Practical1 Report

Name: Rachana Nitinkumar Gugale
UFID: 6353-2454
Email: rgugale@ufl.edu
Date: 18th Feb 2023
Class: CAP6137
Assignment: Practical 1

Executive Summary

The program looks like a **keylogger** and a **password stealer**.

The program is not obfuscated. It uses various functions from USER32.dll like GetKeyboardState, GetKeyState, keybd_event and mouse_event to get the state of the keyboard keys and mouse events. There are plenty of strings like [Enter], [Arrow Left], [Home], etc. present in the program indicating keystrokes and mouse events. The program seems to be stealing login information and passwords from browsers like Firefox, Opera, Chrome and Chromium as is indicated by the following strings present in the program:

%s\Chromium\User Data\Default\Login Data
%s\Mozilla\Firefox\profiles.ini
%s\signons.sqlite
%s\logins.json
select * from moz_logins

The malware has a handle to **\Device\Afd** which it uses for network communication. The program tries to communicate with ***masonchill.jumpingcrab.com*** and ***masonchill.dynamic-dns.net*** via TCP on ports 3360 and 3370. It uses functions from WS2_32.dll like gethostname, socket, send and recv for communicating with what might be a CNC server at these URLs. The malware also uses cryptographic functions like CryptHashData, CryptCreateHash and CryptUnprotectData from ADVAPI32.dll and CRYPT32.dll which suggest that the data it sends to the CNC server might be encrypted.

The program creates and modifies various registry keys but none of them help in figuring out the malware's function.

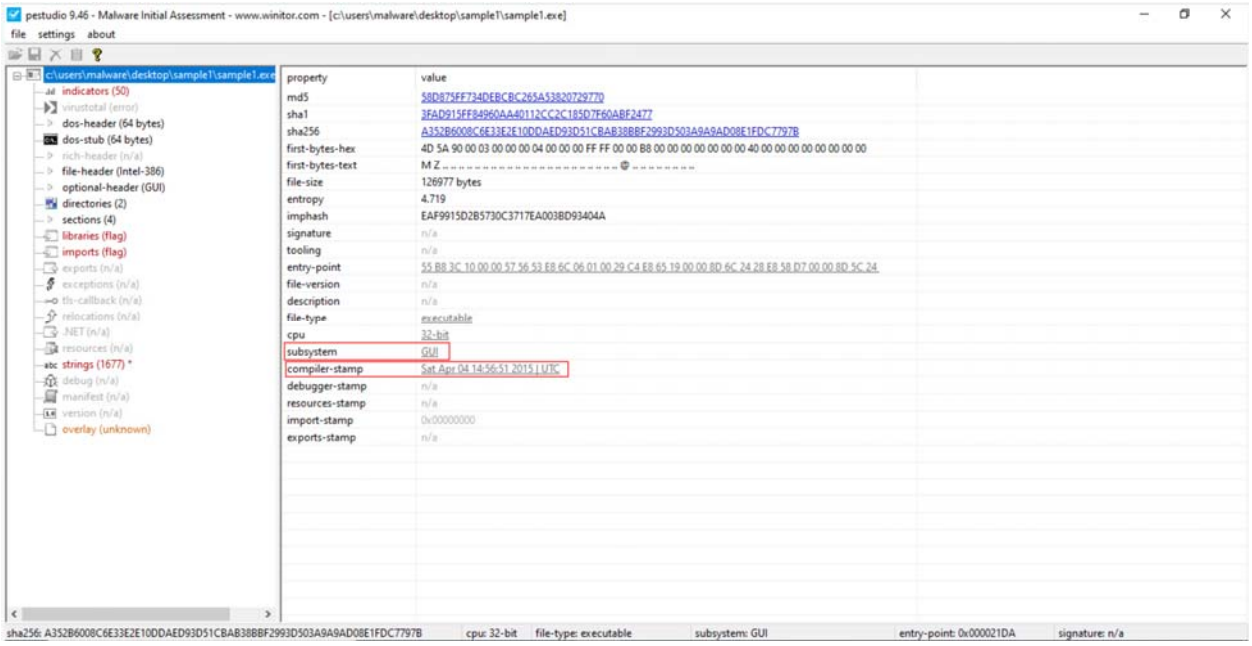
Host-based IPS signature: The MD5 hash value (58D875FF734DEBCBC265A53820729770) of the sample1.exe file could be used to blacklist and delete it.
Host-based IDS signature: Presence of a file called **.Identifier** in the same directory sample1.exe is in.

Static Analysis

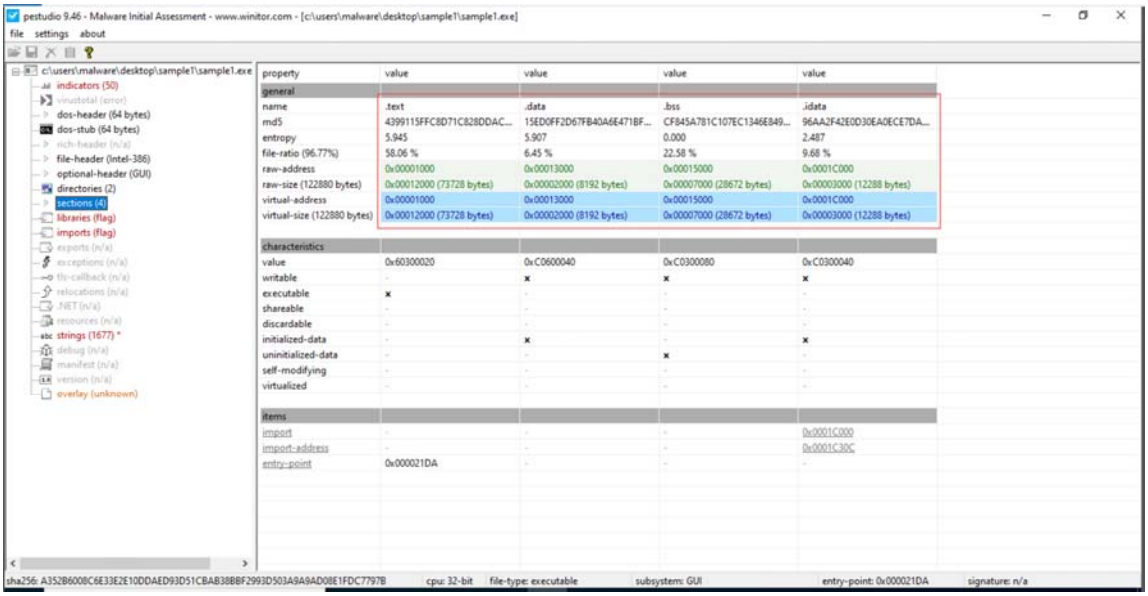
PEStudio Analysis

The following information was uncovered by doing static analysis with PEStudio:

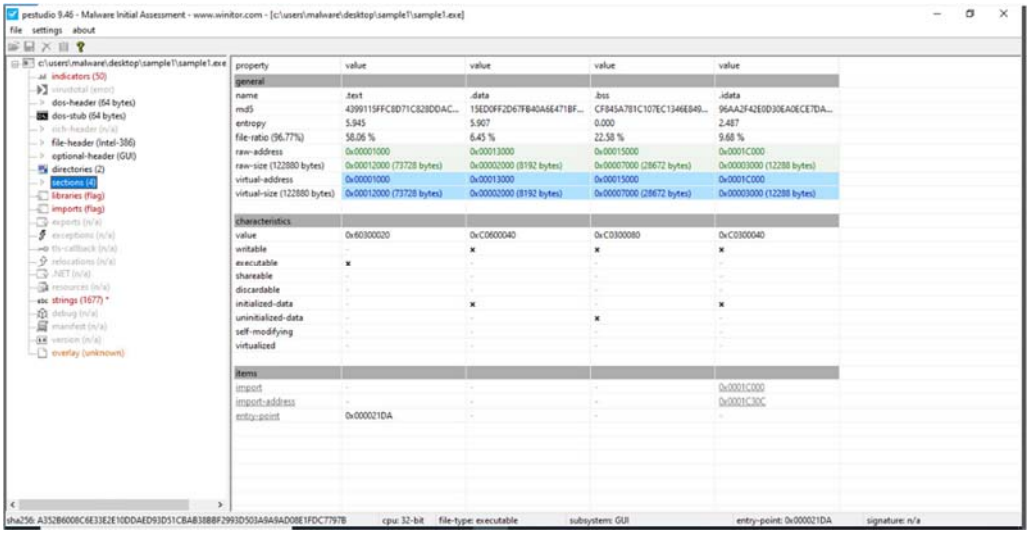
1. **Compilation Date:** Sat Apr 4 2015 at 14:56:51 UTC
2. The malware is a Windows GUI program as can be seen from the “subsystem” value.



3. The PE file contains .text, .data, .bss and .idata sections. Each section contains the following:
    a. **.text:** The text section contains the actual code to be executed.
    b. **.data:** The data section contains variables that have been initialized.
    c. **.bss:** The bss section contains variables that haven’t been initialized.
    d. **.idata:** The idata section contains the import tables.



4. The program is not packed. This conclusion was reached by using the following obfuscation indicators:
    a. **Program signature:** The program signature does not provide any indicators to the malware being packed.
    b. **PE Sections:** The raw and virtual sizes of both .text and .data sections agree with each other and their entropy is between 5 and 6 which is standard for non-obfuscated files. The names of file sections are not mangled providing further evidence that the file is not packed.



    c. **Strings:** The strings are readable and do not appear to be obfuscated.
    d. **Imports and exports:** The malware has a sufficiently large number of imports and exports which again indicates that it is not obfuscated.

Interesting Imports and Strings

- **Imports and strings related to I/O:**
    - The malware seems to be registering input devices with the RegisterRawInputDevices and gets data from these devices using GetRawInputData. It records the state of keyboard keys using various functions like GetKeyState, GetKeyboardState and GetKeyNameText.
    - It emulates mouse events with the mouse_event function.
    - Strings for various non-character keyboard keys like [Backspace], [Arrow Left], [Arrow Right], etc. are present in the program.
    - The presence of these functions and strings strongly suggests that the malware is a **keylogger**.

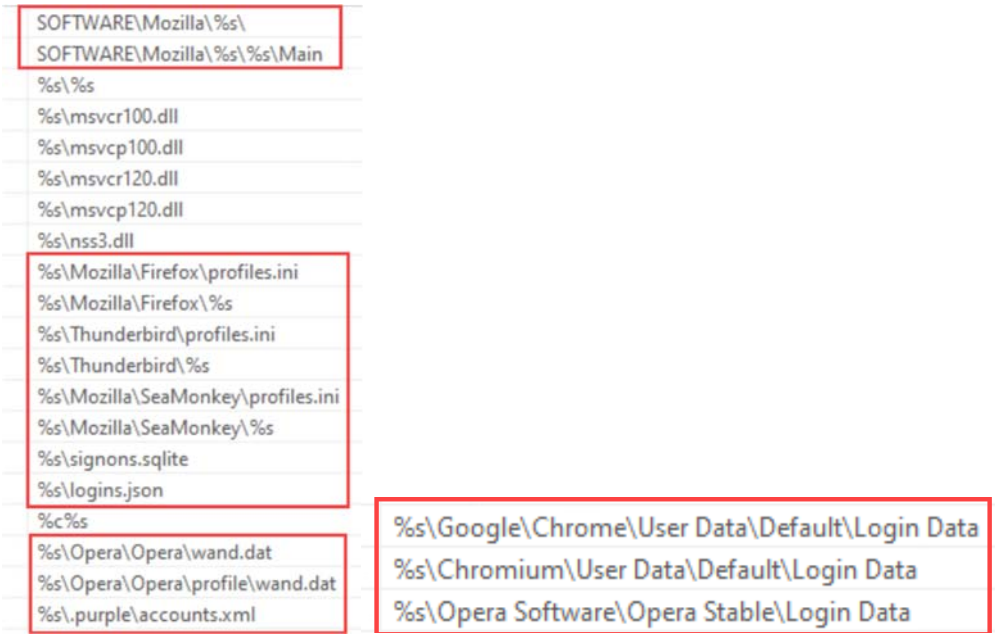
ascii	14	0x0001CC3C	x	import	input-output	GetKeyNameText
ascii	11	0x0001CC4E	x	import	input-output	GetKeyState
ascii	16	0x0001CC5C	x	import	input-output	GetKeyboardState
ascii	13	0x0001CCB6	x	import	input-output	MapVirtualKey
ascii	11	0x0001CD66	x	import	input-output	mouse_event
ascii	23	0x00014770	x	-	input-output	RegisterRawInputDevices
ascii	15	0x00014788	x	-	input-output	GetRawInputData

Figure 1: Suspicious Functions

ascii	11	0x00014694	-	keyboard	-	[Backspace]
ascii	7	0x000146A0	-	keyboard	-	[Enter]
ascii	5	0x000146A8	-	keyboard	-	[Tab]
ascii	12	0x000146AE	-	keyboard	-	[Arrow Left]
ascii	10	0x000146BB	-	keyboard	-	[Arrow Up]
ascii	13	0x000146C6	-	keyboard	-	[Arrow Right]
ascii	12	0x000146D4	-	keyboard	-	[Arrow Down]
ascii	6	0x000146E1	-	keyboard	-	[Home]
ascii	9	0x000146E8	-	keyboard	-	[Page Up]
ascii	11	0x000146F2	-	keyboard	-	[Page Down]
ascii	5	0x000146FE	-	keyboard	-	[End]
ascii	7	0x00014704	-	keyboard	-	[Break]
ascii	8	0x0001470C	-	keyboard	-	[Delete]
ascii	8	0x00014715	-	keyboard	-	[Insert]
ascii	14	0x0001471E	-	keyboard	-	[Print Screen]
ascii	13	0x0001472D	-	keyboard	-	[Scroll Lock]
ascii	11	0x0001473B	-	keyboard	-	[Caps Lock]
ascii	5	0x00014747	-	keyboard	-	[Alt]
ascii	5	0x0001474D	-	keyboard	-	[Esc]
ascii	9	0x00014753	-	keyboard	-	[Ctrl+%c]

Figure 2: Strings for keystrokes

- **Strings related to browsers:**
  - The malware contains strings referencing profiles.ini files of browsers like Firefox and Opera.
  - Login Data of Chrome, Chromium and Opera are being accessed by the malware.
  - There is suspicious SQL query string in the program: ***select * from moz_logins***
  - The presence of these strings hints that the malware could be a **password stealer**.

