Malware Analysis Report: “Practical1.exe”

CAP6137 Malware Reverse Engineering: P0x01

Naman Arora

naman.arora@ufl.edu

February 19, 2021

Contents

1 Executive Summary

The acquired malware binary is a very sophisticated malware sample. It possibly is from the malware family *Carberp* from 2009 and has possible functionalities of mass infection like BotNets, keylogging, browser User agents, and banking data exfiltration. The malware, on execution, copies itself to a location where it gains persistance over reboots. It injects itself to benign services memory regions in *Windows Kernel Space*. Statically analyzing the sample does not lead to much gains so dynamic analysis is more fruitful. It exhibits a very effective obfuscation behavior possibly via code compression which it de-compresses at run time. Other obfuscation methods come into play when it tries to exfiltrate gathered information to its *C2*. All of the information sent is possibly encrypted.

On execution, it decompresses itself, copies itself for persistance and starts a *'svchost'* process where it injects itself. The parent process exists resulting an unlinked *'svchost'* process. The malware then possibly hooks into keylogging API and listens for other as of yet unidentified events. After event logging, it sends back the information via HTTP POST API to its *C2*.

2 Static Analysis

2.1 Basic Identification

The provided malware binary sample has the following information in the binary,

Attribute	Value
Bits	32
Endianess	Little
Operating System	Microsoft Windows
Class	Portable Executable (PE)
Subsystem	Microsoft Windows GUI
Size	157184 bytes
Compiler Timestamp	Wed, Nov 19, 2008, 20:24:19
Debugger Timestamp	Sun, June 19, 2011, 17:31:54
SHA256 Hash	a67a1ca66f666eabef466bd6beba25867fd67ba697c1c7c02cde2c51e4e8289d

2.2 Malware Sample Family Identification

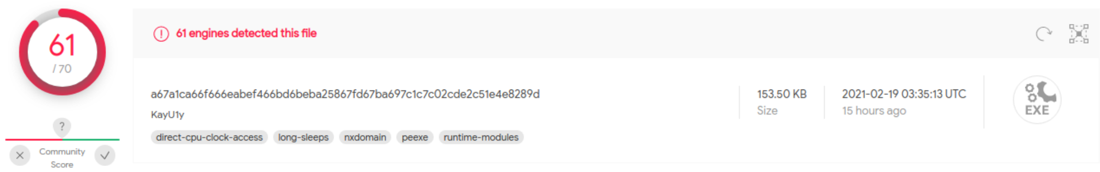


Figure 1: Virustotal Detection

When searched using the SHA256 hash of the sample on *www.virustotal.com* for any prior detections, an overwhelming majority of services detect the sample as malicious executable, *61/70* to be exact. The sample is linked to *Carberp* [1] family of trojans which was infamous for attacking banking systems during 2009.

2.3 PE Sections

The PE has ***five identifiable*** sections (Fig. 2).

2.3.1 The Text section

The text section is supposed to contain the executable code within a binary. While the given binary shows expected permissions of Read and Execute, it shows an unusually high entropy of 7.72 which should be noted and will be referenced in later sections. The size and virtual size of the section are exactly the same, *i.e., 0x13000*.

```
[0x00411710]> iS entropy
[Sections]

nth paddr          size vaddr          vsize perm entropy    name
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
0   0x00000400  0x13000 0x00401000  0x13000 -r-x 7.72730808 .text
1   0x00013400   0xda00 0x00414000   0xe000 -r-- 7.91970334 .rdata
2   0x00020e00   0x4e00 0x00422000  0x2c000 -rw- 7.86249047 .data
3   0x00025c00    0x400 0x0044e000   0x1000 -r-- 2.72709660 .rsrc
4   0x00026000    0x600 0x0044f000   0x1000 -r-- 5.26752657 .reloc
```

Figure 2: Rizin: PE Sections with Entropy

2.3.2 The Rdata Section

The Rdata section contains read only data which the binary needs. This could be global constants and read only data. The Rdata section too shows unusually high entropy of 7.91 which hints more towards compressed or encrypted data. The virtual size of this section *0xe000* is slightly larger than static size *0xda00* which is common. Permissions on this section are as expected, *i.e.*, Read only.

2.3.3 The Data Section

The data section contains writable data for the binary. The permissions on this section, too, are as expected. This section too shows an unusually high entropy of 7.86. Moreover, the virtual size of this section is around 8x times more than the static size, which is very unusual. Overall, this hints towards compressed data.

2.3.4 The Resource and Re-allocation Sections

Both of these sections have a virtual size of *0x1000* with static sizes of *0x400* and *0x600* respectively. The entropies and the size inflation is not too out of the ordinary.

