH/strokes.log
$MALWARE_PATH/tmp.log
$MALWARE_PATH/win33.exe -> $MALWARE_PATH/SecureDll.dll (the malware executable gets replaced by this file) also known as Keylogger.dll

I tried running SecureDll.dll using rundll32.exe but nothing happened

We also found that after running apimonitor that when the executable win33.exe is run, WerFault.exe starts which is a Windows Error reporting fault reporting process. The malware injects itself into iexplore.exe i.e Internet Explorer

Assumptions for the information we have found so far:

When the executable win33.exe is run, it injects its code in the Internet Explorer (iexplore.exe) and runs an instance of it in the background, it crashes something and Windows Error reporting process starts. It also creates a new folder in %APPDATA%/Java Security Plugin and creates an executable called javaplugin.exe, while also setting a registry instance on this file. The orignal Executable deletes itself and in its place, SecureDll.dll is created which is very likely the file from the Resouce Section and creates two file “strokes.log” and “tmp.log” where it stored some encrypted data, which kept increasing in size in real time.

It tries to communicate to the ip address “151.248.115.107” and tries to download a file “/w192183174186210310415 43/gateway.php” which we don’t know anything about yet.

Now using Static Analysis, we will check how much our assumptions holds correctly.

STATIC ANALYSIS

This is the part where we start looking at the assembly code using a disassembler and step through the instructions using a Debugger.

I will be using Ida Pro 5.0

```
.text:00403B14                  push    eax             ; lpFirst
.text:00403B15                  call    ds:StrStrA
.text:00403B1B                  test    eax, eax
.text:00403B1D                  jz      loc_403C02
.text:00403B23                  mov     ecx, [ebp+lpName]
.text:00403B26                  push    ecx             ; lpName
.text:00403B27                  push    0               ; bInitialOwner
.text:00403B29                  push    0               ; lpMutexAttributes
.text:00403B2B                  call    ds:CreateMutexA
.text:00403B31                  mov     [ebp+hObject], eax
.text:00403B34                  push    40h
.text:00403B36                  push    0
```

The first important part we we find here is "**CreateMutexA**" which means that the malware creates a Mutual Exclusion object that ensures that only a single instance of malware is running on a system at a given time. It's a good host-based indictator. If somehow it is not able to create a Mutex object,it closes the handle and the process exits.
Then it starts getting the Pid of several processes and also checks if it is running in a 64-bit OS by calling "**IsWow64Process**". It then starts searching for the path of Internet Explorer and creates a handle of "iexplore.exe", apparently it is a suspended process.We see a method call "WriteProcessMemory" which means it is writing the memory contents of a file in a process.So how about we say that "win33.exe" is injecting its code in "iexplore.exe", signs of a covert/stealth malware.

```
.text:00403A87                  push    eax             ; lpStartAddress
.text:00403A88                  push    0               ; dwStackSize
.text:00403A8A                  push    0               ; lpThreadAttributes
.text:00403A8C                  mov     ecx, [ebp+hProcess]
.text:00403A8F                  push    ecx             ; hProcess
.text:00403A90                  call    ds:CreateRemoteThread
.text:00403A96                  test    eax, eax
.text:00403A98                  jz      short loc_403AA8
.text:00403A9A                  cmp     [ebp+arg_0], 0
.text:00403A9E                  jnz     short loc_403AA8
.text:00403AA0                  push    0               ; uExitCode
.text:00403AA2                  call    ds:ExitProcess
.text:00403AA8
```

Our Doubts are further clarified when we see "**CreateRemoteThread"** being called and then "**ExitProcess"** on Internet Explorer.