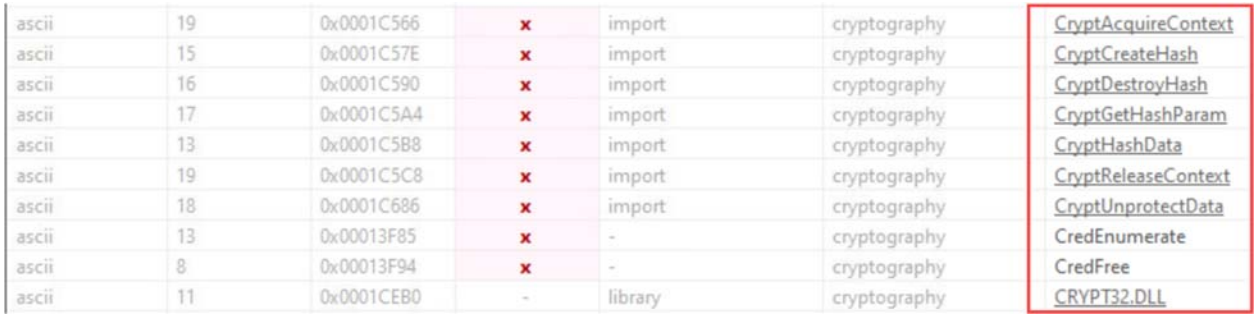


- **Network related imports and strings:**
  - The malware imports standard networking functions from WS2_32.dll like *WSAStartup*, *socket*, *connect*, *send* and *recv* which indicate that the malware might be contacting a CNC server.
  - A string containing a GET request can also be seen in the program. The User-Agent in this GET request is Mozilla suggesting that the program could be trying to fake the request as being from Firefox.

ascii	7	0x0001CDCC	x	utility	network	connect
ascii	6	0x0001CE26	x	utility	network	select
ascii	4	0x0001CE30	x	utility	network	send
ascii	10	0x0001CD74	x	import	network	WSACleanup
ascii	15	0x0001CD82	x	import	network	WSAGetLastError
ascii	8	0x0001CD94	x	import	network	WSAIoctl
ascii	10	0x0001CDA0	x	import	network	WSAStartup
ascii	12	0x0001CDAE	x	import	network	__WSAFDIsSet
ascii	11	0x0001CDBE	x	import	network	closesocket
ascii	13	0x0001CDD6	x	import	network	gethostbyname
ascii	11	0x0001CDE6	x	import	network	gethostname
ascii	9	0x0001CDFC	x	import	network	inet_ntoa
ascii	11	0x0001CE08	x	import	network	ioctlsocket
ascii	10	0x0001CE38	x	import	network	setsockopt
ascii	8	0x0001CE46	x	import	network	shutdown
ascii	19	0x0001490B	x	-	network	GetExtendedTcpTable
ascii	19	0x0001491F	x	-	network	GetExtendedUdpTable
ascii	5	0x0001CDF4	x	-	network	htons
ascii	5	0x0001CE16	x	-	network	ntohs
ascii	4	0x0001CE1E	x	-	network	recv
ascii	6	0x0001CE52	x	-	network	socket
ascii	10	0x0001D0F0	-	library	network	WS2_32.dll
ascii	12	0x000148FE	-	file	network	iphlpapi.dll

```
GET %s HTTP/1.1\r\nHost: %s \r\nUser-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11..
```

- **Cryptography related imports:**
  - The malware imports various cryptographic functions from CRYPT32.dll like *CryptCreateHash*, *CryptHashData* and *CryptUnprotectData* suggest that the malware might be hashing the data it steals.



ascii	19	0x0001C566	x	import	cryptography	CryptAcquireContext
ascii	15	0x0001C57E	x	import	cryptography	CryptCreateHash
ascii	16	0x0001C590	x	import	cryptography	CryptDestroyHash
ascii	17	0x0001C5A4	x	import	cryptography	CryptGetHashParam
ascii	13	0x0001C5B8	x	import	cryptography	CryptHashData
ascii	19	0x0001C5C8	x	import	cryptography	CryptReleaseContext
ascii	18	0x0001C686	x	import	cryptography	CryptUnprotectData
ascii	13	0x00013F85	x	-	cryptography	CredEnumerate
ascii	8	0x00013F94	x	-	cryptography	CredFree
ascii	11	0x0001CEB0	-	library	cryptography	CRYPT32.DLL

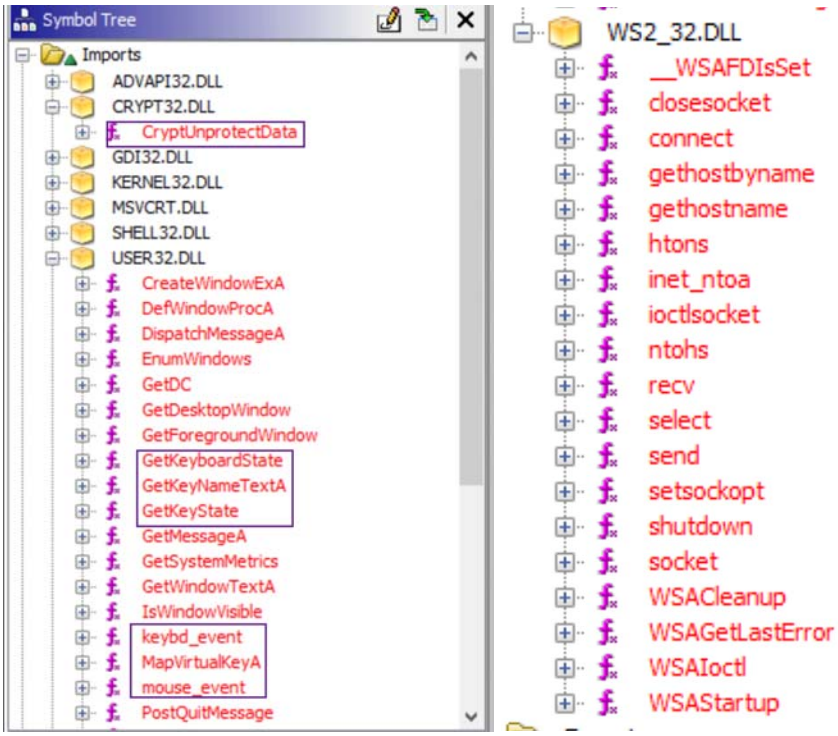
- **Persistence:**
  - There is a string **"SOFTWARE\Microsoft\Windows\CurrentVersion\Run\"** in the malware indicating that it might be trying to establish persistence by modifying this registry entry. But when the program was actually run and the system turned on after a sign out and a restart, the sample1.exe process was not running.

- This could mean either of the following:
  - The malware is not persistent.
  - It detected that it is running in a VM and is therefore not displaying all of its capabilities.
  - It is not getting all the resources it needs to run properly, for example, a proper internet connection.

- **Conclusion from PEStudio analysis:** By analyzing the imports and strings, it can be concluded that there is a strong possibility that the malware is a keylogger that steals passwords and communicates with a CNC server where it might be sending the passwords. It probably encrypts the passwords before sending them to the CNC server.

Ghidra Analysis

- The imports window of Ghidra shows that some suspicious functions are used by the malware from USER32.DLL. The malware also seems to be using cryptographic functions to encode data. From the WS2_32.dll imports, we can conclude that the program tries to exchange data with some remote host.



- The function at 0040350f has three arguments: the socket used to send and receive data, a char pointer pointing to the host URL and a short int with the port number. This function seems to be getting the IP address for the URL, initializing some buffers to 0 and sending and receiving some data through the socket.