2.4 A case for Compression

The given malware section exhibits unusually high entropy for the first four sections. Generally, such a high entropy is associated with either of the three obfuscation techniques *viz.* Compression, Encryption or Packing. Given the static analysis, packing can be eliminated with relatively high degree of confidence since the binary also shows a multitude of embedded strings, library calls and imports. As for encryption versus compression, three imports, in particular, indicate usage of LZMA [2] compression algorithm (Fig. 3). Moreover, an 8x inflation in size of the *.data* section, as mentioned above, too indicates such a possibility.

```
[0x0044f402]> ii | grep -i lz
14   0x00422034 NONE FUNC KERNEL32.dll LZClose
43   0x004220a8 NONE FUNC KERNEL32.dll LZRead
58   0x004220e4 NONE FUNC KERNEL32.dll LZInit
```

Figure 3: Rizin: Imports from lzexpand.h

2.5 Interesting Imports

2.5.1 WS2_32 and Berkley Socket API

Imports such as *ntohl*, *closesocket*, *setsockopt*, *recvfrom*, *getpeername*, *etc.* strongly indicate an internet based socket activity. Also, the presence of other imports from WS2_32.dll, indicate a strong presence of UDP/connectionless protocol. This could be due to the application querying DNS records.

```
[0x0044f402]> ii | grep -i ws2_32
1    0x00422160 NONE FUNC WS2_32.dll   WSALookupServiceEnd
2    0x00422164 NONE FUNC WS2_32.dll   WSARecvFrom
3    0x00422168 NONE FUNC WS2_32.dll   WSAInstallServiceClassA
4    0x0042216c NONE FUNC WS2_32.dll   WSAAsyncGetServByPort
5    0x00422170 NONE FUNC WS2_32.dll   WSACancelBlockingCall
6    0x00422174 NONE FUNC WS2_32.dll   WSALookupServiceBeginA
7    0x00422178 NONE FUNC WS2_32.dll   ntohl
8    0x0042217c NONE FUNC WS2_32.dll   closesocket
9    0x00422180 NONE FUNC WS2_32.dll   WSAWaitForMultipleEvents
10   0x00422184 NONE FUNC WS2_32.dll   WSCWriteProviderOrder
11   0x00422188 NONE FUNC WS2_32.dll   WSAGetLastError
12   0x0042218c NONE FUNC WS2_32.dll   WSARecv
13   0x00422190 NONE FUNC WS2_32.dll   WSCEnumProtocols
14   0x00422194 NONE FUNC WS2_32.dll   WSAProviderConfigChange
15   0x00422198 NONE FUNC WS2_32.dll   WSAGetServiceClassNameByClassIdA
16   0x0042219c NONE FUNC WS2_32.dll   WSASocketA
17   0x004221a0 NONE FUNC WS2_32.dll   WSASetServiceA
18   0x004221a4 NONE FUNC WS2_32.dll   WSANSPIoctl
19   0x004221a8 NONE FUNC WS2_32.dll   getpeername
20   0x004221ac NONE FUNC WS2_32.dll   WSADuplicateSocketA
21   0x004221b0 NONE FUNC WS2_32.dll   WSCGetProviderPath
22   0x004221b4 NONE FUNC WS2_32.dll   getnameinfo
23   0x004221b8 NONE FUNC WS2_32.dll   recvfrom
24   0x004221bc NONE FUNC WS2_32.dll   WSAConnect
25   0x004221c0 NONE FUNC WS2_32.dll   getprotobyname
26   0x004221c4 NONE FUNC WS2_32.dll   WSALookupServiceNextA
27   0x004221c8 NONE FUNC WS2_32.dll   WSASendDisconnect
28   0x004221cc NONE FUNC WS2_32.dll   ioctlsocket
29   0x004221d0 NONE FUNC WS2_32.dll   WSAAsyncGetHostByName
30   0x004221d4 NONE FUNC WS2_32.dll   WSApSetPostRoutine
31   0x004221d8 NONE FUNC WS2_32.dll   WSAStringToAddressA
32   0x004221dc NONE FUNC WS2_32.dll   WSAIoctl
33   0x004221e0 NONE FUNC WS2_32.dll   setsockopt
34   0x004221e4 NONE FUNC WS2_32.dll   freeaddrinfo
```