We now go inside iexplore to see what changes does it make to the system, probably to find things like registry changes, Java Security Plugin etc.

```
.text:00403185                  push    0               ; lpClass
.text:00403187                  push    0               ; Reserved
.text:00403189                  push    offset aSoftwareHelp_0 ; "Software\\HelperSolutions Software"
.text:0040318E                  push    80000001h       ; hKey
.text:00403193                  call    ds:RegCreateKeyExA
.text:00403199                  cmp     [ebp+Type], 1
.text:0040319D                  jnz     short loc_4031B0
.text:0040319F                  mov     edx, [ebp+hKey]
.text:004031A2                  push    edx             ; hKey
.text:004031A3                  call    ds:RegCloseKey
.text:004031A9                  xor     eax, eax
.text:004031AB                  jmp     loc_403248
.text:004031B0 ; ---------------------------------------------------------------------------
```

It create a new Registry subkey called “HelperSolutions Software” and also creates a New Folder

```
.text:004032B6                  push    offset aJavaSecurityPl ; "Java Security Plugin"
.text:004032BB                  lea     ecx, [ebp+var_228]
.text:004032C1                  push    ecx
.text:004032C2                  push    offset aSS      ; "%s\\%s"
.text:004032C7                  push    offset word_409B00 ; LPWSTR
.text:004032CC                  call    ds:wsprintfW
.text:004032D2                  add     esp, 10h
.text:004032D5                  push    0               ; lpSecurityAttributes
.text:004032D7                  push    offset word_409B00 ; lpPathName
.text:004032DC                  call    ds:CreateDirectoryW
.text:004032E2                  push    offset aJavaplugin ; "javaplugin"
.text:004032E7                  push    offset aJavaSecurity_0 ; "Java Security Plugin"
.text:004032EC                  lea     edx, [ebp+var_228]
.text:004032F2                  push    edx
.text:004032F3                  push    offset aSSS_exe ; "%s\\%s\\%s.exe"
.text:004032F8                  push    offset word_409B00 ; LPWSTR
.text:004032FD                  call    ds:wsprintfW
.text:00403303                  add     esp, 14h
.text:00403306                  push    0               ; bFailIfExists
.text:00403308                  push    offset word_409B00 ; lpNewFileName
.text:0040330D                  push    offset ExistingFileName ; lpExistingFileName
.text:00403312                  call    ds:CopyFileW
```

called **Java Security Plugin** and creates a file called “**javaplugin.exe**”, we see that it also does a DeletFile. We observed in our Dynamic Analysis that our orignal win33.exe got deleted and in its place came SecureDll.dll, we still don’t know it was actually deleted or replaced.

```
.text:00403306                  push    0               ; bFailIfExists
.text:00403308                  push    offset word_409B00 ; lpNewFileName
.text:0040330D                  push    offset ExistingFileName ; lpExistingFileName
.text:00403312                  call    ds:CopyFileW
.text:00403318
.text:00403318 loc_403318:                              ; CODE XREF: sub_403250+E2↓j
.text:00403318                  push    offset ExistingFileName ; lpFileName
.text:0040331D                  call    ds:DeleteFileW
.text:00403323                  test    eax, eax
.text:00403325                  jnz     short loc_403334
.text:00403327                  push    3E8h            ; dwMilliseconds
.text:0040332C                  call    ds:Sleep
.text:00403332                  jmp     short loc_403318
```

We also see that the main file was copied before being deleted, don’t know yet where it is copied. Furthermore we see a Registry data being persisted on “Sun Java Security Plugin” with **RegSetValueExA**

Moreover, we see more Registry Instances being created, set and closed on **LowRiskFileTypes** like .reg, .exe, .vbs, .bat

```
.text:00403524                push    ecx             ; phkResult
.text:00403525                push    0               ; lpSecurityAttributes
.text:00403527                push    0F003Fh         ; samDesired
.text:0040352C                push    0               ; dwOptions
.text:0040352E                push    0               ; lpClass
.text:00403530                push    0               ; Reserved
.text:00403532                push    offset aSoftwareMicr_0 ; "Software\\Microsoft\\Windows\\CurrentVersi"...
.text:00403537                push    80000001h       ; hKey
.text:0040353C                call    ds:RegCreateKeyExA
.text:00403542                push    offset a_exe_bat_reg_v ; ".exe;.bat;.reg;.vbs;"
.text:00403547                call    ds:lstrlenA
.text:0040354D                mov     [ebp+cbData], eax
.text:00403550                mov     edx, [ebp+cbData]
.text:00403553                push    edx             ; cbData
.text:00403554                push    offset a_exe_bat_reg_v ; ".exe;.bat;.reg;.vbs;"
.text:00403559                push    1               ; dwType
.text:0040355B                push    0               ; Reserved
.text:0040355D                push    offset aLowriskfiletyp ; "LowRiskFileTypes"
.text:00403562                mov     eax, [ebp+hKey]
.text:00403565                push    eax             ; hKey
.text:00403566                call    ds:RegSetValueExA
.text:0040356C                mov     ecx, [ebp+hKey]
.text:0040356F                push    ecx             ; hKey
.text:00403570                call    ds:RegCloseKey
.text:00403576                lea     edx, [ebp+hKey]
.text:00403579                push    edx             ; phkResult
.text:0040357A                push    0F003Fh         ; samDesired
.text:0040357F                push    0               ; ulOptions
.text:00403581                push    offset aSoftwareMicr_1 ; "Software\\Microsoft\\Windows\\CurrentVersi"...
.text:00403586                push    80000001h       ; hKey
.text:0040358B                call    ds:RegOpenKeyExA
```