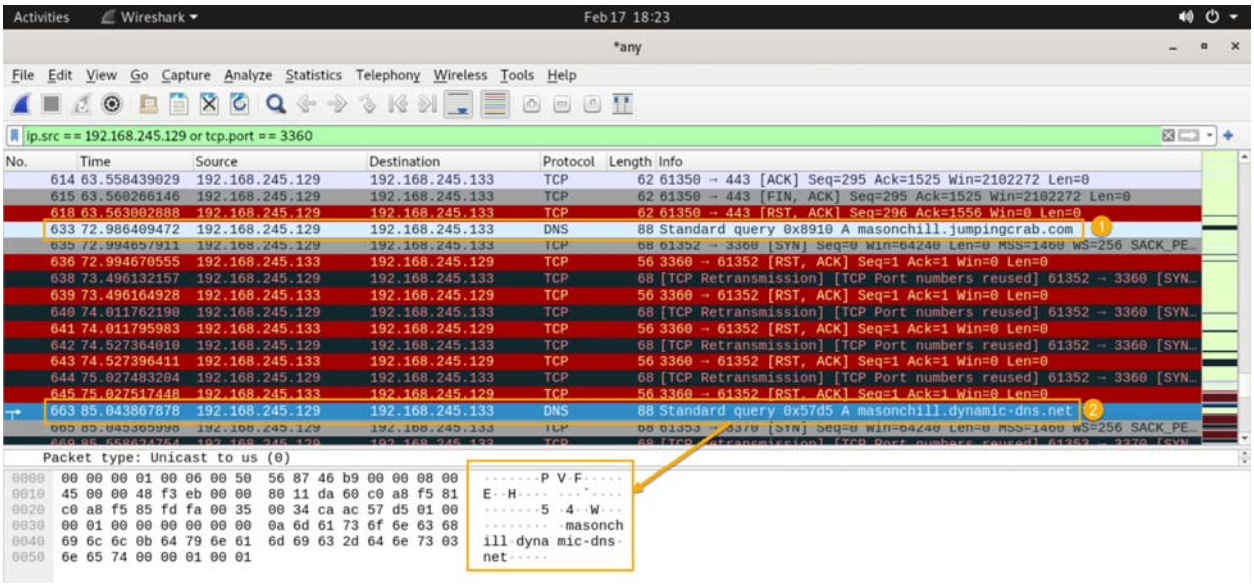
```
bool __cdecl FUN_0040350f(SOCKET socket,char *hostURL,u_short portNumber)

{
  undefined4 hostAddr;
  int iVar1;
  char local_13d;
  char local_13c;
  char local_139;
  undefined local_138;
  undefined local_137 [2];
  undefined local_135 [4];
  undefined local_131;
  undefined2 local_130;
  undefined local_12e [2];
  undefined local_12c [12];
  fd_set local_120;

  hostAddr = getAddr(hostURL,portNumber,&local_130);
  if ((char)hostAddr != '\0') {
    initializeBuffer((int)&local_139,0,9);
    local_139 = '\x04';
    local_138 = 1;
    FUN_0040b30f((int)local_137,(int)local_12e,2);
    FUN_0040b30f((int)local_135,(int)local_12c,4);
    local_131 = 0;
    iVar1 = send(socket,&local_139,9,0);
    if (iVar1 == 9) {
      initializeBuffer((int)&local_13d,0,4);
      local_120.fd_array[0] = socket;
      local_120.fd_count = 1;
      iVar1 = select(socket + 1,&local_120,(fd_set *)0x0,(fd_set *)0x0,(timeval *)0x0);
      if (((0 < iVar1) && (iVar1 = __WSAFDIsSet(socket,&local_120), iVar1 != 0)) &&
         (iVar1 = recv(socket,&local_13d,4,0), iVar1 == 4)) {
        return local_13d == '\0' && local_13c == 'Z';
      }
    }
  }
      }

  .
```

Dynamic Analysis

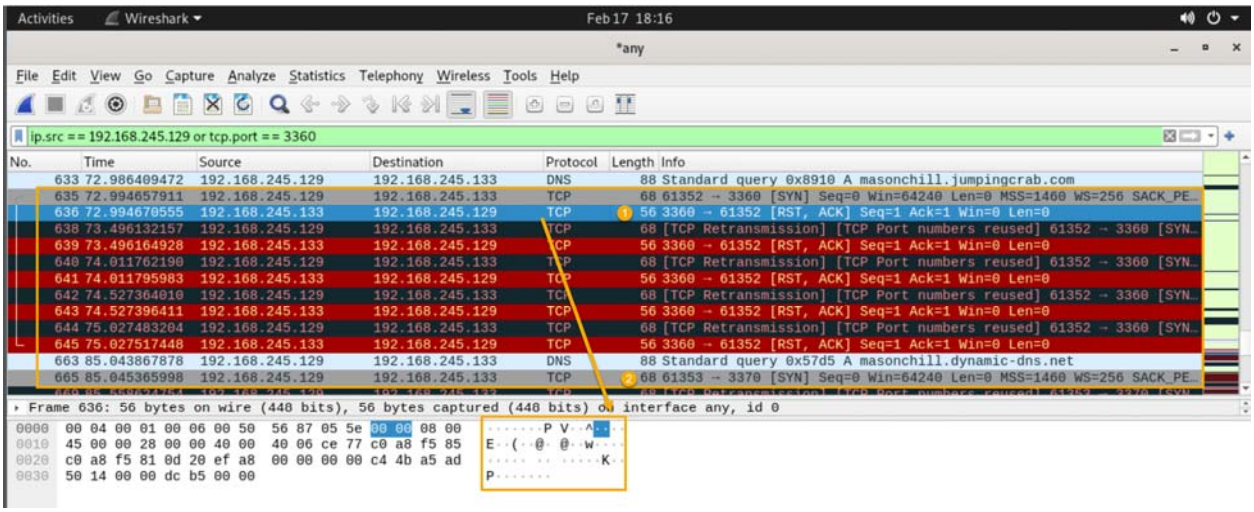
Wireshark Analysis

By sniffing packets using Wireshark, I could get the following information.

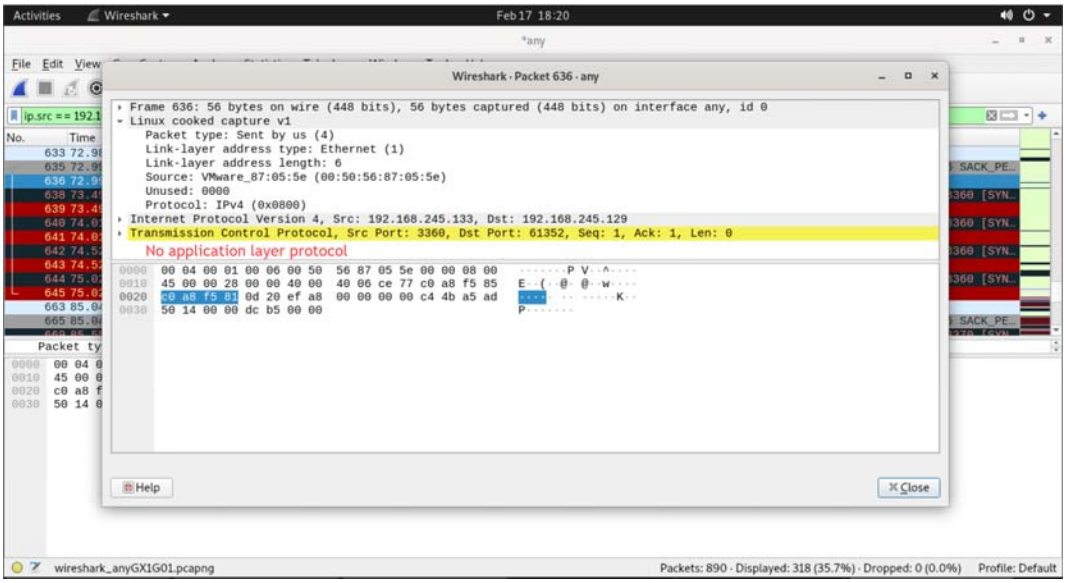
- The malware tries to resolve the IP for the two URLs given below by constantly sending DNS requests. The CNC servers for the malware could be located at these addresses.
  1. **masonchill.jumpingcrab.com**
  2. **masonchill.dynamic-dns.net**



- The malware tries to send data on non-standard ports, 3360 and 3370, using TCP. The data being sent didn't reveal any information. It could be encrypted.