Figure 4: Rizin: Imports WS2_32.dll

2.5.2 Wininet API

```
[0x0044f402]> ii | grep -i wininet
1   0x00422300 NONE FUNC WININET.dll  FindFirstUrlCacheContainerW
2   0x00422304 NONE FUNC WININET.dll  InternetSetCookieExW
3   0x00422308 NONE FUNC WININET.dll  InternetCloseHandle
4   0x0042230c NONE FUNC WININET.dll  InternetSecurityProtocolToStringW
5   0x00422310 NONE FUNC WININET.dll  InternetTimeFromSystemTime
6   0x00422314 NONE FUNC WININET.dll  UrlZonesDetach
7   0x00422318 NONE FUNC WININET.dll  InternetCheckConnectionA
8   0x0042231c NONE FUNC WININET.dll  SetUrlCacheEntryGroupW
9   0x00422320 NONE FUNC WININET.dll  ForceNexusLookup
10  0x00422324 NONE FUNC WININET.dll  LoadUrlCacheContent
11  0x00422328 NONE FUNC WININET.dll  SetUrlCacheEntryInfoA
12  0x0042232c NONE FUNC WININET.dll  HttpOpenRequestA
13  0x00422330 NONE FUNC WININET.dll  InternetSetOptionExW
14  0x00422334 NONE FUNC WININET.dll  FtpDeleteFileA
15  0x00422338 NONE FUNC WININET.dll  InternetQueryOptionW
16  0x0042233c NONE FUNC WININET.dll  InternetWriteFileExW
17  0x00422340 NONE FUNC WININET.dll  FtpGetFileEx
18  0x00422344 NONE FUNC WININET.dll  InternetEnumPerSiteCookieDecisionA
19  0x00422348 NONE FUNC WININET.dll  RunOnceUrlCache
20  0x0042234c NONE FUNC WININET.dll  InternetConfirmZoneCrossingA
21  0x00422350 NONE FUNC WININET.dll  InternetAlgIdToStringA
22  0x00422354 NONE FUNC WININET.dll  HttpQueryInfoW
23  0x00422358 NONE FUNC WININET.dll  InternetCheckConnectionW
24  0x0042235c NONE FUNC WININET.dll  InternetInitializeAutoProxyDll
25  0x00422360 NONE FUNC WININET.dll  InternetSetDialStateW
26  0x00422364 NONE FUNC WININET.dll  IsHostInProxyBypassList
27  0x00422368 NONE FUNC WININET.dll  CommitUrlCacheEntryW
28  0x0042236c NONE FUNC WININET.dll  GetUrlCacheEntryInfoExW
29  0x00422370 NONE FUNC WININET.dll  FindFirstUrlCacheEntryW
30  0x00422374 NONE FUNC WININET.dll  InternetGetLastResponseInfoW
31  0x00422378 NONE FUNC WININET.dll  InternetGetCookieExA
32  0x0042237c NONE FUNC WININET.dll  RetrieveUrlCacheEntryStreamA
33  0x00422380 NONE FUNC WININET.dll  UnlockUrlCacheEntryFile
34  0x00422384 NONE FUNC WININET.dll  GopherCreateLocatorA
35  0x00422388 NONE FUNC WININET.dll  InternetCreateUrlA
```

Figure 5: Rizin: Imports Wininet.dll

- HTTP
  Imports such as *HttpQueryInfo, HttpOpenRequest, etc.* strongly indicate towards HTTP related activity.
- FTP
  Imports such as *FtpGetFileEx, FtpDeleteFileA, etc.* strongly indicate towards read and write activity to a remote server over FTP protocol.
- Browser Related Activity
  Imports such as *InternetSetCookieExW, LoadUrlCacheContent, etc.* suggest some interaction with browser cache.

Even though these imports strongly suggest their mentioned intentions, no strings that point to any remote address have been located in the binary statically. This makes the case for compression based obfuscation stronger.

2.5.3 Other Miscallineous Imports

```
[0x0044f402]> ii | grep -i thread
2   0x00422004 NONE FUNC KERNEL32.dll SetThreadPriorityBoost
3   0x00422008 NONE FUNC KERNEL32.dll OpenThread
```

Figure 6: Rizin: Imports Threading

```
[0x0044f402]> ii | grep -i dir
1   0x00422000 NONE FUNC KERNEL32.dll SetCurrentDirectoryW
12  0x0042202c NONE FUNC KERNEL32.dll RemoveDirectoryA
15  0x00422038 NONE FUNC KERNEL32.dll CreateDirectoryExA
68  0x0042210c NONE FUNC KERNEL32.dll GetWindowsDirectoryW
[0x0044f402]> ii | grep -i file
21  0x00422050 NONE FUNC KERNEL32.dll LockFileEx
57  0x004220e0 NONE FUNC KERNEL32.dll GetFileInformationByHandle
74  0x00422124 NONE FUNC KERNEL32.dll DeleteFileA
75  0x00422128 NONE FUNC KERNEL32.dll GetProfileStringA
85  0x00422150 NONE FUNC KERNEL32.dll SetFilePointer
```

Figure 7: Rizin: Imports File System

Other interesting imports,

- Threading functionality (Fig. 6)
- File System related functionality (Fig. 7)

3 Dynamic Analysis

3.1 Network Based Analysis

3.1.1 External domains contacted

```
>> Matched Request - fromamericawhichlov.com.
>> Matched Request - fromamericawhichlov.com.
>> Matched Request - malborofrientro.com.
>> Matched Request - hillaryklinton.com.
>> Matched Request - fromamericawhichlov.com.
>> Matched Request - malborofrientro.com.
>> Matched Request - hillaryklinton.com.
```

