New Dridex Variant Being Spread By Crafted Excel Document | FortiGurad Labs

By Xiaopeng Zhang

Published: 2021-09-10 · Archived: 2026-04-05 18:02:57 UTC

FortiGuard Labs Threat Research Report

Affected platforms: Microsoft Windows
Impacted parties: Windows Users
Impact: Collects sensitive information from victims' computers and delivers and executes malicious modules on victims' device.
Severity level: Critical

Dridex is a Trojan malware, also known as Bugat or Cridex, which is capable of stealing sensitive information from infected machines and delivering and executing malicious modules (dll).

FortiGuard Labs recently captured new phishing email campaigns in the wild that included a specially crafted Excel document attachment. I did a deep research on one of them and discovered that once the malicious Excel document is opened on a victim's machine, it downloads a new variant of Dridex.

In this analysis, I will elaborate how the Excel document downloads Dridex, how this version of Dridex runs on a victim's device, what sensitive information it collects, and how it delivers malicious modules (dll).

The Phishing Email of the Dridex Variant

Figure 1.1 shows one of the recent phishing emails with a malicious Excel attachment infected with Dridex.

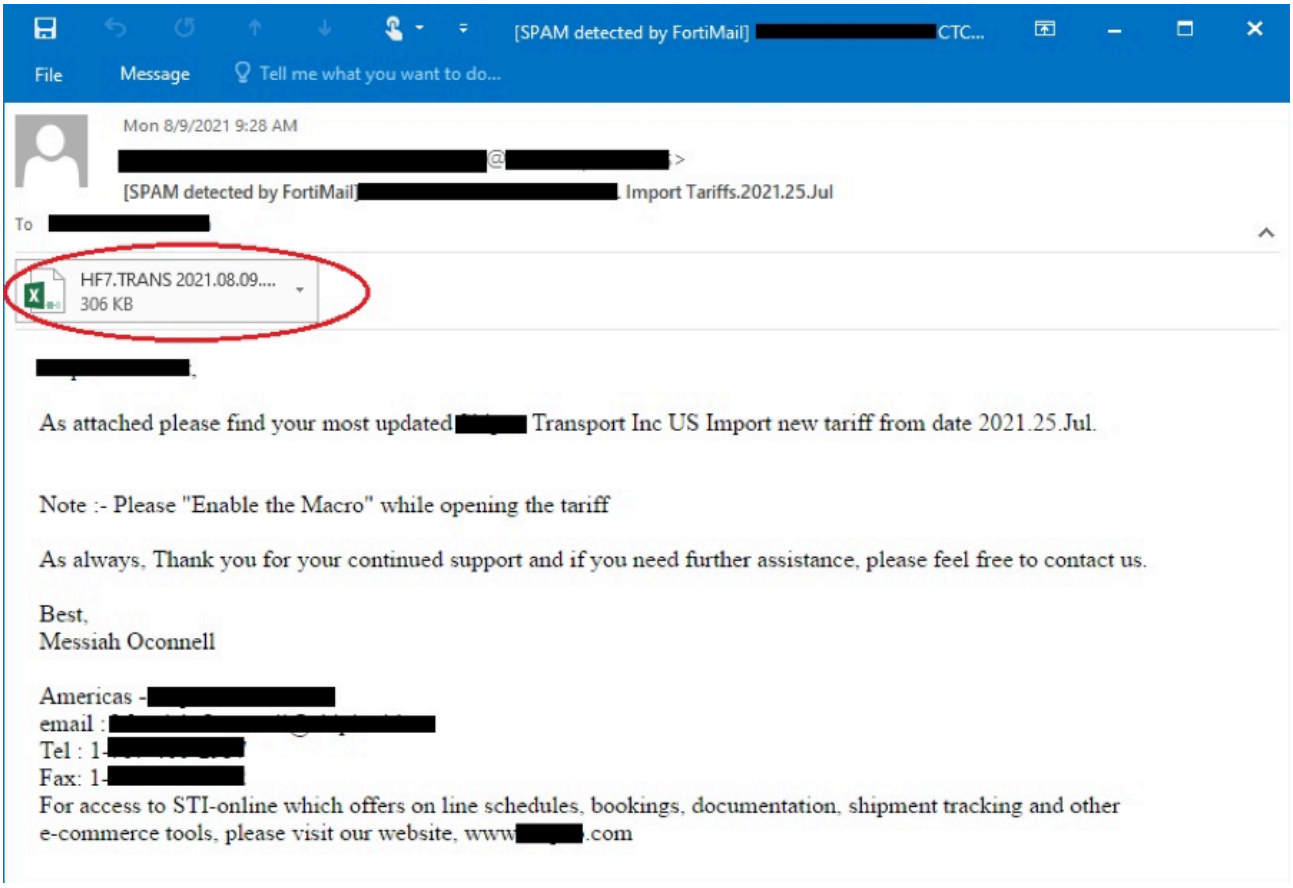

Figure 1.1 – Text of a recently captured phishing email

As you can see, this email disguises itself as sending Import Tariffs data to a customer, and then asking him/her to view the details by opening the attached Excel file (in this case, “HF7.TRANS 2021.08.09.xlsb”).

Analysis of the Macro Inside the Excel Document

When the recipient opens the attached Excel document, it contains a message at the top of the document in bold red letters asking them to “Please enable macros.” However, Excel displays a yellow “Security Warning” bar telling the end user that macros are currently diabled, with the implication that clicking the button “Enable Content” may be risky, as shown in Figure 2.1.