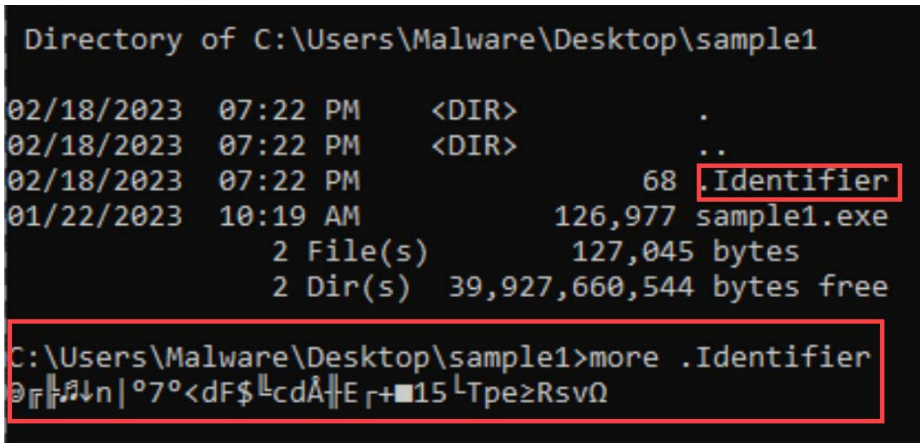


- Opening the above packet to see its details shows that it doesn't use any application layer protocol. It only uses TCP.



ProcMon Analysis

- ProcMon shows that a file called "**.Identifier**" gets created in Desktop\sample1 and some information is written to it and is getting queried from it. Opening this file shows some unreadable data indicating that the data might be encrypted.



- The malware creates a new thread and performs multiple writes to a log file located at *C:\Users\malware\AppData\Roaming\Logs*. Once it is done writing to the file, it terminates the thread. Checking the log file being written to did not give a lot of information as the data was encoded. Trying to decode the log file in HxD decoder also did not help much.