Figure 8: FakeDNS: DNS Domain Lookups

The malware attempts to make a connection to the following domains(Fig. 8),

- fromamericawhichlove.com

- hillaryklinton.com
- malborofrientro.com

3.1.2 Internet Protocols Used

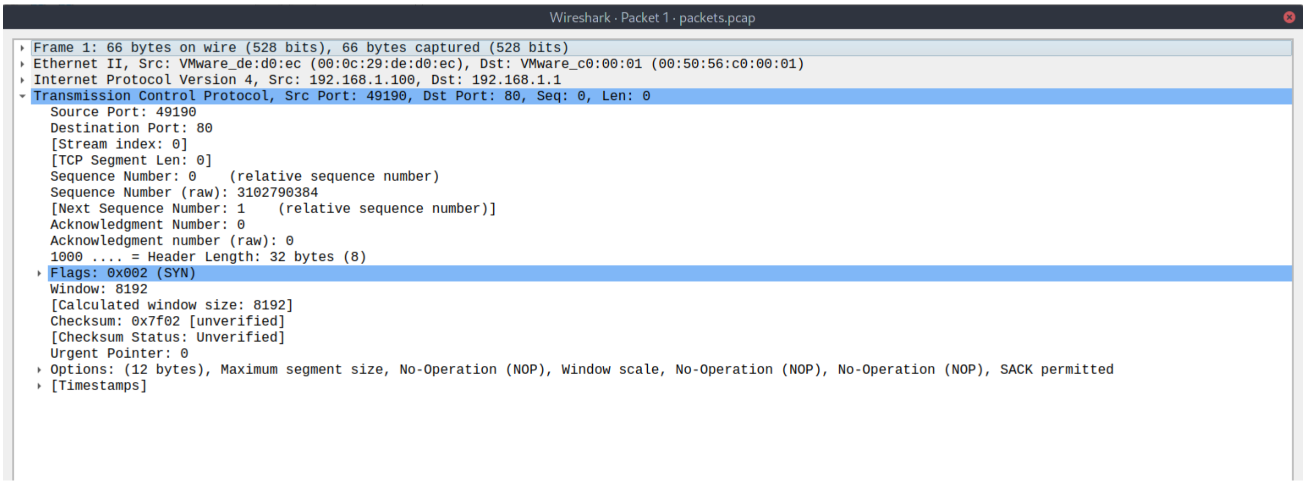


Figure 9: Wireshark: TCP SYN request

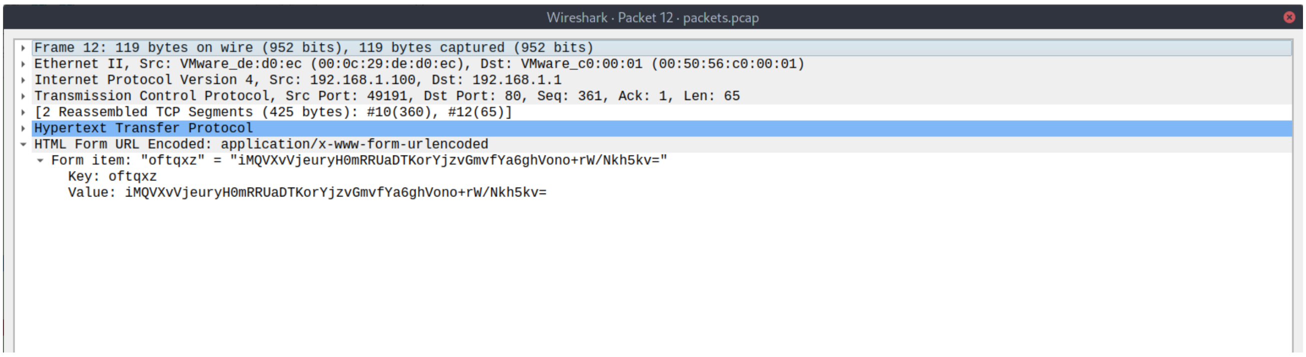


Figure 10: Wireshark: HTTP Post with Base64 Encoded value

The malware on execution uses the following Protocols

- DNS over UDP
  The malware queries for hosts using DNS lookup. The DNS server used is the same as the default on the host system.
- HTTP POST over TCP
  The malware sends multiple POST requests (every 60sec or so) containing base64 encoded files. The files can have extensions *.phtml, .php3, .phtm, .inc, .cgi, .doc, .rtf, .tpl and .rar*. The list of extensions will be proved using memory forensics.

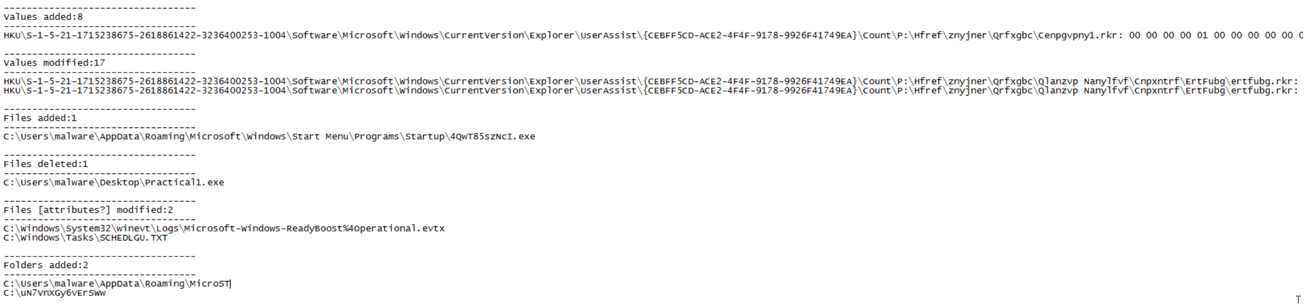
3.1.3 Contents of Communication

The malware communicates information over the internet to previously specified domains using HTTP POST requests. The communication contains a file with key value pair. The *key* is seemingly random while the *value* happens to be a Base64 encoded string. On decoding the *value*, a seemingly encrypted sequence of bytes is obtained.