We find that it creates more threads and also tries to find data from the resource section using **LoadResource,** which we found in our Initial Triage

```
.text:004057D6                push    3               ; lpType
.text:004057D8                push    offset a1       ; "1"
.text:004057DD                mov     eax, hModule
.text:004057E2                push    eax             ; hModule
.text:004057E3                call    ds:FindResourceA
.text:004057E9                mov     [ebp+hResInfo], eax
.text:004057EC                mov     ecx, [ebp+hResInfo]
.text:004057EF                push    ecx             ; hResInfo
.text:004057F0                mov     edx, hModule
.text:004057F6                push    edx             ; hModule
.text:004057F7                call    ds:SizeofResource
.text:004057FD                mov     [ebp+nNumberOfBytesToWrite], eax
.text:00405800                mov     eax, [ebp+hResInfo]
.text:00405803                push    eax             ; hResInfo
.text:00405804                mov     ecx, hModule
.text:0040580A                push    ecx             ; hModule
.text:0040580B                call    ds:LoadResource
.text:00405811                mov     [ebp+hResData], eax
.text:00405814                mov     edx, [ebp+hResData]
.text:00405817                push    edx             ; hResData
.text:00405818                call    ds:LockResource
.text:0040581E                mov     [ebp+var_10], eax
.text:00405821                push    4               ; flProtect
.text:00405823                push    1000h           ; flAllocationType
.text:00405828                mov     eax, [ebp+nNumberOfBytesToWrite]
.text:0040582B                imul    eax, 3
.text:0040582E                push    eax             ; dwSize
.text:0040582F                push    0               ; lpAddress
.text:00405831                call    ds:VirtualAlloc
.text:00405837                mov     [ebp+lpAddress], eax
.text:0040583A                lea     ecx, [ebp+nNumberOfBytesToWrite]
```

We finally found that, the binary data from the Resources section was **SecureDll.dll** and the main executable loaded it before it deleted itself.Eventually after loading this file, strokes.log and tmp.log were also created in the same Directory as SecureDll.dll
There is one debug.log created as well which we will see later

```
.text:00405841              push    edx
.text:00405842              mov     eax, [ebp+var_10]
.text:00405845              push    eax
.text:00405846              mov     ecx, [ebp+nNumberOfBytesToWrite]
.text:00405849              imul    ecx, 3
.text:0040584C              push    ecx
.text:0040584D              mov     edx, [ebp+lpAddress]
.text:00405850              push    edx
.text:00405851              push    102h
.text:00405856              call    dword_409F74
.text:0040585C              push    0               ; hTemplateFile
.text:0040585E              push    2               ; dwFlagsAndAttributes
.text:00405860              push    2               ; dwCreationDisposition
.text:00405862              push    0               ; lpSecurityAttributes
.text:00405864              push    0               ; dwShareMode
.text:00405866              push    10000000h       ; dwDesiredAccess
.text:0040586B              push    offset aSecuredll_dll ; "SecureDll.dll"
.text:00405870              call    ds:CreateFileA
.text:00405876              mov     [ebp+hObject], eax
.text:00405879              mov     [ebp+NumberOfBytesWritten], 0
.text:00405880              push    0               ; lpOverlapped
.text:00405882              lea     eax, [ebp+NumberOfBytesWritten]
.text:00405885              push    eax             ; lpNumberOfBytesWritten
.text:00405886              mov     ecx, [ebp+nNumberOfBytesToWrite]
.text:00405889              push    ecx             ; nNumberOfBytesToWrite
.text:0040588A              mov     edx, [ebp+lpAddress]
.text:0040588D              push    edx             ; lpBuffer
.text:0040588E              mov     eax, [ebp+hObject]
.text:00405891              push    eax             ; hFile
.text:00405892              call    ds:WriteFile
```