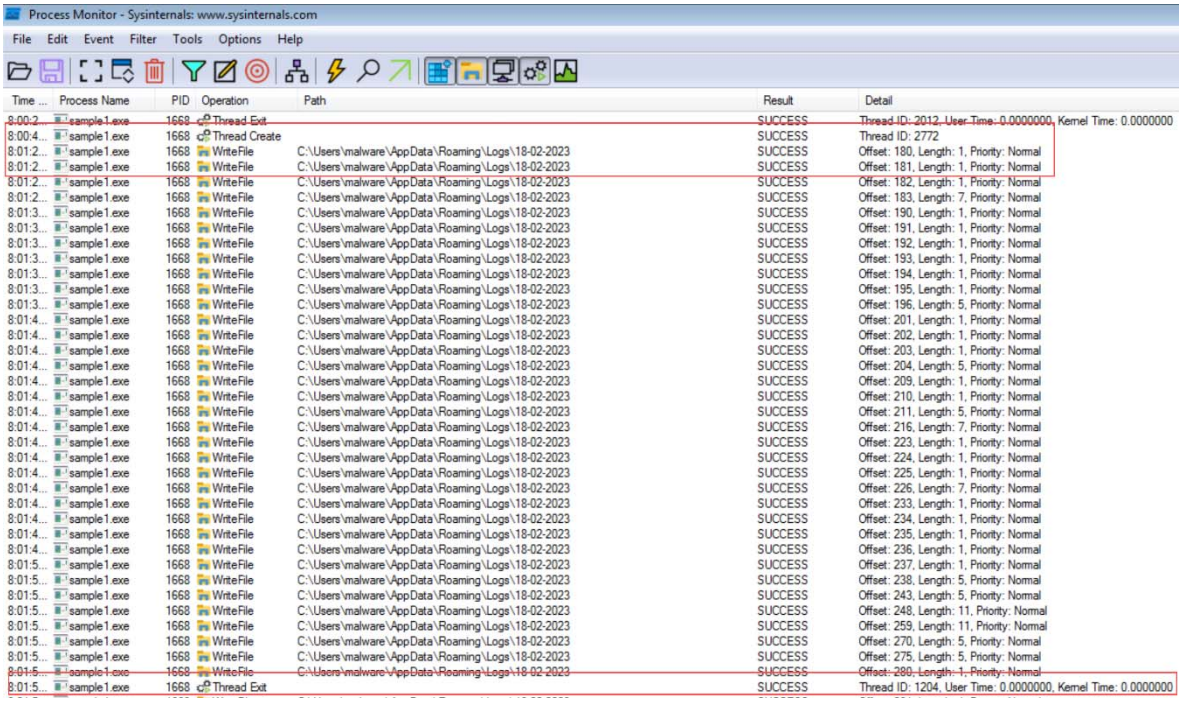


Figure 1: Malware writes to a log file

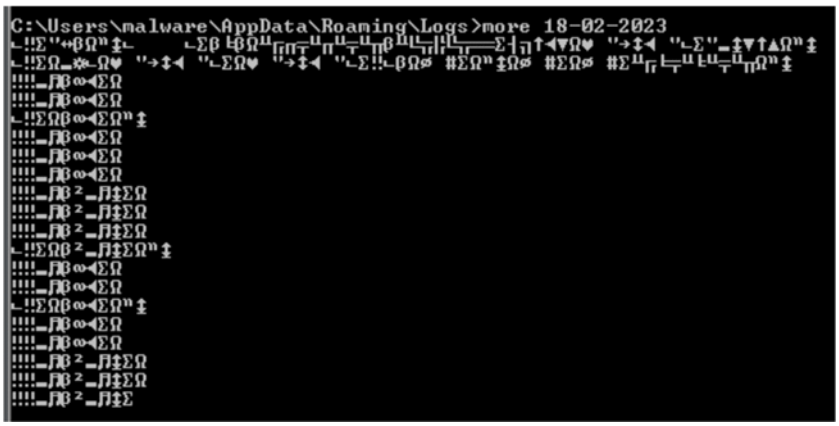


Figure 2: The log file contains some encoded text

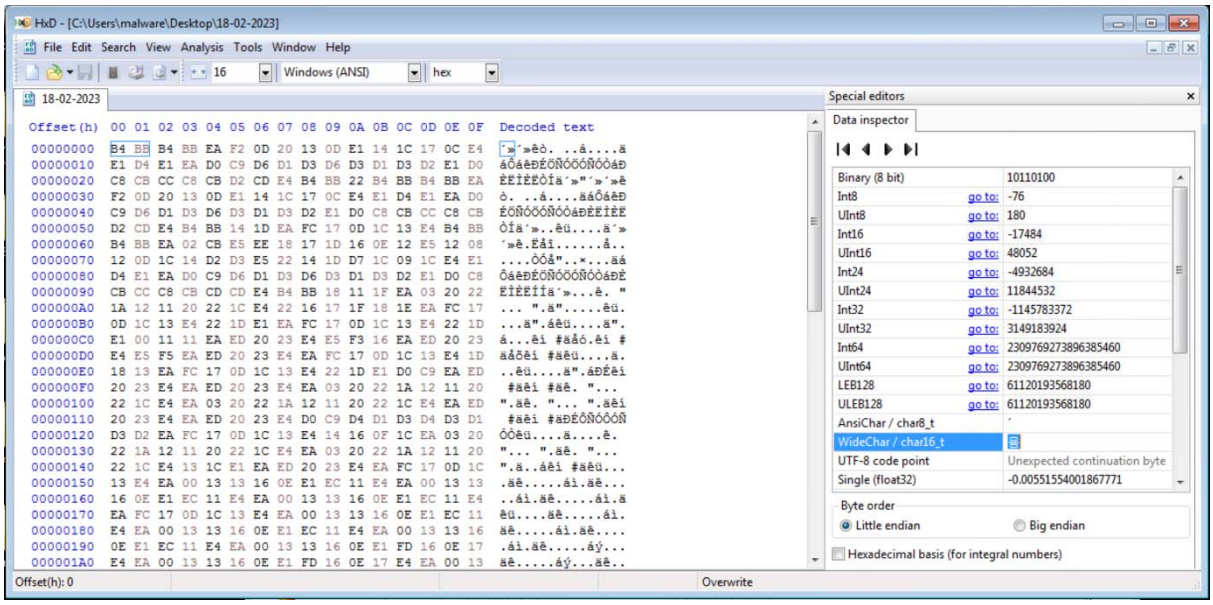
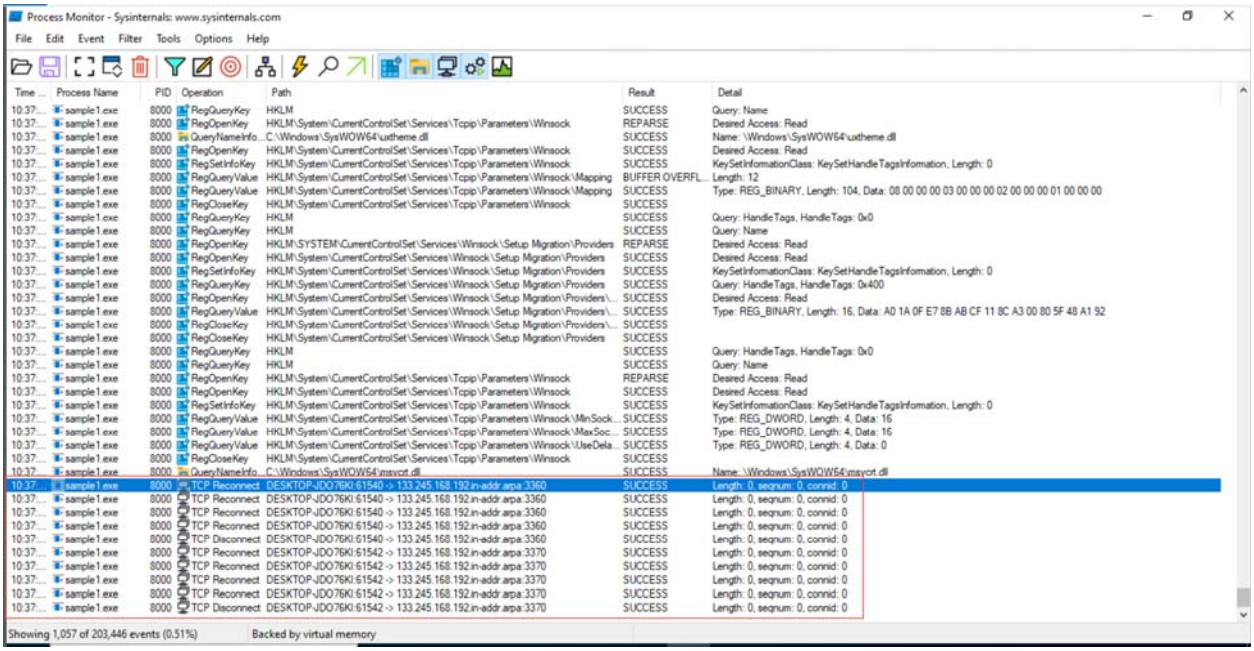


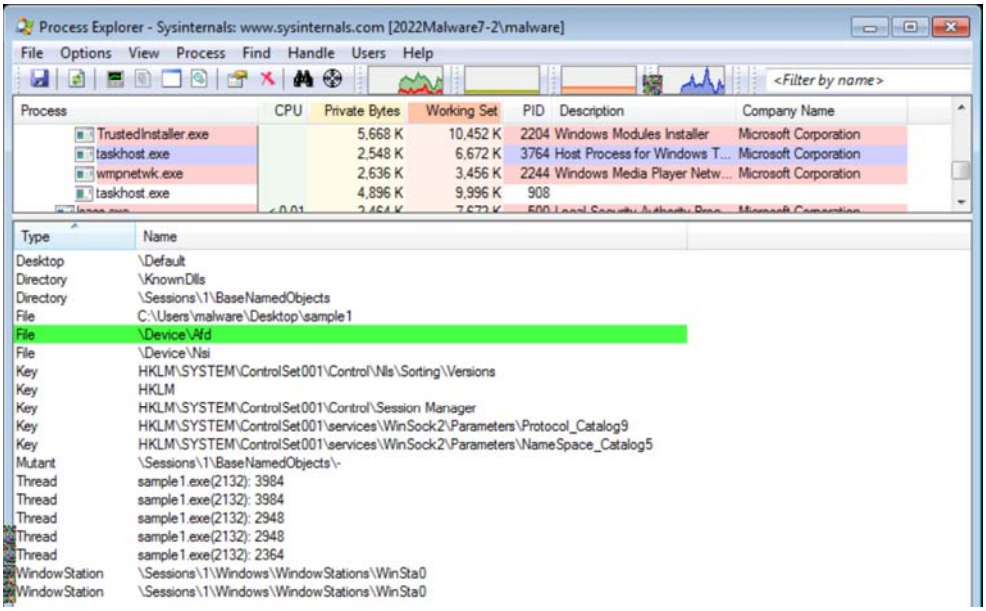
Figure 3: Decoding the file with HxD doesn't help