3.2 File System Based Analysis

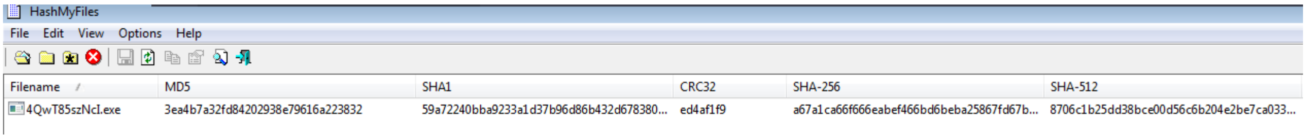
3.2.1 File System Changes

The copy of the binary which is executed at first instantiation is deleted. The malware then achieves persistance by copying itself to the location *"%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\4*
The dropped file has the same hash as the original file (Fig. 12).



```
------------------------------------
Values added:8
------------------------------------
HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Cenpgvpny1.rkr: 00 00 00 00 01 00 00 00 00 00 0

------------------------------------
Values modified:17
------------------------------------
HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Qlanzvp Nanylfvf\Cnpxntrf\ErtFubg\ertfubg.rkr:
HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Qlanzvp Nanylfvf\Cnpxntrf\ErtFubg\ertfubg.rkr:

------------------------------------
Files added:1
------------------------------------
C:\Users\malware\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\4QwT85szNcI.exe

------------------------------------
Files deleted:1
------------------------------------
C:\Users\malware\Desktop\Practical1.exe

------------------------------------
Files [attributes?] modified:2
------------------------------------
C:\Windows\System32\winevt\Logs\Microsoft-Windows-ReadyBoost%4Operational.evtx
C:\Windows\Tasks\SCHEDLGU.TXT

------------------------------------
Folders added:2
------------------------------------
C:\Users\malware\AppData\Roaming\MicroST
C:\uN7VnXGy6vErSWw
```

Figure 11: RegShot: File system Changes



HashMyFiles

File Edit View Options Help

Filename	MD5	SHA1	CRC32	SHA-256	SHA-512
4QwT85szNcI.exe	3ea4b7a32fd84202938e79616a223832	59a72240bba9233a1d37b96d86b432d678380...	ed4af1f9	a67a1ca66f666eabef466bd6beba25867fd67b...	8706c1b25dd38bce00d56c6b204e2be7ca033...

Figure 12: HashMyFiles: Hash of dropped file

The malware creates two directories, *viz.*

- *"%USERPROFILE%\AppData\Roaming\MicroST"*
- *"C:\uN7vnXGy6vErSWw"*

3.2.2 Windows Registry Changes

The malware changes the registry key, *"HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{ ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Qlanzvp Nanylfvf\Cnpxntrf\ErtFubg\ertfubg.rkr"*

The malware creates a registry key, *"HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{ ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Cenpgvpny1.rkr"*

3.3 Memory Forensics

The memory of a running Virtual Machine infected with the malware was dumped and analyzed.

3.3.1 The malicious memory regions

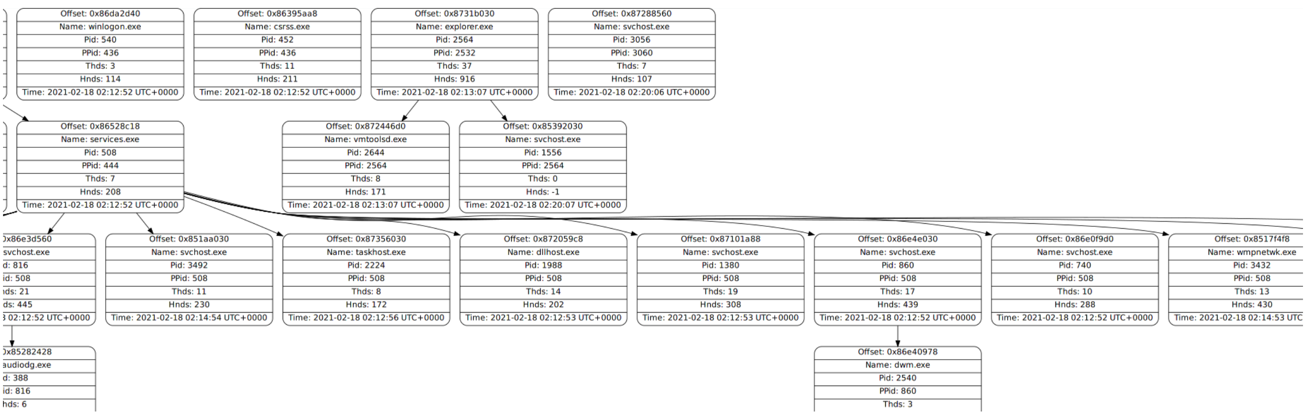


Figure 13: Volatility pstree: Running Process tree

The process tree of the running process in the dumped memory shows *svchost.exe (PID 3056, top right)* being spawned as disjoint from *services.exe (PID 508, left)*. This, in an ordinary system is not possible and hence makes the process *PID 3056* suspicious.