Loading Library:

```
.text:004058C9              call    sub_4057D0
.text:004058CE              push    offset aSecuredll_dl_0 ; "SecureDll.dll"
.text:004058D3              call    ds:LoadLibraryA
.text:004058D9              mov     [ebp+hmod], eax
.text:004058DF              mov     [ebp+lpProcName], 1
.text:004058E6              mov     eax, [ebp+lpProcName]
.text:004058E9              push    eax             ; lpProcName
.text:004058EA              mov     ecx, [ebp+hmod]
.text:004058F0              push    ecx             ; hModule
.text:004058F1              call    ds:GetProcAddress
.text:004058F7              mov     [ebp+lpfn], eax
.text:004058FD              mov     [ebp+lpProcName], 2
.text:00405904              mov     edx, [ebp+lpProcName]
.text:00405907              push    edx             ; lpProcName
.text:00405908              mov     eax, [ebp+hmod]
.text:0040590E              push    eax             ; hModule
.text:0040590F              call    ds:GetProcAddress
.text:00405915              mov     [ebp+var_110], eax
.text:0040591B              push    0FFh
.text:00405920              push    0
.text:00405922              lea     ecx, [ebp+Buffer]
.text:00405928              push    ecx
.text:00405929              call    sub_401200
.text:0040592E              add     esp, 0Ch
.text:00405931              push    0FFh
.text:00405936              push    0
```

strokes.log and tmp.log

```
.text:00405960              push    0FFh            ; nBufferLength
.text:00405965              call    ds:GetCurrentDirectoryA
.text:0040596B              push    offset aStrokes_log ; "strokes.log"
.text:00405970              lea     eax, [ebp+Buffer]
.text:00405976              push    eax
.text:00405977              push    offset aSS_0    ; "%s\\%s"
.text:0040597C              push    offset FileName ; LPSTR
.text:00405981              call    ds:wsprintfA
.text:00405987              add     esp, 10h
.text:0040598A              push    offset aTmp_log ; "tmp.log"
.text:0040598F              lea     ecx, [ebp+Buffer]
.text:00405995              push    ecx
.text:00405996              push    offset aSS_1    ; "%s\\%s"
.text:0040599B              push    offset byte_4099E0 ; LPSTR
.text:004059A0              call    ds:wsprintfA
.text:004059A6              add     esp, 10h
.text:004059A9              push    1               ; dwType
.text:004059AB              push    offset FileName ; lpString
.text:004059B0              call    ds:lstrlenA
.text:004059B6              push    eax             ; cbData
.text:004059B7              push    offset FileName ; lpData
.text:004059BC              push    offset aVal1    ; "val1"
.text:004059C1              call    sub_401DA0
.text:004059C6              add     esp, 10h
.text:004059C9              push    1               ; dwType
.text:004059CB              push    offset byte_4099E0 ; lpString
.text:004059D0              call    ds:lstrlenA
.text:004059D6              push    eax             ; cbData
.text:004059D7              push    offset byte_4099E0 ; lpData
.text:004059DC              push    offset aVal2    ; "val2"
```

```
.text:00405A5C                 mov     edx, [ebp+arg_4]
.text:00405A5F                 push    edx
.text:00405A60                 lea     eax, [ebp+Buffer]
.text:00405A63                 push    eax
.text:00405A64                 mov     ecx, [ebp+arg_0]
.text:00405A67                 push    ecx
.text:00405A68                 call    sub_401470
.text:00405A6D                 add     esp, 10h
.text:00405A70                 push    0               ; dwErrCode
.text:00405A72                 call    ds:SetLastError
.text:00405A78                 push    0               ; hTemplateFile
.text:00405A7A                 push    2               ; dwFlagsAndAttributes
.text:00405A7C                 push    4               ; dwCreationDisposition
.text:00405A7E                 push    0               ; lpSecurityAttributes
.text:00405A80                 push    0               ; dwShareMode
.text:00405A82                 push    10000000h       ; dwDesiredAccess
.text:00405A87                 push    offset aDebug_log_0 ; "debug.log"
.text:00405A8C                 call    ds:CreateFileA
.text:00405A92                 mov     [ebp+hObject], eax
.text:00405A95                 call    ds:GetLastError
.text:00405A9B                 cmp     eax, 0B7h
.text:00405AA0                 jz      short loc_405ABF
.text:00405AA2                 mov     [ebp+NumberOfBytesWritten], 0
.text:00405AA9                 push    0               ; lpOverlapped
.text:00405AAB                 lea     edx, [ebp+NumberOfBytesWritten]
.text:00405AAE                 push    edx             ; lpNumberOfBytesWritten
.text:00405AAF                 push    4               ; nNumberOfBytesToWrite
.text:00405AB1                 lea     eax, [ebp+Buffer]
.text:00405AB4                 push    eax             ; lpBuffer
.text:00405AB5                 mov     ecx, [ebp+hObject]
```