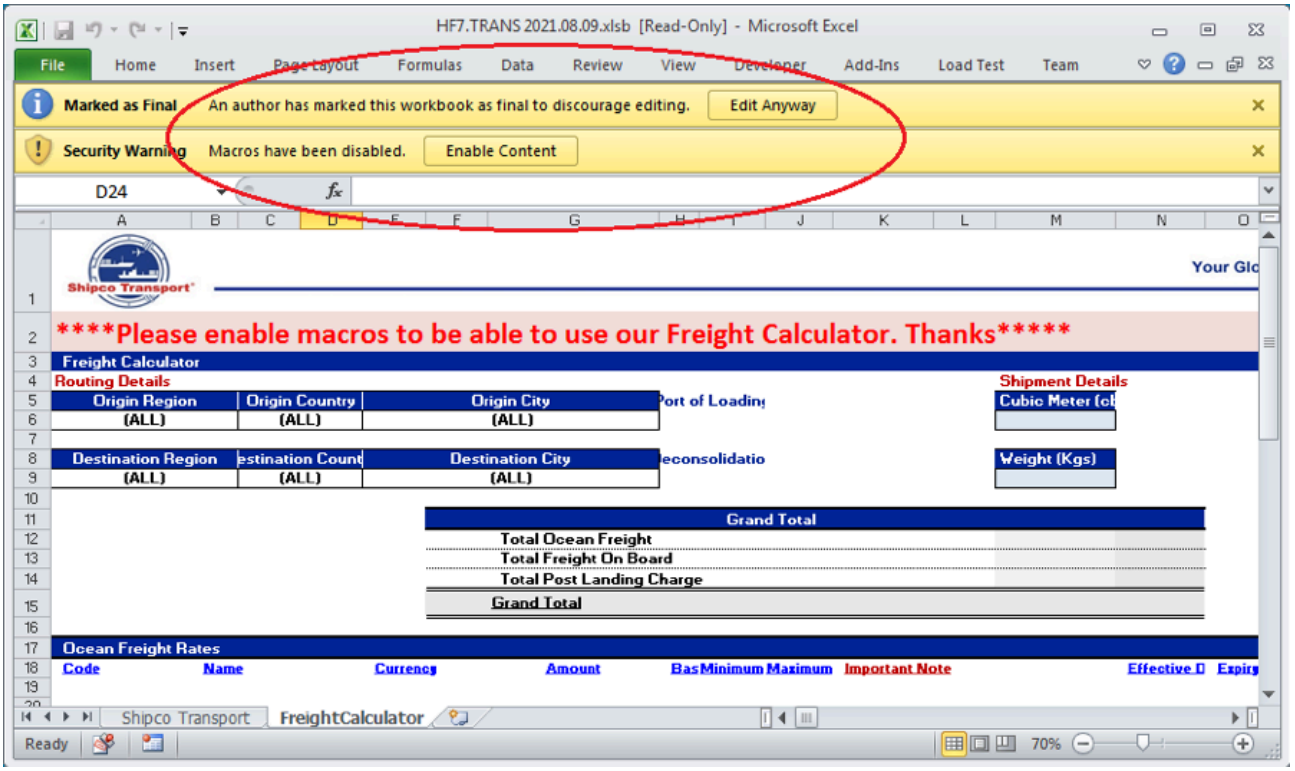

Figure 2.1 – Excel displays a warning bar when opening the infected Excel document

Looking into the internal details of the Excel file, I found that it not only used auto-run Macro(VBA) but also Excel 4.0 Macro. There is an auto-run function called Workbook_Open() in the Macro(VBA), which is automatically called when the Excel file is opened.

The code is included below:

```
Sub Workbook_Open()
    ActiveWorkbook.Sheets("Macro1").Range("A1").Value=Environ("allusersprofile")&
    "\KgmsgJbgP.sct"
End Sub
```

It only sets the value Environ("allusersprofile")& "\KgmsgJbgP.sct" (“C:\ProgramData\KgmsgJbgP.sct” in my testing environment) to the “A1” cell of a sheet called “Macro1”.

“Macro1” is a hidden sheet that contains and executes the Excel 4.0 Macro, which is defined in the file “xl\workbook.xml” as shown in Figure 2.2.

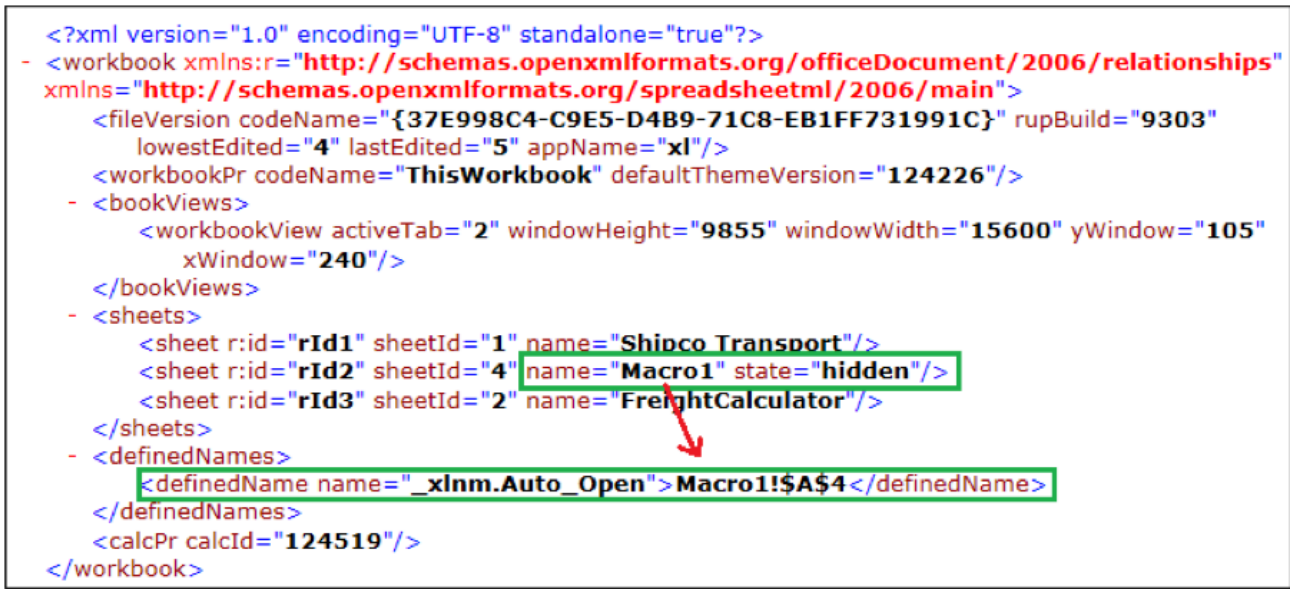

Figure 2.2 – Excel 4.0 Macro sheet is defined in “Workbook.xml”

Excel 4.0 macros use formulas in various cells to execute code. As long as you give it a starting cell, it executes code from top to down, then from left to right.

After the auto-run Macro(VBA) is executed, the Excel 4.0 Macro is executed automatically starting from the cell “Macro1!A4”.

The Excel 4.0 Macro extracts data from a bunch of cells within the “Macro1” sheet into a local file, whose file path is saved in A1, which is “C:\ProgramData\KgmsgJbgP.sct”. The extracted data is an HTML application (.hta file) with a piece of VBScript code. The last step of the Excel 4.0 Macro is to execute this “KgmsgJbgP.sct” file using the “mshta” command, which is =EXEC(CONCATENATE("mshta ", CHAR(34), A1, CHAR(34))).

The final command to be executed is “mshta.exe C:\ProgramData\KgmsgJbgP.sct“. “mshta.exe” is a Windows default program that is used to execute an html application (.hta file) with HTML, Dynamic HTML, and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript.