The process *PID 3056* was started with the option *'-k netsvcs'* (Fig. 14).

```
git/SRE-Practical1 [master●] » volatility -f bin/memdump/memory.dmp --profile=Win7SP1x86_23418 cmdline -p 3056
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
svchost.exe pid:   3056
Command line : -k netsvcs
```

Figure 14: Volatility cmdline: Commandline for the PID 3056

```
git/SRE-Practical1 [master●] » volatility -f bin/memdump/memory.dmp --profile=Win7SP1x86_23418 dlllist -p 3056
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
svchost.exe pid:   3056
Command line : -k netsvcs


Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x00120000     0x8000     0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\system32\svchost.exe
```

Figure 15: Volatility dlllist: Path of the executable

Moreover, the executable path corresponds to legitimate path of the process (Fig. 15). Also, the *"ldrmodules"* output is consistent with the *"dlllist"* output. This asserts that none of the libraries that were loaded by this process have been maliciously replaced from their path on the file system or no new library has been loaded without the notice to userspace.

```
SRE-Practical1/vaddump [master●] » strings -f *.dmp | grep -e "fromamerica\|malboro\|hillary"
svchost.exe.bda88560.0x00180000-0x001bffff.dmp: hillaryklinton.com
svchost.exe.bda88560.0x002d0000-0x0030ffff.dmp: fromamericawhichlov.com
svchost.exe.bda88560.0x00380000-0x0047ffff.dmp: Host: fromamericawhichlov.com
svchost.exe.bda88560.0x00380000-0x0047ffff.dmp: Host: fromamericawhichlov.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: hillaryklinton.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: fromamericawhichlov.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: fromamericawhichlov.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: fromamericawhichlov.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: malborofrientro.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: fromamericawhichlov.com
svchost.exe.bda88560.0x01b50000-0x01b8ffff.dmp: fromamericawhichlov.com
```

Figure 16: Volatility vaddump: Running strings on dumped VAD memory nodes

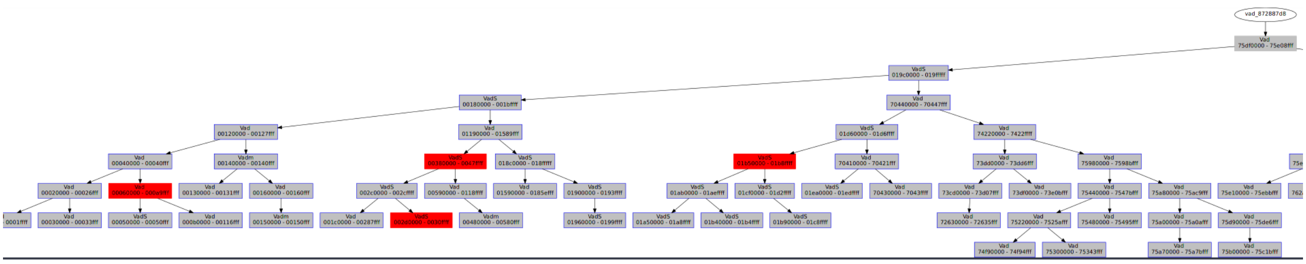
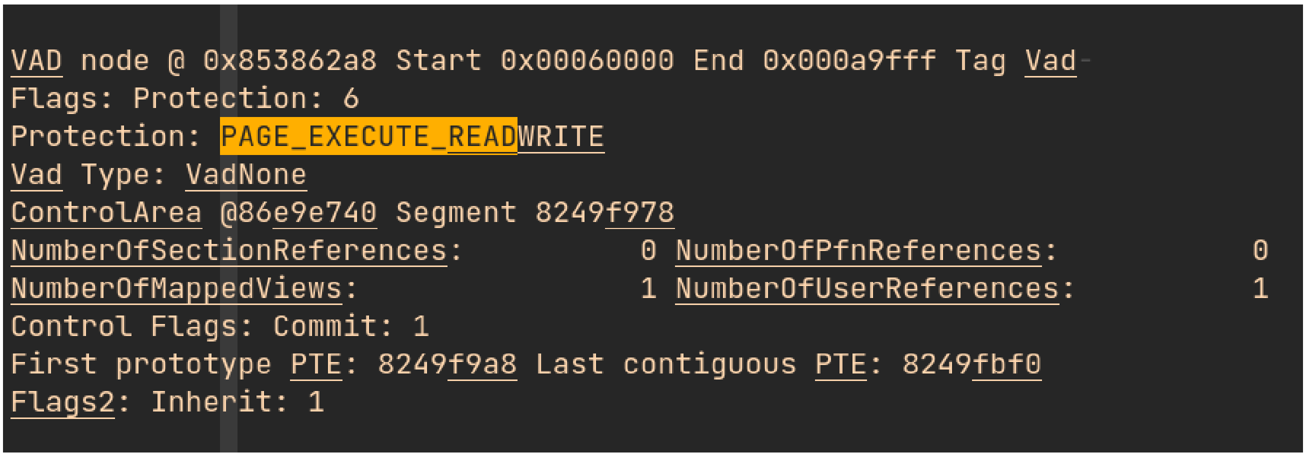


Figure 17: Volatility vadtree: VAD tree of the process with malicious nodes in red

However, on dumping all the memory regions from the *Virtual Address Descriptor (VAD) tree*, malicious memory regions come into view (Fig. 16). A partial *"VAD tree"* of the process *"PID 3056"* is visually represented in Fig. 18.

