

Introduction

Credential stealing malware is commonly observed in the landscape of cyber attacks today. Zscaler ThreatLabz team has discovered many new types of stealer malwares across different attack campaigns. Stealers are malicious programs that threat actors use to collect sensitive information with various techniques including keylogging, cookie stealing, and sending stolen information to the Command and Control Server. Recently, ThreatLabz identified a novel windows based malware creating a registry key as FFDroider. Based on this observation, ThreatLabz named this new malware the Win32.PWS.FFDroider. Designed to send stolen credentials and cookies to a Command & Control server, FFDroider disguises itself on victim’s machines to look like the instant messaging application “Telegram”.

ThreatLabz observed multiple campaign related to FFDroider stealer in our zscaler cloud which arrived via the compromised URL download.studymathlive[.]com/normal/lilay.exe and are all connected by a malicious program embedded into cracked version of installers and freeware.

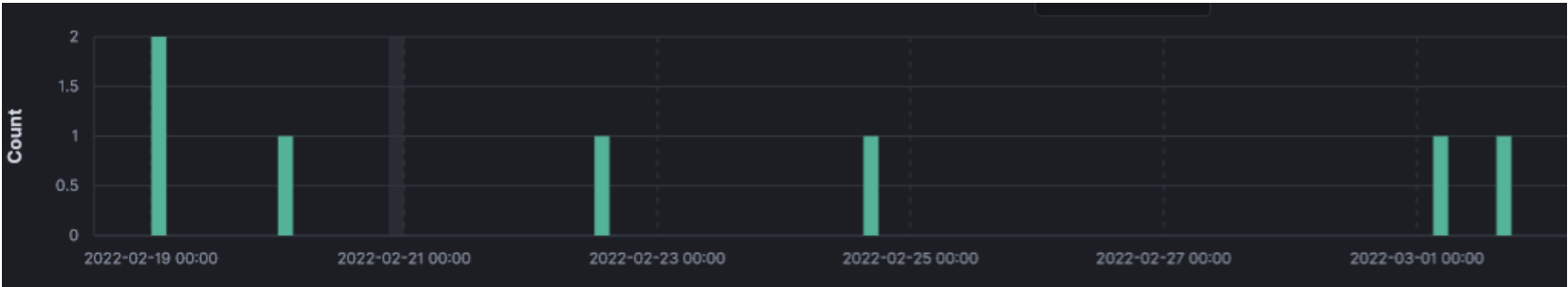


Figure 1: FFDroider campaign observed in Zscaler cloud

Key features of this attack

- Steals cookies and credentials from the victim’s machine.
- Targeting social media platforms to steal the credentials and cookies.
- The stealer signs into victims' social media platforms using stolen cookies, and extracts account information like Facebook Ads-manager to run malicious advertisements with stored payment methods and Instagram via API to steal personal information..
- Leverages inbound whitelisting rules in Windows Firewall allowing the malware to be copied at desired location.
- Attacker uses iplogger.org to track the infection counts.