HTML Application Used to Download the Dridex Payload

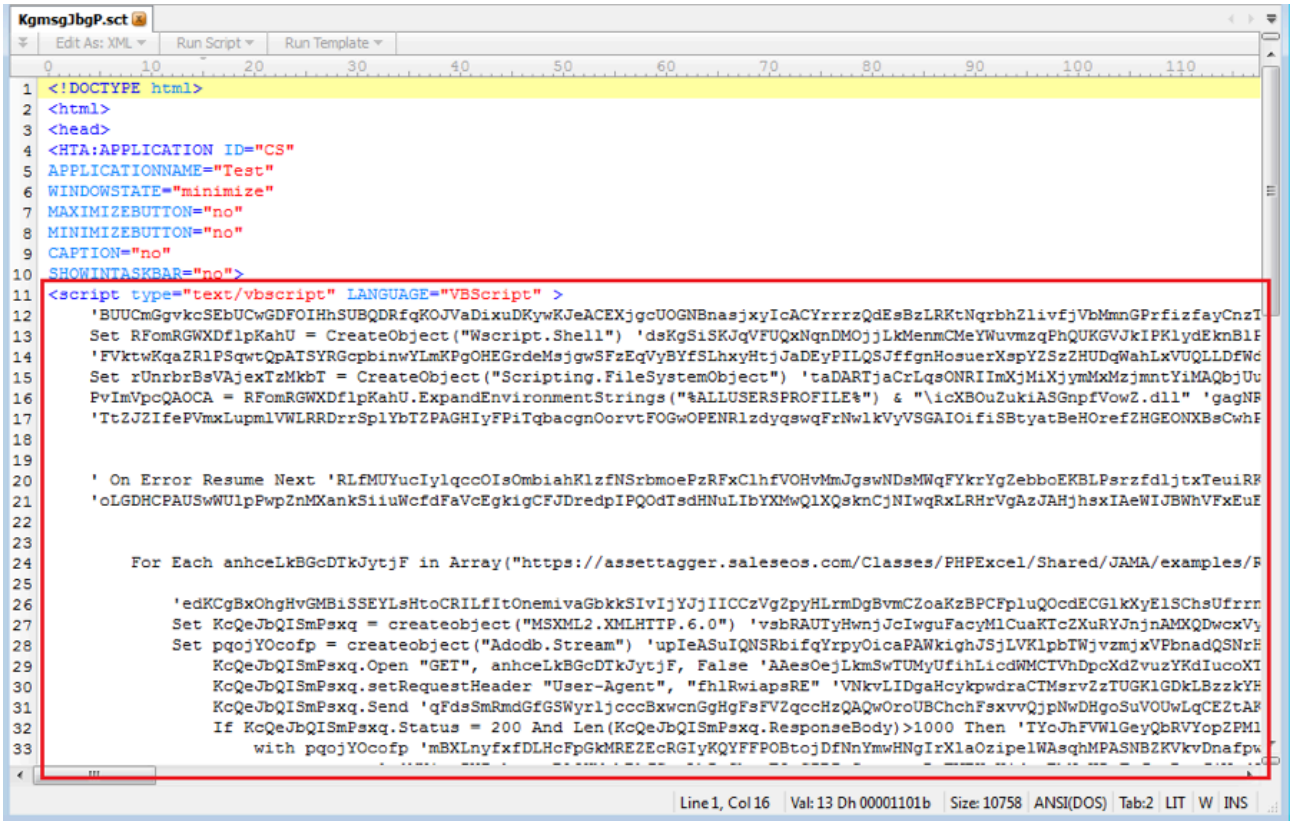

Figure 3.1 – Malicious VBScript code in the extracted hta file

The VBScript code, as shown in figure 3.1, includes an array of ten URLs (refer to the “IOCs” section below for details) that link to the Dridex payload. It downloads Dridex from these ten URLs in a for-loop into a local file, “%ALLUSERSPROFILE%\\icXBOuZukiASGnpfVowZ.dll”, that is hardcoded in this VBScript code. When Dridex is downloaded successfully, it then executes “wmic.exe” (the WMI command-line) to create a new process of “rundll32.exe”.

The simplified code looks like this:

CreateObject("Wscript.Shell").Exec("**wmic process call create \"Rundll32.exe %ALLUSERSPROFILE%\\icXBOuZukiASGnpfVowZ.dll ReportDeviceAdd\""**)

Finally, Rundll32.exe loads the Dridex payload file “icXBOuZukiASGnpfVowZ.dll” and the calls its export function, named “ReportDeviceAdd”, to execute its malicious functions.

Diving Into the Downloaded Dridex Payload File

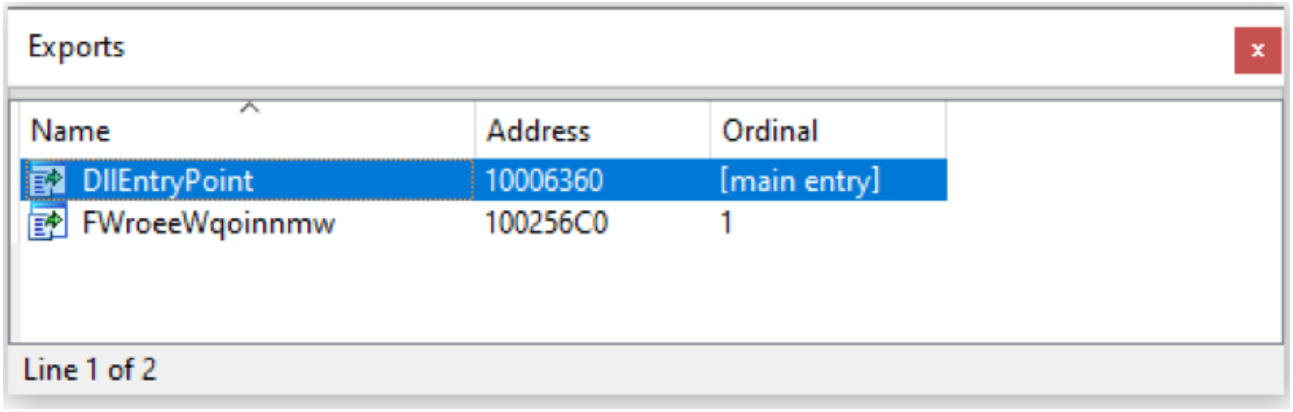

Figure 4.1 – The Dridex variant's export function list in IDA Pro

Figure 4.1 shows the export function list of the payload file of Dridex in IDA Pro. It contains two functions: DllEntryPoint() is the entry function for this dll. And FWroeeWqoinnmw() is the real entry function. An odd thing here is that there is no function for "ReportDeviceAdd", which should be the starting point of this Dridex variant.

To figure this out, we analyzed the internal strategy Rundll32.exe uses to load a module and then call its export function. Figure 4.2 shows the export function list of the Dridex payload file after it is unpacked. It provides six export functions. The fourth function is "ReportDeviceAdd".