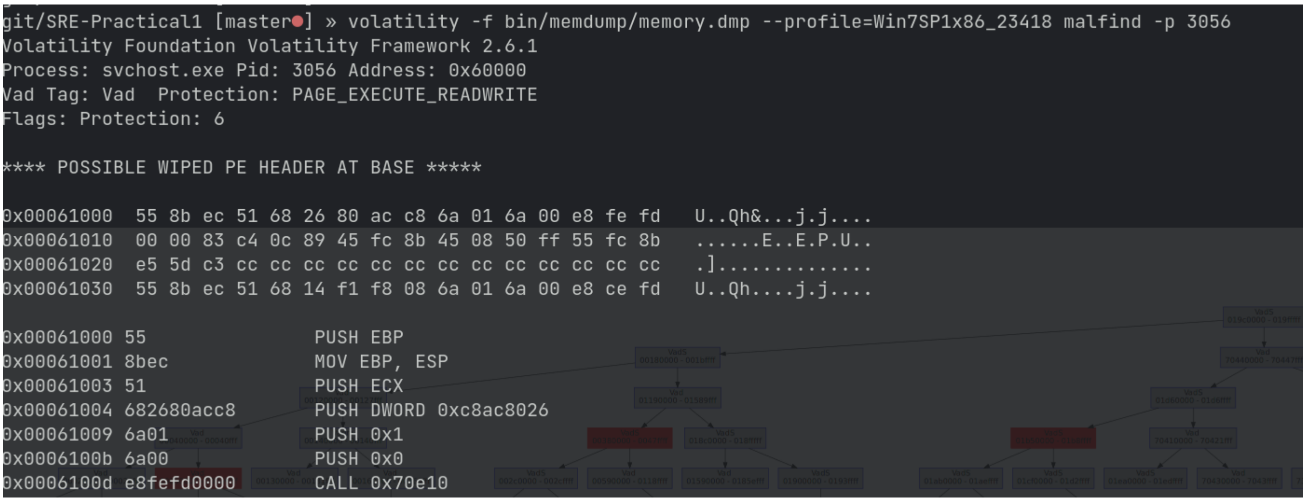
3.3.2 A case for Code Injection

The output to *"vadinfo"* module reveals that a memory region, consistent with previous observations, has a *'PAGE_EXECUTE_READWRITE'* permission. This permission and memory base combination along with a possibly wiped *PE header* is also reported by *'malfind'* plugin, confirming the suspicion of injected code (Fig. 19).



```
VAD node @ 0x853862a8 Start 0x00060000 End 0x000a9fff Tag Vad
Flags: Protection: 6
Protection: PAGE_EXECUTE_READWRITE
Vad Type: VadNone
ControlArea @86e9e740 Segment 8249f978
NumberOfSectionReferences:          0 NumberOfPfnReferences:          0
NumberOfMappedViews:                1 NumberOfUserReferences:         1
Control Flags: Commit: 1
First prototype PTE: 8249f9a8 Last contiguous PTE: 8249fbf0
Flags2: Inherit: 1
```

Figure 18: Volatility vadinfo: Memory section with injected code



```
git/SRE-Practical1 [master●] » volatility -f bin/memdump/memory.dmp --profile=Win7SP1x86_23418 malfind -p 3056
Volatility Foundation Volatility Framework 2.6.1
Process: svchost.exe Pid: 3056 Address: 0x60000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

**** POSSIBLE WIPED PE HEADER AT BASE *****

0x00061000  55 8b ec 51 68 26 80 ac c8 6a 01 6a 00 e8 fe fd   U..Qh&...j.j....
0x00061010  00 00 83 c4 0c 89 45 fc 8b 45 08 50 ff 55 fc 8b   ......E..E.P.U..
0x00061020  e5 5d c3 cc cc cc cc cc cc cc cc cc cc cc cc cc   .]..............
0x00061030  55 8b ec 51 68 14 f1 f8 08 6a 01 6a 00 e8 ce fd   U..Qh....j.j....

0x00061000 55               PUSH EBP
0x00061001 8bec             MOV EBP, ESP
0x00061003 51               PUSH ECX
0x00061004 682680acc8       PUSH DWORD 0xc8ac8026
0x00061009 6a01             PUSH 0x1
0x0006100b 6a00             PUSH 0x0
0x0006100d e8fefd0000       CALL 0x70e10
```

Figure 19: Volatility malfind: Malfind report

```
.phtml
.php3
.phtm
.inc
.cgi
.doc
.rtf
.tpl
.rar
```

Figure 20: The HTTP POST file extensions

3.3.3 Memory region analysis

On analyzing the memory regions, a few following features are observed (shown in Appendix A attached at the end of the report),

- Multiple string references to *"bank"* and a URL confirms to a certain degree that malware is somehow associated with banking applications.
- Multiple string references to browser profiles and *"Micromedia"* imply applications such as browsers and flash player, possibly having to do with internet User Agents.
- A peculiar string with reference to *bot id* hints towards a BotNet like functionality.
- A set of extensions being found in Fig. 20 shows how many extensions can the files sent via HTTP POST have.
- A reference to *"Java"* indicates some java related functionality.
- Some strings, a reference to *'Keylogger.h'* and keylogger thread hint towards keylogger functionality.