The attack cycle

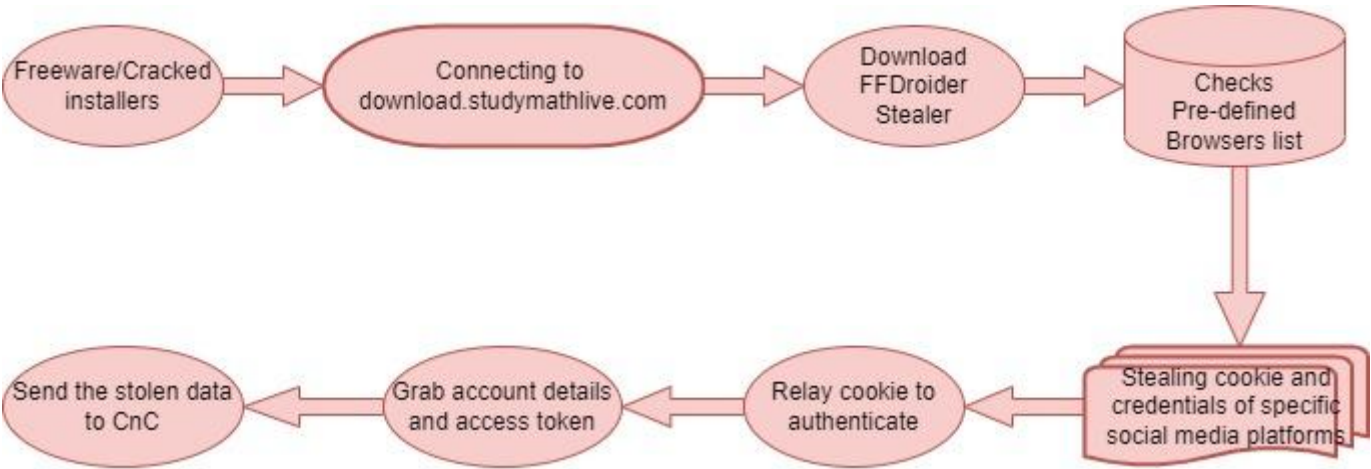


Figure 2: Attack cycle Infographic

This article focuses primarily on the dissection of the stealer and its functionality.

FFDroider stealer analysis

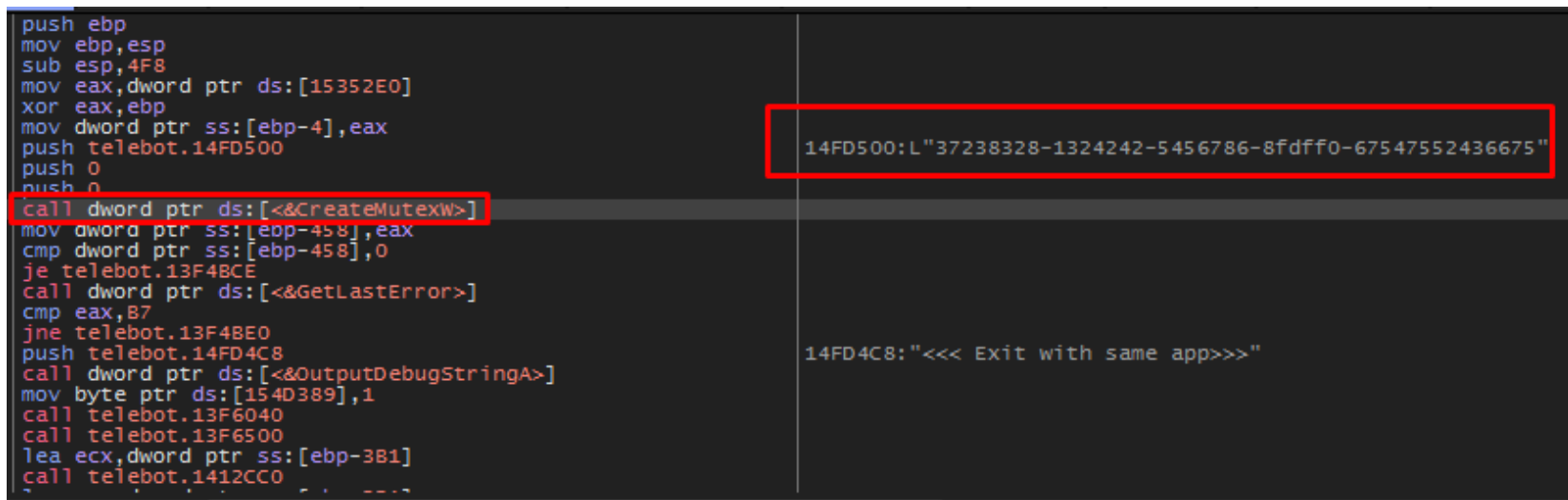
The FFDroider stealer is packed with the popular “ASPack v2.12” packer. To best understand how the stealer works, ThreatLabz unpacked. decompiled, and debugged the malware, performing the following tasks during execution:

- To detect the full malware campaign across the Zscaler cloud, researchers unpacked the file to expose the PDB path: F:\FbRobot\Release\FbRobot.pdb

debugger-stamp	0x62175388 (Thu Feb 24 09:44:40 2022 UTC)
path	F:\FbRobot\Release\FbRobot.pdb
Guid	2E252F00-4D9F-4E94-82A-5F72DBB77080