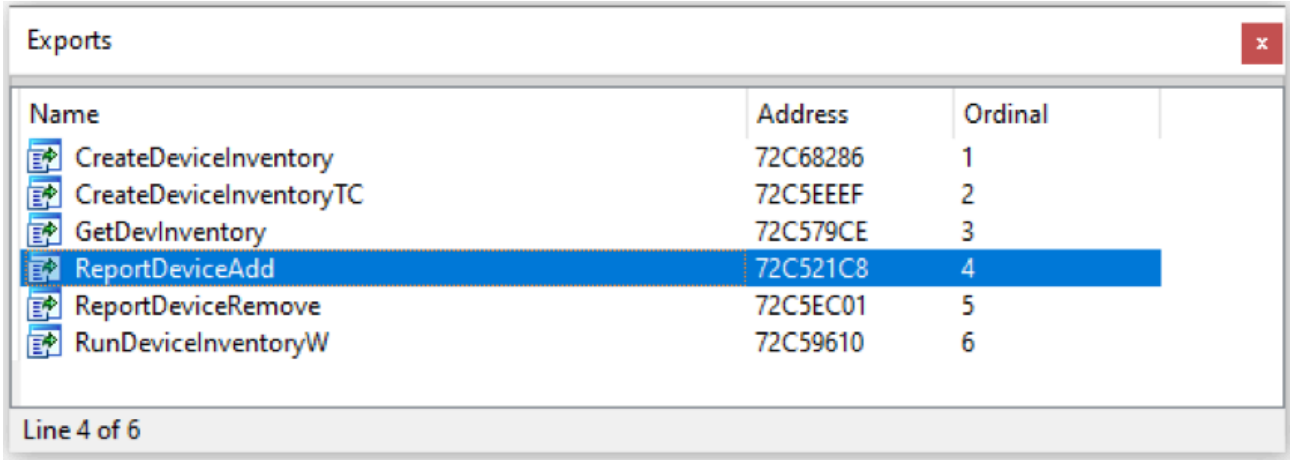

Figure 4.2 – The export function list of unpacked payload file

Here are the steps for how the Dridex payload file is loaded by Rundll32.exe.

The steps used by Rundll32 to load a dll and invoke the export function:

1. Rundll32.exe calls the API LoadLibrary() to load the dll into the memory and then deploy it according to its PE structure.
2. It first invokes the dll's entry point function—DllEntryPoint()—to initialize the module.
3. It then calls API GetProcAddress() with the function name "ReportDeviceAdd" to obtain the function address from the initialized module in step 2.
4. Rundll32.exe then calls the function address obtained in step 3.

This payload file also contains a packer-like program to protect itself from being researched by people. It does the unpacking in step 2 when the payload file's DllEntryPoint() is called.

From this point, Rundll32.exe is able to obtain ReportDeviceAdd by calling the API GetProcAddress().

Anti-analysis Techniques Used in Dridex

Most modern malware includes anti-analysis techniques in their code to prevent it from being analyzed.

This Dridex variant uses anti-analysis techniques similar to another Dridex variant I analyzed last year, which are:

- All APIs are hidden and are found by its name's hash code.
- Entire constant strings are encrypted in memory and decrypted just before using.
- Some APIs are called in a crafted way to raise an exception (0x80000003) on purpose. It then captures the exception in the exception handler function to actually call the API.

Format of the Packet Sent to the C2 Server

Dridex collects sensitive data from the victim's infected device, which is then placed into a formated packet, encrypted, and sent to the C2 server.

Figure 5.1 is a screenshot of the first packet to a C2 server that was about to be encrypted. All the packets to the C2 server have the same packet format. As an example, I'll elaborate on the format of the packet below, which has been separated into many fields by a red pipe in the screenshot.

The selected part is common data for all packets, which is refered as a "packet header" in this analysis.

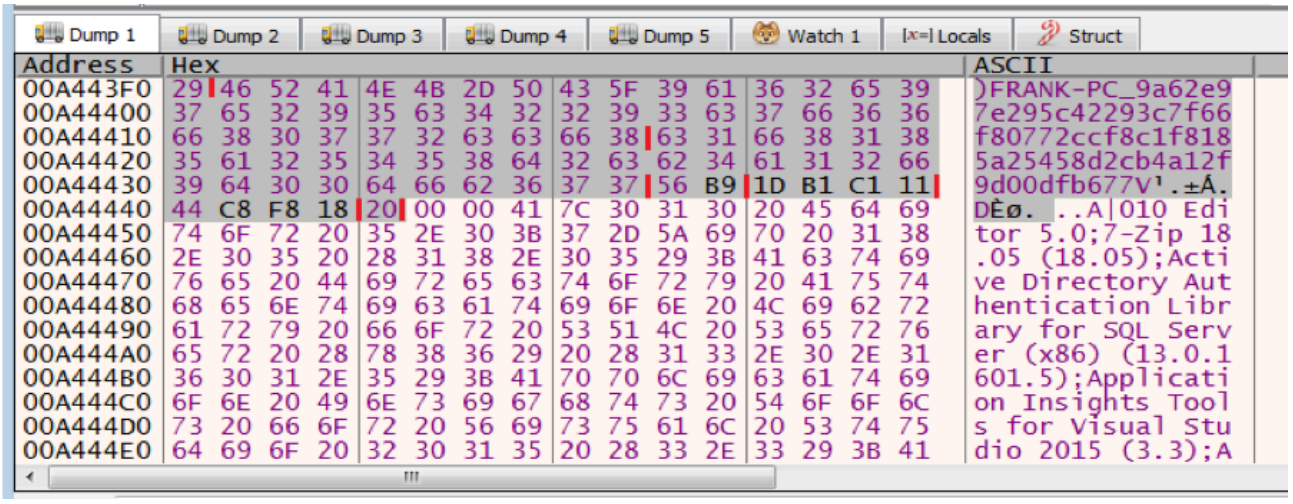

Figure 5.1 – The first packet to the C2 server before encrypted

Table 5.1 explains the content of each field in the packet format.

Offset	Length	Data

0x00	0x01	The length of the victim's ID string.
0x01	0x29	The victim's ID string, which is variable.
0x2A	0x20	The hard drive's volume Information for the infected device.
0x4A	0x02	Dridex version information—which is 0x56B9 in this variant.
0x4C	0x04	A dword mixed with infected Windows version information.
0x50	0x04	Packet Type Identification. 0x18F8C844 is for the first packet.
0x54	0x01	Windows platform. 0x20 for 32-bit, 0x40 for 64-bit.
0x55	variable	Collected data from the victim's system.

Table 5.1 – Decryption of the fields of the packet

- The victim's ID string contains the computer name, underscore, and an MD5 hashcode of a string that includes the computer name, user name, and the Windows system's install date.

- The hard drive's volume information is an MD5 value made from data of the volume information of "C:\" and the Windows installation date.

- 0x56B9 is hardcoded data in the malware, possibly the malware version.

- 0x11C1B11D is a mixed data set of Windows version information, which is obtained from the result of the APIs GetVersionEx() and GetSystemInformation().

- Dridex has five packet type IDs in this variant used to notify the C2 server. They are 0x18F8C844, 0x69BE7CEE, 0x11041F01, 0xD3EF7577, and 0x32DC1DF8.

- The data following 0x20 indicates that the victim's Windows system is 32-bit platform.

The field values of every Dridex packet header (except for packet type ID) are the same for all the packets on the same machine.

The collected data (starting from offset 0x55) is appended to the packet header, which has two fields—the collected data size (four bytes in network byte order) and the collected data followed.