We also find that a function “**SetWindowsHookEx”** is being used, which is a hook function which is called whenever a certain event is called.It is commonly used with keyloggers and spyware, which also provides an easy way to load a DLL into all GUI processes. So it is very likely that strokes.log means “Keystrokes.log” and is catching the keyboard inputs from the user.

Futher down, we saw that there are several strings from the assembly code like “update-, checkin, download, uninstall, scanin-” . I am not sure what exactly they are but I think they might be responses from the C&C server.

In the end of the .text section , we find that tons of functions that were used thorughout were mentioned.

We also found that the malware tries to connect to C&C server

7	5.349118677	192.168.56.102	151.248.115.107	TCP	66 1031 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
8	8.348413639	192.168.56.102	151.248.115.107	TCP	66 [TCP Retransmission] 1031 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
9	14.351054468	192.168.56.102	151.248.115.107	TCP	62 [TCP Retransmission] 1031 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
10	25.936636115	192.168.56.102	192.168.56.101	DNS	76 Standard query 0x5801 A dns.msftncsi.com
11	25.936671846	192.168.56.101	192.168.56.102	ICMP	104 Destination unreachable (Port unreachable)
12	25.937181727	192.168.56.102	8.8.8.8	DNS	76 Standard query 0x5801 A dns.msftncsi.com
13	26.373684368	192.168.56.102	151.248.115.107	TCP	66 1032 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
14	26.935200450	192.168.56.102	8.8.8.8	DNS	76 Standard query 0x5801 A dns.msftncsi.com
15	28.936450983	192.168.56.102	8.8.8.8	DNS	76 Standard query 0x5801 A dns.msftncsi.com
16	29.389162554	192.168.56.102	151.248.115.107	TCP	66 [TCP Retransmission] 1032 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
17	31.026758769	PcsCompu_6e:4d:68	PcsCompu_99:b1:5f	ARP	42 Who has 192.168.56.102? Tell 192.168.56.101
18	31.027621772	PcsCompu_99:b1:5f	PcsCompu_6e:4d:68	ARP	60 192.168.56.102 is at 08:00:27:99:b1:5f
19	32.938444532	192.168.56.102	8.8.8.8	DNS	76 Standard query 0x5801 A dns.msftncsi.com
20	35.408913718	192.168.56.102	151.248.115.107	TCP	62 [TCP Retransmission] 1032 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1

FINAL THOUGHTS

This was my first Malware analysis report on any malware, which I enjoyed a lot learning from. I learned about many Windows Functions, how a malware can be sneaky or covert by hiding some of its part in it's resource section. In future, I hope to improve my Malware analysis report writing skills and make it easy to understand for Non-Technical readers and sat the same time not boring enough for other Malware Analysts and Researchers, Threat Hunters to read through

This malware did not require any reversing, just normal Code analysis and Dynamic Analysis, it did not have any code obfuscation or even the malware wasn't packed. This made it not a very tough task to analyze this sample.I did not write signatures for this sample as I am not very well versed in it, but I am planning on writing signatures for this sample and various other samples using YARA in the future.

This is a field I enjoy a lot learning about and will keep on learning more.The next sample I will be working on is **Reveton**.

Thanks for Reading