Figure 3: PDB path

- During execution the stealer creates a Mutex to avoid reinfecting the host with different instances of the same malware. The observed mutex value is: “37238328-1324242-5456786-8fdff0-67547552436675”



```
push ebp
mov ebp, esp
sub esp, 4F8
mov eax, dword ptr ds:[15352E0]
xor eax, ebp
mov dword ptr ss:[ebp-4], eax
push telebot.14FD500
push 0
push 0
call dword ptr ds:[<&CreateMutexW>]
mov dword ptr ss:[ebp-458], eax
cmp dword ptr ss:[ebp-458], 0
je telebot.13F48CE
call dword ptr ds:[<&GetLastError>]
cmp eax, 87
jne telebot.13F48E0
push telebot.14FD4C8
call dword ptr ds:[<&OutputDebugStringA>]
mov byte ptr ds:[154D389], 1
call telebot.13F6040
call telebot.13F6500
lea ecx, dword ptr ss:[ebp-3B1]
call telebot.1412CC0
```

14FD500:L"37238328-1324242-5456786-8fdff0-67547552436675"

14FD4C8:"<<< Exit with same app>>>"

Figure 4: Mutex name created by malware

- To make copies of itself for further execution, the malware creates a Directory in the Documents folder named “VlcpVideov1.01”.

Figure 5: Creating a copy of itself in the desired directory with a renowned application icon.

- Then it executes a String Decryption Routine which is basically a XOR Decryption loop amongst the encrypted string and the key where in it decrypts the following strings which include DLL and API names which are further loaded and fetched using LoadLibraryA() and GetProcAddress() WinApi’s

1. “Wininet.dll”
2. “InternetGetCookieExW”
3. “ieframe.dll”
4. “IEGetProtectedModeCookie”
5. “Netapi32.dll”
6. “NetWkstaGetInfo”
7. “NetAPIBufferfree”
8. “Advapi32.dll”
9. “Iphapi.dll”
10. “RegCreateKey”
11. ‘GetAdaptersInfo
12. “FFDroider”

Figure 6: String Decryption Routine

- To decrypt strings across the malware sample, ThreatLabz rewrote the XOR decryption logic in python, shown in the screenshot below. The complete decryption code snippet can be found in the Appendix section of this article.

Figure 7: String decryption emulation in Python

- Inspiring the name, the FFDroider stealer uses the decrypted string and RegCreateKey() function to create the registry key: “HKCU\Software\ffdroider\FFDroider”.

Figure 8: Creates a registry key named “FFDroider”

- Then the malware creates multiple threads using CreateThread() to speed the theft of cookies and credentials while hindering reverse engineering.

An initial GET request is sent to the Command & Control Server along with the filename via WinHTTPSndRequest().

- GET Request: http[:]//152[.]32[.]228[.]19/seemorebty/il.php?e=<filename>
- Referrer: https[:]facebook[.]com
- Previously a Cobalt Strike server according to third-party threat intel sources

Figure 9: Initial request to the C&C server logs the filename and IP address of the infected host.

The response to this request is an iplogger.org URL which is used to log the Public IP address of the environment where the malware has been detonated and might be used by the attackers to track location and IP addresses details of the victim. After analyzing the statistics of multiple Embedded iplogger URLs we can see how the IP addresses have been logged in the screenshots below. IPLogger URL: <https://iplogger.org/logger/ey4zrs2miAY6>

Figure 10: IP address of Infected host logged using iplogger.org

- The malware further initiates the Cookie and Credential Stealer functionalities and targets the following browsers and websites:

- Target Browsers:

1. Google Chrome
2. Mozilla Firefox
3. Internet Explorer
4. Microsoft Edge

Figure 12: List of target browsers

- Target Websites:

1. www.facebook.com
2. www.instagram.com
3. www.amazon.ca/cn/eg/fr/de/in/it.co.jp/nl/pl/sa/sg/es/se/ae/co.uk/com/com.au/com.br/mx/tr
4. www.all-access.wax.io
5. www.ebay.com
6. www.etsy.com
7. www.twitter.com

Figure 13: List of Target Web applications - uses stack strings

Understanding the Cookie and Credential Stealer Routine:

- Google Chrome: The FFDroider steals cookies and saved login credentials for the Chrome browser from the following data stores:

i) Reads and parses the Chromium SQLite Cookie store from the `C:\Users\<username>\AppData\Local\Google\Chrome\UserData\Default\Network\Cookies` and writes the file onto the path where the binary resides using `WriteFile()` named as “d”

Figure 14: Reads and parses the Chromium SQLite cookie store

ii) Reads and parses the Chromium SQLite Credential Store from the `C:\Users\<username>\Appdata\Local\Chrome\User Data\Default>Login Data` containing the saved credentials and writes that onto the path where the binary resides using `WriteFile()` named as “p” as the credential store is been locked in the AppData directory.

Figure 15: Reads and parses the “Chrome Saved Login Credentials”

iii) The Chrome SQLite Credential store includes attributes - `action_url`, `username_value`, `password_value`, out of this the `password_value` is encrypted using Windows Crypt API namely `CryptProtectData`. The malware in this case decrypts the encrypted password blob by first parsing the “Login Data” credential store by executing an SQL query such as “`select username_value, password-value FROM logins where origin_url like \'%ebay.com/%\';`” as seen in the screenshot below.

Figure 16: Execution of SQL queries across the Login Data Credential store for parsing the required credentials

The password cache is fetched from the output and passed to the `CryptUnProtectData()` function for in memory decryption, revealing clear-text credentials stolen from the targeted web application Credential Store.

Figure 17: Call to `CryptUnprotectData` to decrypt Saved chrome passwords in memory

iv) Then it reads and parses the local state cookies stored at `C:\Users\<username>\AppData\Local\Google\Chrome\UserData\LocalState` and uses `WriteFile()` to write to the path named “u” where the binary resides.

Figure 18: Reads and parses the local state chromium cookies

Here the Cookies are also decrypted in memory using the CryptUnprotectData() function by loading the json “Local State” file and filtering out the two parameters: os_crypt and encrypted_key and then decrypted using the CryptUnprotectData() and stored in memory.

Figure 19: Decrypts the Local state Cookies in memory using CryptUnprotectData

The following decryption routine takes place to steal the cookies and stored credentials for all the Chrome stores implementing the same process using the SELECT SQL queries via sqlite3 library to fetch the required value and then CryptUnprotectData() function to decrypt the cookies and credentials in memory as per the target website.

Figure 20: Different SQL Queries implemented in the binary to parse Cookie and Credentials stores

- Internet Explorer/Edge: The FFDroider Stealer gathers cookie,browser history and other user specific information from the internet explorer in the following way:

i) The malware executes InternetGetCookieRxW() function to retrieve the cookies for the target websites mentioned above (HTTP ONLY cookies are been read) if they are restricted the IEGetProtectedModeCookie() function is been used to access low integrity cookies for all the target applications during which it launches an another process “IElowutil.exe” which is a utility in place to access the low integrity cookies and processes.

Figure 21: Execution of InternetGetCookieExW & IEGet ProtectedMode Cookie to steal cookies from the Internet Explorer browser

It also reads the Appdata\Roaming\Microsoft\Windows\Cookies and fetches the Cookie and the URL details from the Cookie store along with that it also parses the Cookies,History and downloads from the Microsoft Edge WebCache:

C:\Users\<username>\Appdata\Local\Microsoft\Windows\WebCache\WebCacheV01.dat by copying it onto the place where the binary resides into the file named “d” which earlier had the chrome cookies.

Figure 22: Reads and parses the web cache file of the Edge browser to steal cookies, browsing history, and session data.

Furthermore, it reads and parses the Appdata\Roaming\Microsoft\Windows\History\History.IE5 and \Appdata\Local\Microsoft\Windows\Temporary internet files\Content.IE5 which would allow the malware to read the browsing history and the The Internet Explorer cache from the stores wherein it queries for attributes such as URL visited,Filename other metadata for the target websites. Also the malware plans to steal saved VPN/Dial Up credentials from the \Appdata\Microsoft\Network\Connections\Pbk\rasphone.pbk & \Pbk\rasphone.pbk if present, by leveraging the Rasapi32.dll API calls.