Sending Collected Information to the C2 Server

As with its previous version, the IP address and port of C2 servers are hardcoded in the data. Below is the IP list in binary of the three C2 servers.

```
.data:72C6D02C        dd 2C94B67h     ; IP: 103.75.201.2
.data:72C6D030        dw 1BBh         ; port: 443
.data:72C6D032        dd 6C01DF9Eh    ; IP: 158.223.1.108
.data:72C6D036        dw 1851h        ; port: 6225
.data:72C6D038        dd 0F21C16A5h   ; IP: 165.22.28.242
.data:72C6D03C        dw 1238h        ; port: 4664
```

The malware chooses one IP address and port pair in an for-loop. Once one connection to the C2 server is successfully established, it is used throughout the process life.

“0x18F8C844” is the packet ID for the very first packet. The collected data consists of the entire installed software (including software name and version) and all environment variables defined in the infected system.

It obtains the installed software information one-by-one by enumerating the sub-keys under the key “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall” in the system registry.

It also steals the values of environment variables defined in the victim’s system. On my testing machine, they are *ALLUSERSPROFILE, APPDATA, CommonProgramFiles, COMPUTERNAME, ComSpec, FP_NO_HOST_CHECK, HOMEDRIVE, HOMEPATH, LOCALAPPDATA, LOGONSERVER, NUMBER_OF_PROCESSORS, OS, Path, PATHEXT, PROCESSOR_ARCHITECTURE, PROCESSOR_IDENTIFIER, PROCESSOR_LEVEL, PROCESSOR_REVISION, ProgramData, ProgramFiles, PSModulePath, PUBLIC, QT_AUTO_SCREEN_SCALE_FACTOR, SystemDrive, SystemRoot, TEMP, TMP, USERDOMAIN, USERNAME, USERPROFILE, VS140COMNTOOLS, and windir*.

It calls the API GetEnvironmentStringsW() to obtain all the values of the above environment variables. Figure 6.1 shows a partial set of the name-value pairs obtained in memory.

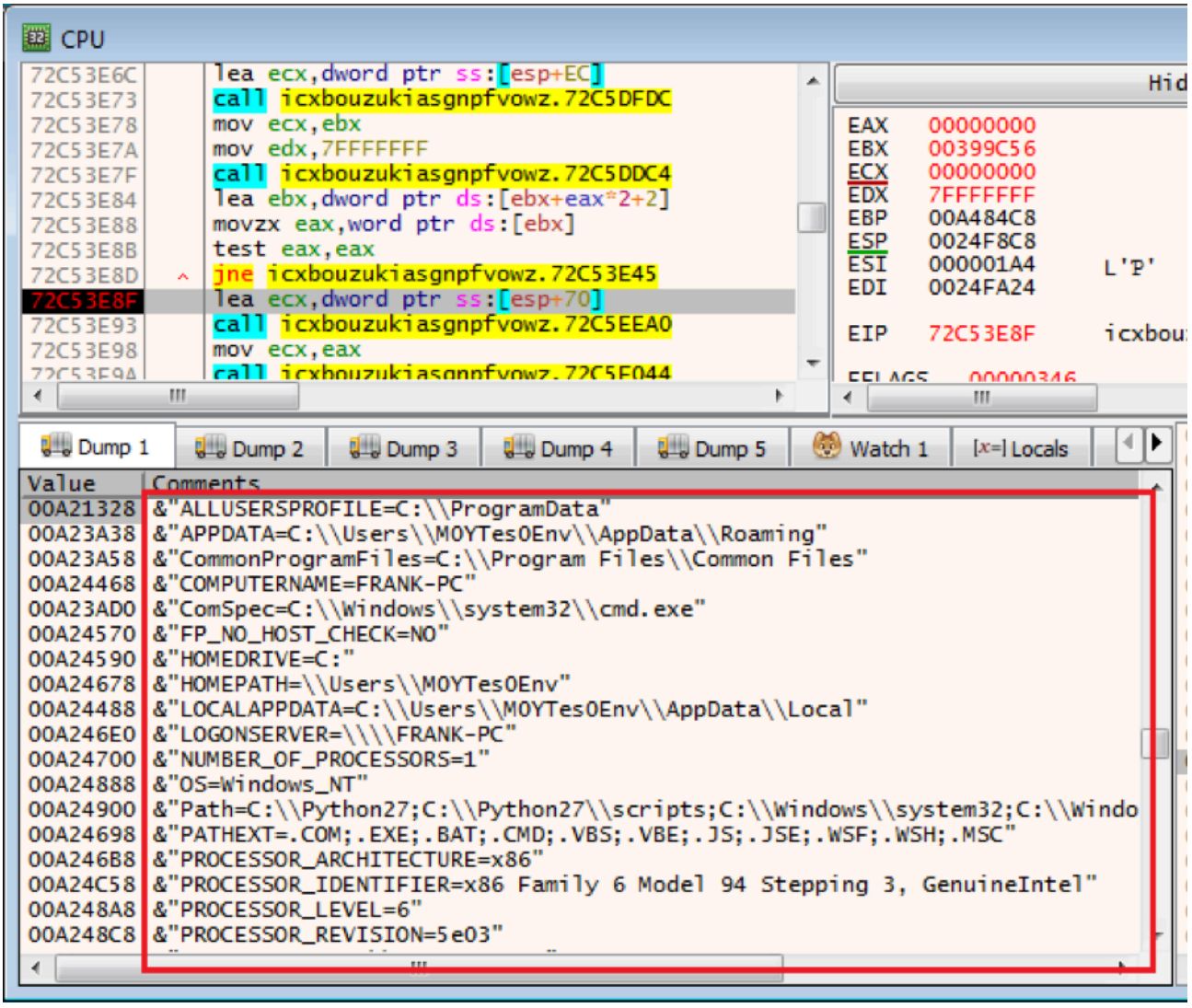

Figure 6.1 – Display of obtained environment variables in the victim's device

Once the first packet is finished, Dridex encrypts the packet and sends it to the C2 server using the HTTP POST method. It invokes a group of APIs to send and receive the data, such as InternetConnectW(), HttpOpenRequestW(), HttpSendRequestW(), HttpQueryInfoW(), and InternetReadFile().

Unfortunately, its C2 servers were down during my analysis, so it was unable to send/receive data to/from the C2 servers. But according to its code workflow, I could create a fake C2 server of Dridex to simulate the server's behaviors to receive and reply to Dridex to continue my research. The following analysis is based on this simulated data.

Deploying a Malicious Module From C2 Server and Performing Persistence

After receiving the response packet to the first packet from the C2 server, it sends the second packet (packet ID 0x11041F01) with no collected data. It should reply with a malicious module (dll file) in the response packet. Dridex verifies the response packet by comparing the packet's hash code, which is the first four bytes of the packet. Next, Dridex sends another packet (packet ID 0xD3EF7577) to inform the C2 server that it has successfully received the module.