4 Indicators of Compromise

4.1 Network Based

The network based indicators is the connection/lookup to the following domains:

- fromamericawhichlove.com
- hillaryklinton.com
- malborofrientro.com

4.2 Host Based

```
git/SRE-Practical1 [master●] » volatility -f bin/memdump/memory.dmp --profile=Win7SP1x86_23418 handles -p 3056 | grep -i mutant
Volatility Foundation Volatility Framework 2.6.1
0x851d77b8    3056       0x74   0x1f0001 Mutant
0x86609df8    3056       0x80   0x1f0001 Mutant           kp_svc_mt
git/SRE-Practical1 [master●] »
```

Figure 21: Volatility handles: Suspicious handle of PID 3056

The host based indicators is the presence of the following,

- *"%USERPROFILE%\AppData\Roaming\MicroST"* directory
- *"C:\uN7vnXGy6vErSWw"* directory
- *"HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserA ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Qlanzvp Nanylfvf\Cnpxntrf\ErtFubg\ertfubg.rkr"* registry key.
- *"HKU\S-1-5-21-1715238675-2618861422-3236400253-1004\Software\Microsoft\Windows\CurrentVersion\Explorer\UserA ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\znyjner\Qrfxgbc\Cenpgvpny1.rkr"* registry key.
- *"%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\4QwT85szNcI.exe"* executable file.
- Mutant handle named *"kp_svc_mt"* (Fig. 21)

5 Appendix A: Memory Dump string analysis screenshots

```
application/x-www-form-urlencoded
multipart/form-data; boundary=
```

```
HELLO\n
%s:%s\n
READY\n
GET /stat?uptime=%d&downlink=%d&uplink=%d&id=%s&statpass=%s&comment=%s HTTP/1.0\r\n\r\n
POST
HEAD
DELETE
LINK
UNLINK
CONNECT
```

```
BBSBank
BBSBank
null
&bal=
http://ibanksystemdwersfssnk.com/boffl.php?uid=
```

```
reboot
: %s
bootkit
  %s
bootkit
3  %s
bootkit
  %s
```

```
%bot_id%
POST
```

```
Url: %s\r\nLogin: %s\r\nPassword: %s\r\nUserAgent: %s\r\n
Information.txt
screen.jpeg
NetInfo.txt
```

```
utf16le Internet Explorer_Server
ascii   Internet Explorer_Server
ascii   Shell DocObject View
utf16le Internet Explorer_Server
```

```
lla\Firefox\Profiles\
cookies.sqlite
sessionstore.*
Macromedia\Flash Player\
*.sol
C:\WINDOWS\system32\Macromed\
cookie:
Cookies\index.dat
Cookies\
*.txt
Cookies\index.dat
%userprofile%
\Cookies\index.dat
cookie:
Vista
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

```
IBank.cpp
%AllUsersProfile%
\uid.txt
ERROR
Java FAIL \r\nReboot is needed!
Patch is completed !
```

```
ascii   [backspace_down]
ascii   [tab_down]
ascii   [enter_down]
```

References

[1] Microsoft. *Win32/Carberp*. `https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=Win32/Carberp`. [Online; accessed 19-Feb-2021]. 2017.

[2] Wikipedia. *Lempel–Ziv–Markov chain algorithm*. `https://en.wikipedia.org/wiki/Lempel-Ziv-Markov_chain_algorithm`. [Online; accessed 19-Feb-2021]. 2017.

```
Downloaded: javassist.jar
java_patcher
/jni.dll
Failed Download jni.dll
java_patcher
jni.dll
CRC32: jni.dll incorrect checksum
java_patcher
%ALLUSERSPROFILE%
\jni.dll
java_patcher
RT PATCH: Can`t create %temp%\jni.dll
java_patcher
java_patcher
RT PATCH: BitesWritten != rtIniLen
java_patcher
Downloaded: jni.dll
java_patcher
Downloading Complete
java_patcher
java_patcher
Clearing tmp folder
java_patcher
DownloadAndSave
java_patcher
java_patcher
DownloadAndSave fail
java_patcher
Clearing tmp folder
```

```
ascii    {Enter}
ascii    {#%0x}
utf16le {Enter}
utf16le {BackSpace}
utf16le {#%0x}
ascii    {Click}
ascii    {RClick}
ascii    %s (x=%d;y=%d)
utf16le {Click}
utf16le {RClick}
utf16le %s (x=%d;y=%d)
ascii    %u,%u,%u,%u
utf16le \Hpr\
utf16le 0x%X
utf16le \hash%u.dats
utf16le \scr%u.jpg
utf16le 0x%u
```

```
utf16le \KYL\
ascii   <HEAD>
utf16le KYL\fi.dat
utf16le KYL\fi.dat
utf16le *.dats
ascii   start thraed for keyloging SendLoadedThred
```

```
ascii   cyberplat
utf16le 0x%u
ascii   [%d,%d,%d,%d]hWnd: %x: WndClass: %s Symbols: %s
ascii   {BackSpace}
utf16le [%d,%d,%d,%d]hWnd: %x: WndClass: %s Symbols: %s
ascii   {Enter}
ascii   {#%0x}
utf16le {Enter}
utf16le {BackSpace}
utf16le {#%0x}
ascii   {Click}
ascii   {RClick}
ascii   %s (x=%d;y=%d)
utf16le {Click}
utf16le {RClick}
utf16le %s (x=%d;y=%d)
ascii   %u,%u,%u,%u
utf16le \Hpr\
utf16le 0x%X
utf16le \hash%u.dats
utf16le \scr%u.jpg
utf16le 0x%u
ascii   %02X
ascii   ind file
ascii   KeyLogger.h
```