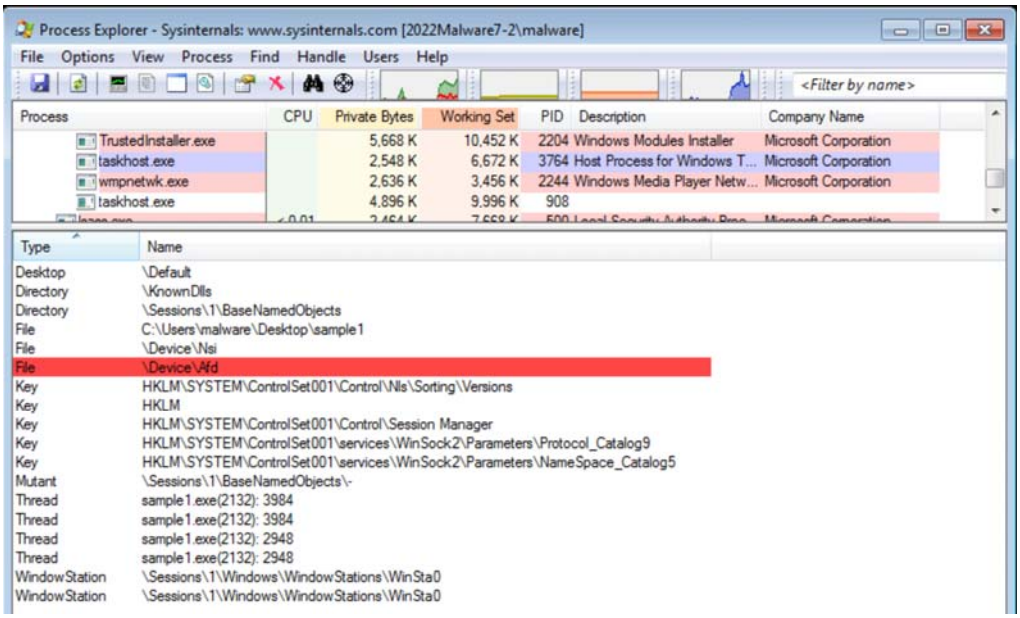
- We can see that the malware tries to communicate via a TCP connection on ports 3360 and 3370. This provides more evidence about the data gleaned from Wireshark.



Process Explorer Analysis

- The malware constantly keeps adding and deleting the handle to \Device\Afd which could indicate that it is using this Afd device to send out and receive network traffic.





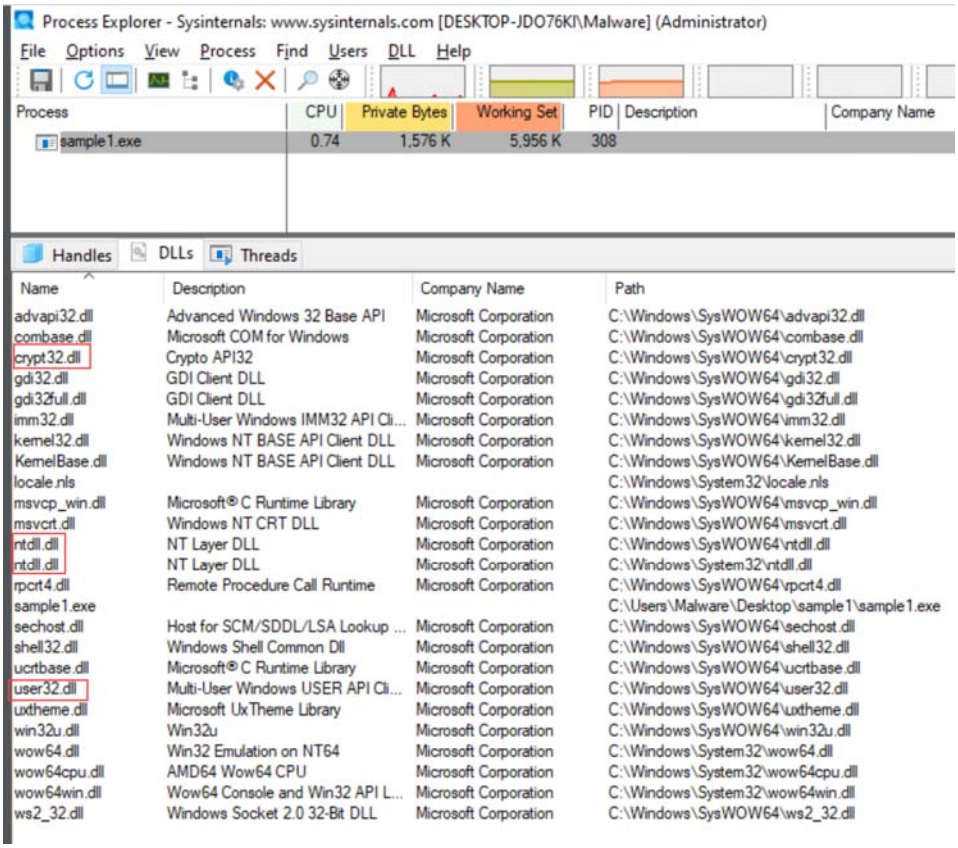
- Process Explorer shows that the malware accesses these registry keys but legitimate programs also access them so we can't concretely conclude the presence of any malicious through this.

Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions
Key	HKLM
Key	HKLM\SYSTEM\ControlSet001\Control\Session Manager
Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Parameters\Protocol_Catalog9
Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Parameters\NameSpace_Catalog5

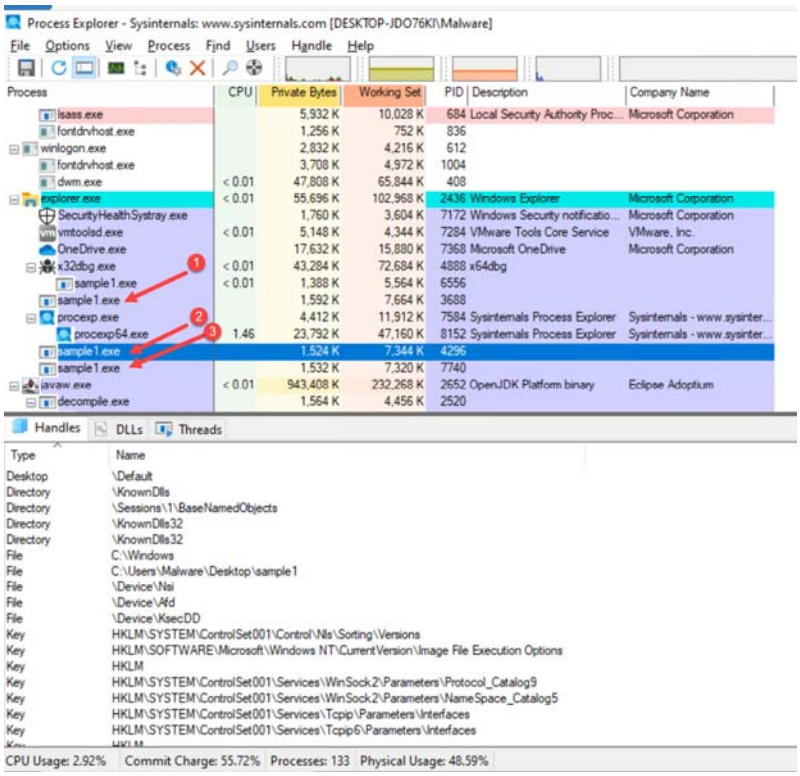
- Process Explorer shows that the malware creates and deletes multiple threads while it is executing. But it does not spawn any new processes or services.

Thread	sample1.exe(2132): 3984
Thread	sample1.exe(2132): 3984
Thread	sample1.exe(2132): 2948
Thread	sample1.exe(2132): 2948

- The DLLs being used by the malware can be seen in Process Explorer. DLLs like crypt32.dll (used for encryption), ntdll.dll (used for calling kernel functions directly via the Native API) and user32.dll (used for monitoring keyboard and mouse status to take user input) are suspicious as they are often used by malware. Though only the presence of these DLLs isn't sufficient to conclude that a program is malicious.