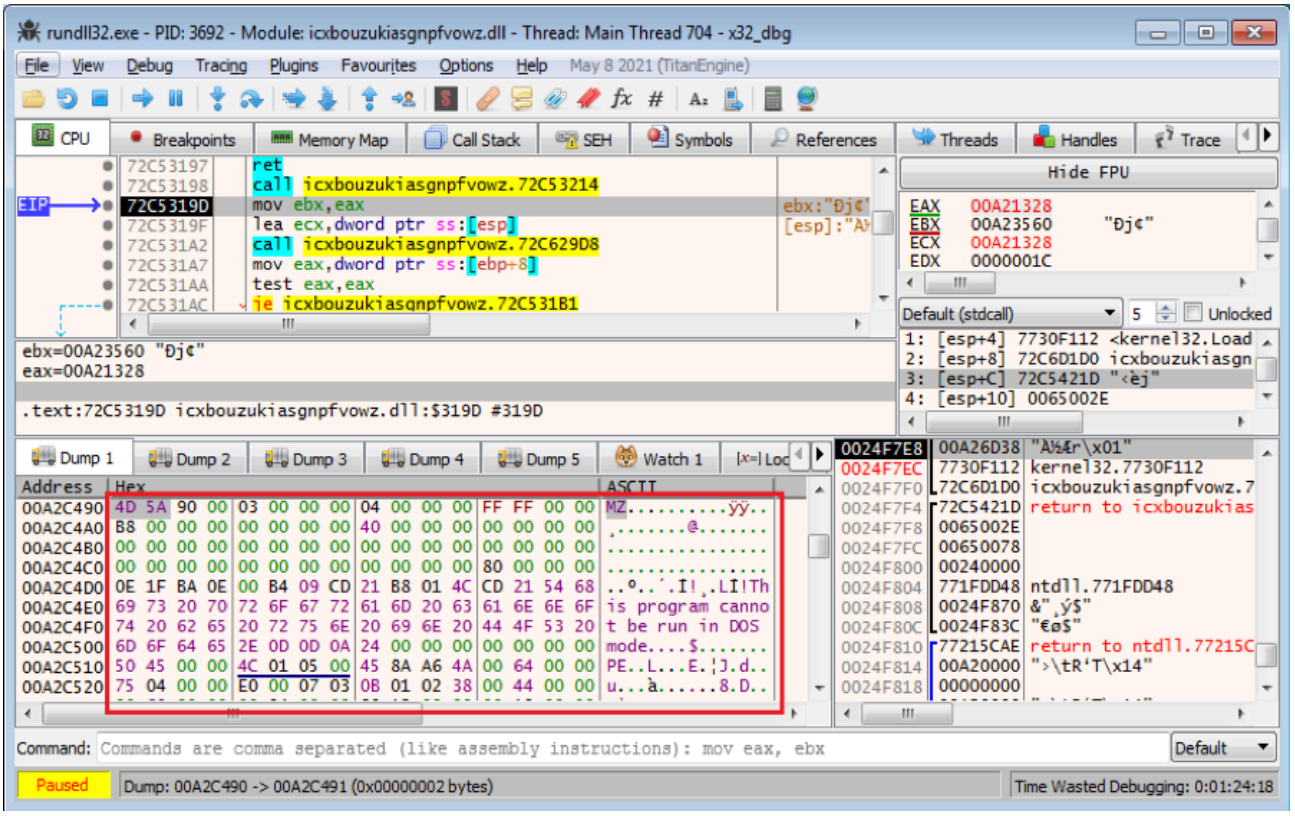

Figure 7.1 - A module extracted from the second response packet.

The second response packet contains an encrypted module (dll) . After Dridex verifies the received packet, it decrypts the module—which is like the memory data displayed at the bottom of Figure 7.1.

Dridex then proceeds to deploy this malicious module onto the victim's machine and then creats a scheduled task to run the module. Let's see how it does this.

To keep the module running secretly on the victim's machine, Dridex uses a Windows default program to load and run it. It randomly chooses a pair of Windows program (exe) and a dll file from "%windir%\system32" that the chosen program has to load. Next, Dridex can override the chosen dll file with the received module. Hence, once the chosen program starts, the malicious module within the chosen dll is executed.

In this way, the victim only supposes that a Windows program is running, not a malware module.

Figure 7.2 is a screenshot of the just-chosen pair of Windows program and the dll file from "%windir%\system32\"

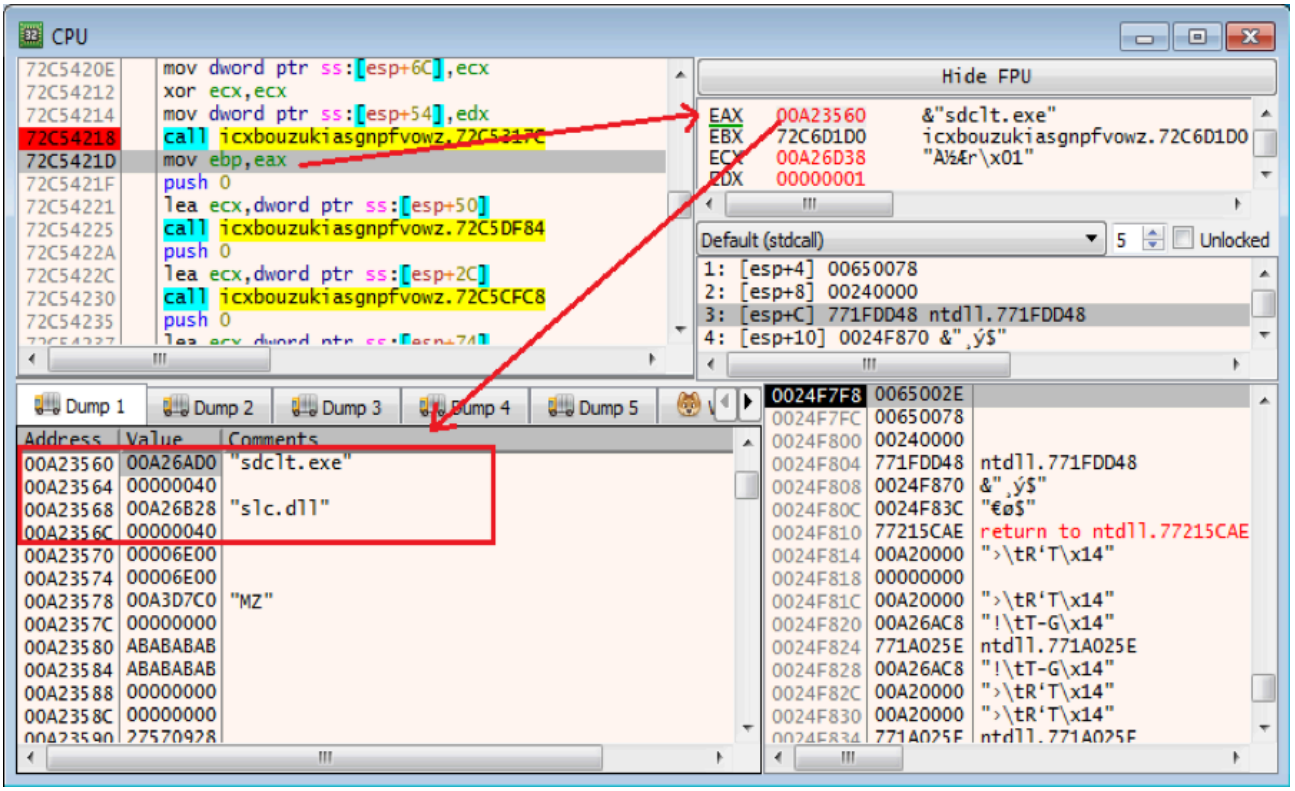

Figure 7.2 – Chosen Windows program and dll

Dridex copies the chosen Windows program (this time, it's"sdclt.exe") into a newly-created folder, with random string (like "Okuo") under the "%appdata%" folder. Meanwhile, it reads the chosen dll ("slc.dll") into memory and then it overrides its data with the malicious module obtained from the response to the second packet. Finally, Dridex calls the API WriteFile() to save it to the same folder of the copied Windows program. From now on, whenever the Windows program—"sdclt.exe"—starts, it loads and executes the "slc.dll" that contains the malicious module.

Dridex then creates a scheduled task in the infected Windows system to achieve persistence on the victim's machine. The action of the task is just to start the copied Windows program (i.e. "sdclt.exe") and it is triggered to repeat this action every 30 minutes.

Figure 7.3 shows a screenshot of "Task Scheduler" with the added task named "Tixvzwbtojdsmg", as well as the copied "sdclt.exe" and "slc.dll" files in the folder "Okuo".

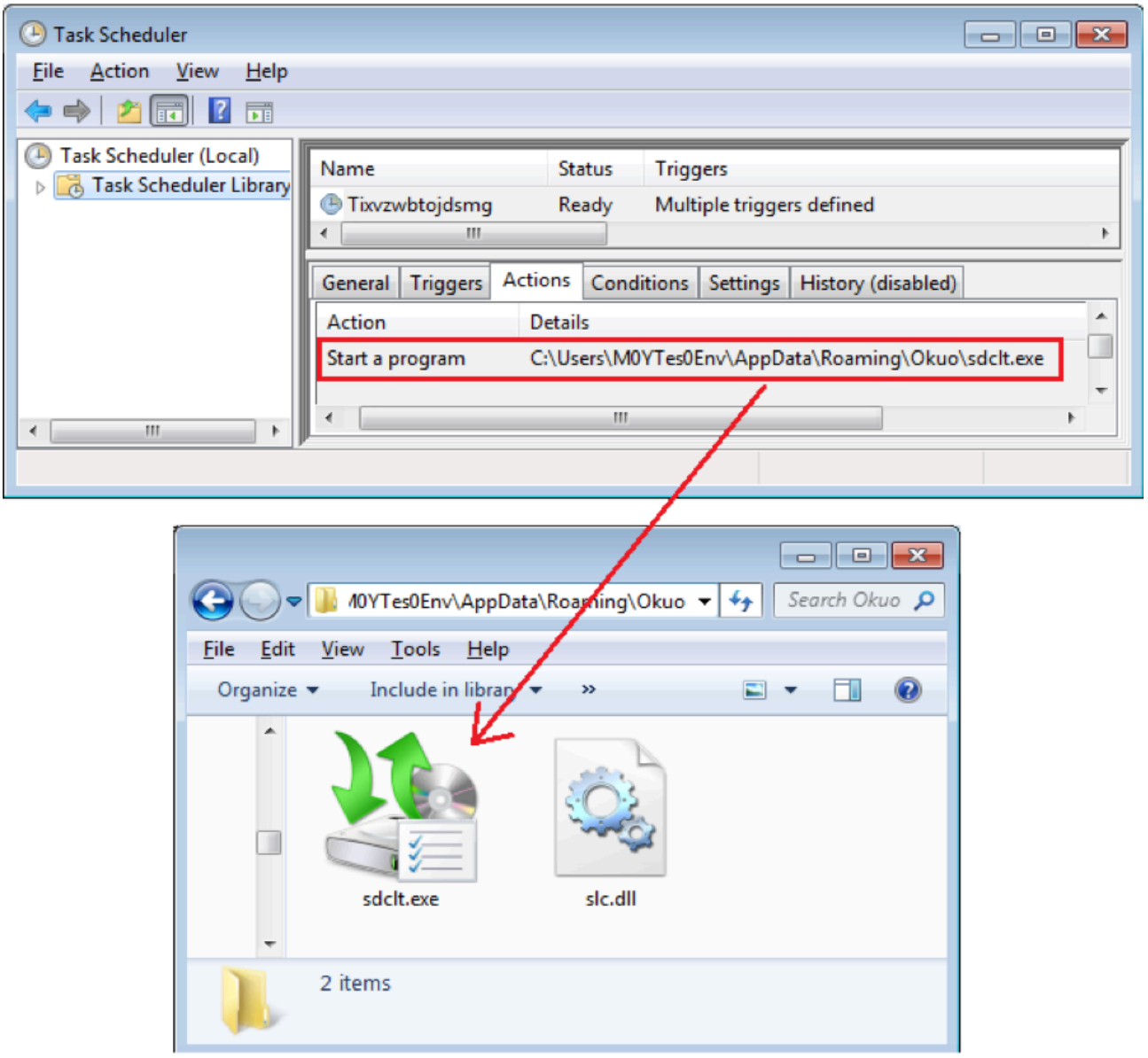

Figure 7.3 – Added scheduled task and copied Window program and dll files.

Other than adding to the scheduled task, it also calls the API CreateProcessW() to run “sdclt.exe” once just after it has been deployed.

Before Dridex exits, it sends a packet with the ID 0x69BE7CEE to inform the C2 server that the malicious payload has been successfully installed on the victim’s machine. Figure 7.4 is a code snippet about to generate and send this packet.

```
loc_72C56330:                          ; CODE XREF: entry_to_handle_C2_packets+352↑j
            xor     eax, eax
            mov     edx, 69BE7CEEh  ; ;; Packet type ID
            push    eax             ; 0
            push    eax             ; 0
            lea     ecx, [esp+108h+var_28]
            call    Generate_Packet_and_Send ; It send packet 0x69BE7CEE to inform C2 server that
                                    ; the molicious module has installed.
            lea     ecx, [esp+108h+var_28]
            call    _call_RtlFreeHeap  ; RtlFreeHeap([ecx+8])
            jmp     short return1
entry_to_handle_C2_packets endp
```

Figure 7.4 – Code snippet of handling packet 0x69BE7CEE

Conclusion - Dridex Variant

You have now learned how this Dridex campaign is run, including the phishing email, how the malicious code inside the attached Excel document is executed to extract an HTML application file, and finally, how a Rundll32.exe is called to execute the downloaded Dridex payload file.

I elaborated on how this variant of Dridex communicates with its C2 server, the fields contained in the packet, how it asks the C2 server for a malicious module, and how the module is deployed onto the infected system.

I also made a flow chart of how Dridex communicates with its C2 servers below in Figure 8.1. It clearly shows what packet and data was sent to the C2 server and when it received the malicious module. It will help you better understand the entire process.