- Running the malware multiple times creates multiple sample1.exe processes. This shows that the malware can't detect multiple copies of itself running on the same machine.



FakeDNS

- Just like Wireshark, FakeDNS also shows that the malware is trying to resolve the IPs for the following two URLs:
    - masonchill.jumpingcrab.com
    - masonchill.dynamic-dns.net

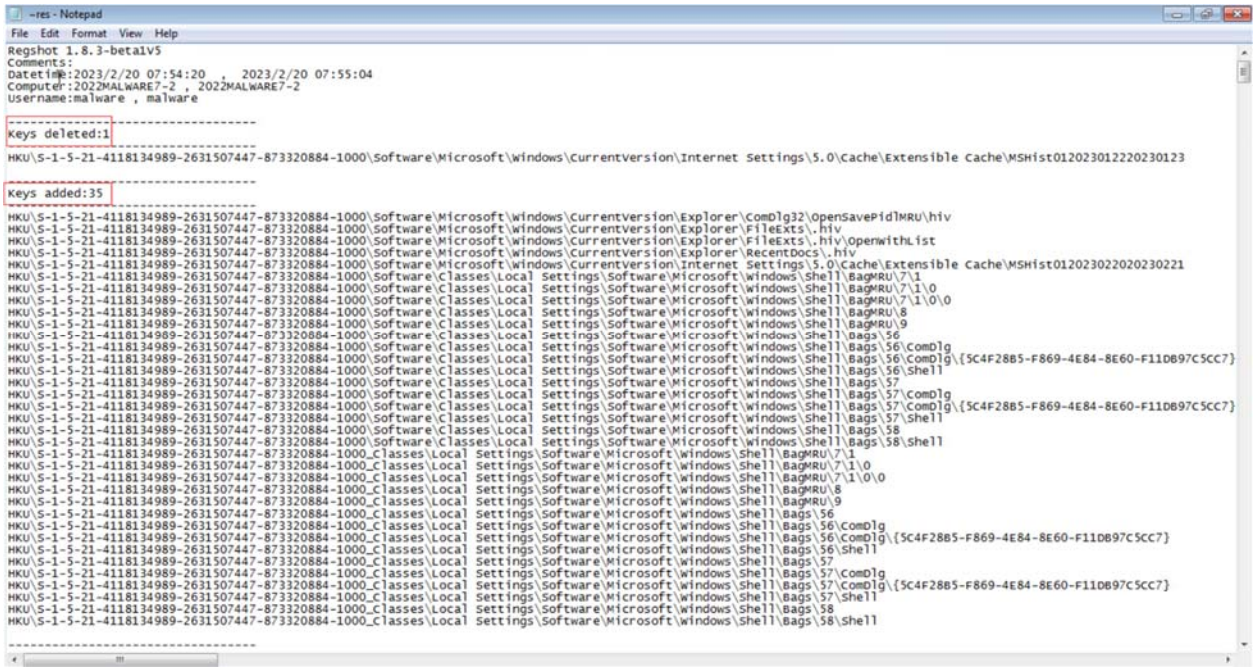
```
remnux@remnux:~/Desktop$ fakedns
fakedns[INFO]: dom.query. 60 IN A 192.168.245.133
fakedns[INFO]: Response: masonchill.jumpingcrab.com -> 192.168.245.133
fakedns[INFO]: Response: masonchill.dynamic-dns.net -> 192.168.245.133
fakedns[INFO]: Response: 133.245.168.192.in-addr.arpa -> 192.168.245.133
fakedns[INFO]: Response: masonchill.jumpingcrab.com -> 192.168.245.133
fakedns[INFO]: Response: fs.microsoft.com -> 192.168.245.133
fakedns[INFO]: Response: ctldl.windowsupdate.com -> 192.168.245.133
fakedns[INFO]: Response: checkappexec.microsoft.com -> 192.168.245.133
fakedns[INFO]: Response: masonchill.dynamic-dns.net -> 192.168.245.133
fakedns[INFO]: Response: masonchill.jumpingcrab.com -> 192.168.245.133
```

InetSim

- InetSim showed no activity on ports 3360 and 3370 as the malware doesn't communicate using any application layer protocol.

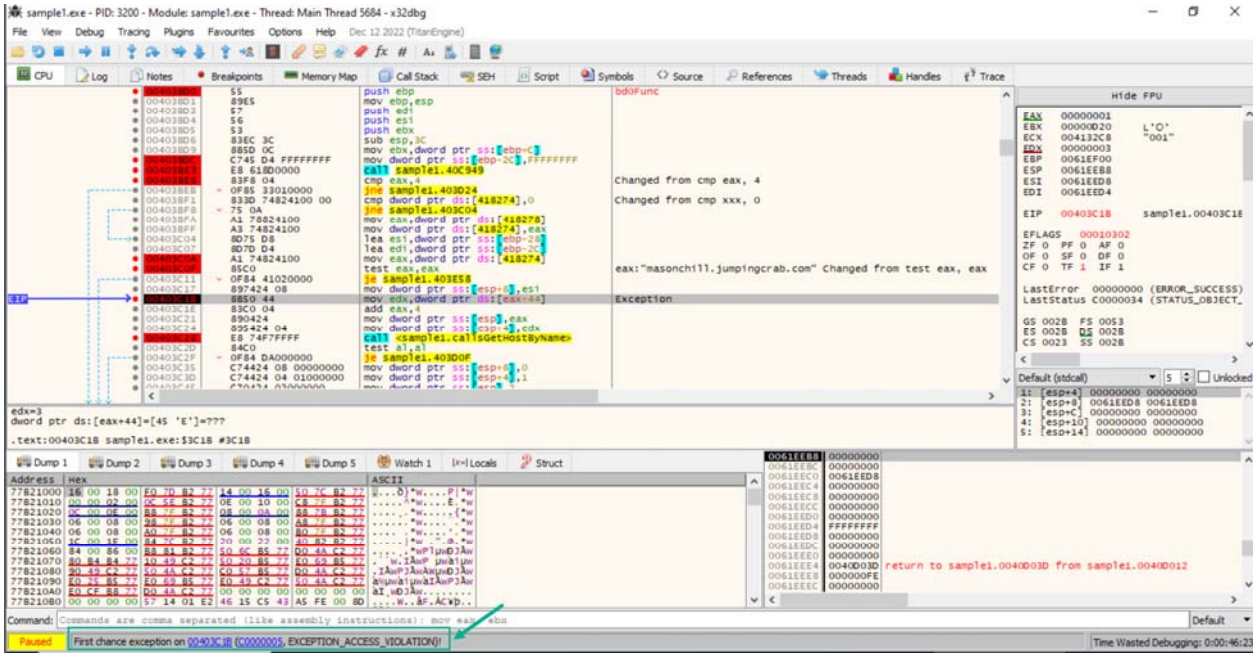
RegShot

- RegShot analysis shows that the malware deletes a registry key and adds a bunch of keys but none of them help much with analyzing the malware's behavior.



X32dbg

I tried to debug the sample using x32dbg but kept getting a “Access Violation” exception at address 00403C1B when I changed some of the register values on the path to the function at address 0040350F.



Indicators of Compromise

The program creates a **.Identifier** file in the same folder where sample1.exe is stored. This can be used as a host-based indicator of compromise.

The malware tries to resolve DNS requests to get the IP for **masonchill.jumpingcrab.com** and **masonchill.dynamic-dns.net**. These can be used as network-based indicators of compromise.