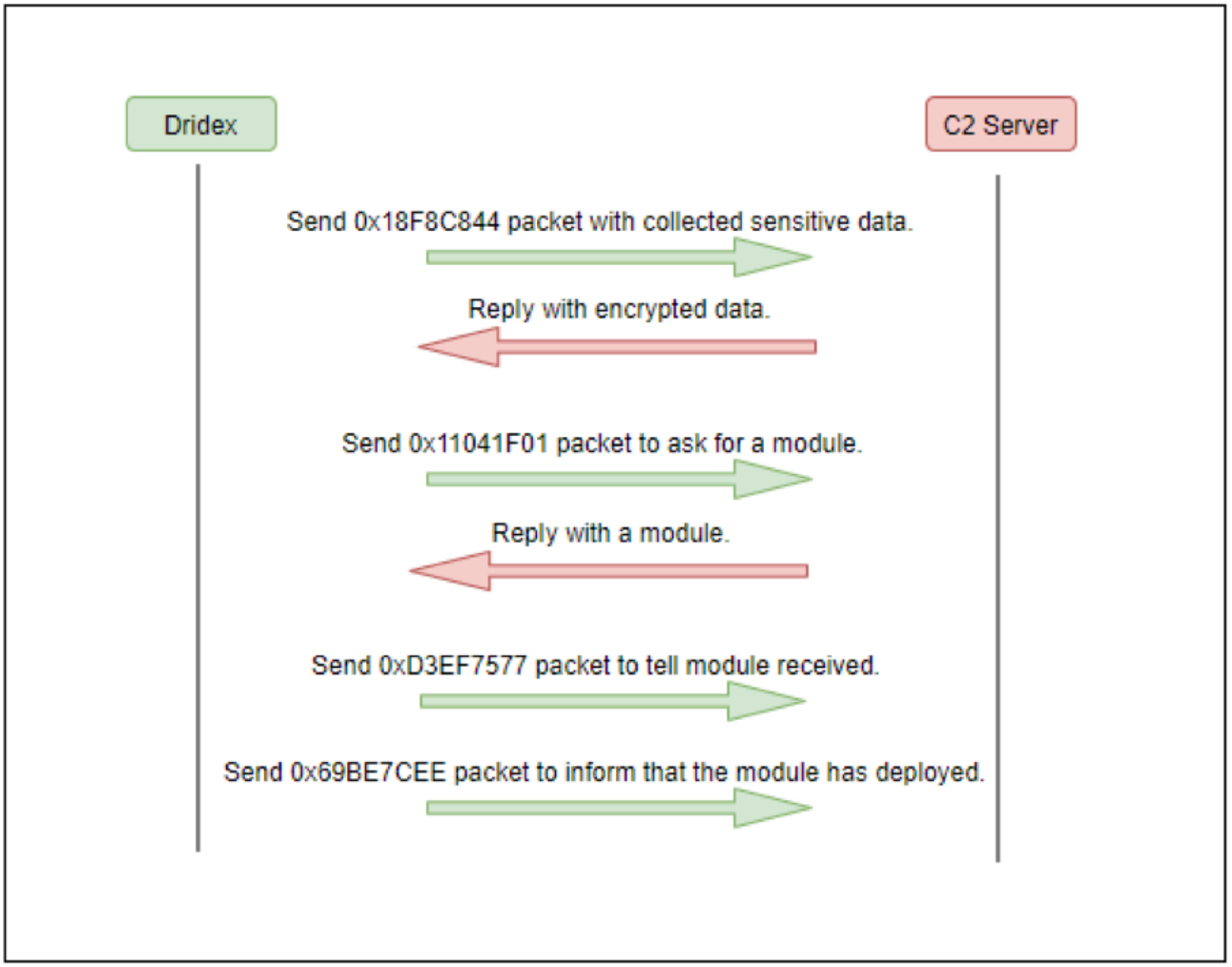

Figure 8.1 - Communication flow chart between Dridex and its C2 server.

Fortinet Protections

Fortinet customers are already protected from this malware by FortiGuard's Web Filtering, AntiVirus and FortiEDR services, as follows:

The downloading URLs have been rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

The attached Excel document and downloaded Dridex payload file are detected as "**MSExcel/Dridex.AC!tr**" and "**W32/Dridex.HMAH!tr**" and are blocked by the FortiGuard AntiVirus service.

FortiMail users are protected by FortiGuard AntiVirus, which detects the original Excel document as a malicious attachment in the phishing email.

FortiEDR detects the downloaded executable file as malicious based on its behavior.

IOCs

URLs:

"hxxps[:]//assettagger[.]saleseos[.]com/Classes/PHPExcel/Shared/JAMA/examples/RLFBubHuLTnm[.]php"
"hxxps[:]//reportingdashboard[.]mobilisedev[.]co[.]uk/includes/6WSSUhQrM[.]php"
"hxxps[:]//loans[.]uhuruloans[.]com/wp-includes/sodium_compat/namespaced/Core/ChaCha20/X8av4FUl7STEot3[.]php"
"hxxps[:]//practice[.]haylawdesign[.]com/wp-content/themes/twentynineteen/template-parts/content/jE4zYiuJ0iIw[.]php"
"hxxps[:]//kings[.]inforwizztechnologies[.]com/wp-content/plugins/aapside-master/elementor/widgets/tfOSpcBiZpffptj[.]php"
"hxxps[:]//pizzaplus[.]com[.]ng/wp-content/themes/twentytwentyone/template-parts/content/TZ6qTYLx7l[.]php"
"hxxps[:]//efshub[.]com/PHPMailer-master/examples/images/zunuLqqNQIGJPht[.]php"
"hxxps[:]//user[.]kasikoi[.]info/static/lib/ckeditor/skins/moono/2h80F9GORDfIB[.]php"
"hxxps[:]//deepsource[.]in/ncsitebuilder/css/flag-icon-css/flags/1x1/wcToKXeb7FxQ[.]php"
"hxxps[:]//ebanking[.]hentostreasury[.]com/account/umSqqCiyMf[.]php"

C2 Sever IP and Port:

"103.75.201.2:443"
"158.223.1.108:6225"
"165.22.28.242:4664"

Sample SHA-256:

[HF7.TRANS 2021.08.09.xlsb]
59C8D87A450F0647BEA930EBA1AA692B75D82DEF1358F1601C4FE9A561B4707E
[DTCZ SHIP_2021.08.09.xlsb]
C8065BD2A1443FF988E9BA95022554F6EE302E9BCB4082C3D9B2B8D74C5A4BE5
[icxbouzukiasgnpfvowz.dll]
6556E4029CF50C9538F4E02D0BCCA5356F28E6870E62838E164020A31B3DF096

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.

Learn more about Fortinet's free cybersecurity training, an initiative of Fortinet's Training Advancement Agenda (TAA), or about the Fortinet Network Security Expert program, Security Academy program, and Veterans program. Learn more about FortiGuard Labs global threat intelligence and research and the FortiGuard Security Subscriptions and Services portfolio.