- Mozilla FireFox: The FFDroider Stealer steals and parses the cookies from the Firefox browser initially by reading the profiles.ini (Path: C:\Users\<username>\Appdata\Roaming\Mozilla\Firefox\profiles.ini) which consists of the name(s) of the used profile(s). Further the malware uses those profile names to access SQLite cookie stores named: “cookies.sqlite” (Path: C:\Users\<username>\Appdata\Roaming\Mozilla\Firefox\Profiles\<profilename>\cookies.sqlite) for the user profiles. The cookie attributes are then parsed by using few SQL Queries such as “SELECT host,name,path,value,expiry FROM moz_cookies to fetch the Hostkey,Name,Path,Value and the Expiry of the Cookies stored in the Firefox Cookie store.

Figure 23: Reads and parses the Mozilla Firefox SQLite Cookie store.

Facebook and Instagram Data Gathering:

The FFDroider Stealer holds another functionality wherein if the malware grabs cookies for facebook.com or instagram.com from any of the target browsers the cookies are replayed to www[.]facebook[.]com and www[.]instagram[.]com to gather intelligence from the Users Facebook or Instagram accounts.

Facebook: Following requests were executed by the malware post grabbing the cookie values:

i) Initially it sends a GET request to the https[:]//facebook[.]com along with the stealed facebook cookie from the target browsers to check whether the malware is able to authenticate using the following set of stealed cookies.

Figure 24: Passes the stolen facebook cookie to facebook[.]com for authentication

ii) If the cookies are valid and provide proper authentication, further it sends a GET /settings with the Access Token to facebook.com along with the authenticated stealed cookies in order to fetch the User Account settings of the Compromised Account.

Figure 25: Grabs Account details and Access Token from the Compromised facebook account

iii) Further it starts enumerating whether the compromised user account is a business account and having access to Facebook Ads Manager and fetch the following details using the stealed cookies by parsing the responses:

- Fetch Account Billing and Payment Information from the Facebook Adsmanager
- Fetch users Facebook pages and bookmarks.
- Number of facebook friends and other user related information

Figure 26: Fetches Account Billing information from Ads manager along with the Facebook Bookmark & Pages information.

The following information may be leveraged later to run malicious advertisements from the victims account and utilize the compromised accounts payment method to spread the malware further.

Instagram:

In the case of instagram, whenever the malware grabs any instagram cookies from the target browser cookies stores, it performs the following routine in to steal user account details from the Instagram account as follows:

i) Initially it sends a GET request to the `https[:]//instagram[.]com` along with the stealed instagram cookie to check whether the malware is able to authenticate using the following set of stealed cookies and parses the html response.

Figure 27: Passes the Stolen Instagram cookie to `instagram[.]com` for authentication

ii) If there is a valid response, it sends the next GET request to the instagram server with the username of the compromised account `GET /<username>` which was parsed from the previous response and basically visits the profile page.

Figure 28: Visits the profile of the Victim in order to grab required user information

iii) Further it sends another request: `GET /accounts/edit/` to `www[.]instagram.com` which opens up the Account settings containing all the personal account related information such as the account email address, mobile number and other details of the compromised account.

Figure 29: Grabs Personal information such as email address, phone number from the instagram account edit webpage.

Furthermore, in the same manner all of the account related information such as username,password,mobile number and other account details are been grabbed from the target websites in the form of cookies,saved credentials and fetched using different API's and then sent to the command and control server in an encrypted manner to the threat actors as discussed below.

Exfiltration of Stolen Information to the C2 Server:

Then the malware sends an HTTP POST request to the C2 server: `http[:]//152[.]32[.]228[.]19/seemorebty` along with the encrypted cache of data for exfiltration.

Figure 30: Encrypted request sent to Command & Control server for exfiltrating the encrypted data cache.

Such kind of encrypted data using modified base64 encoding is sent to the C&C from the infected system when a valid facebook account cookie was provided to the malware in the chrome browser. The decrypted json body can be seen in the screenshot below for Facebook related exfiltration where in a lot of Facebook user account information has been transmitted to the C2:

Figure 31: Decrypted Request consisting of the Stolen information from the compromised facebook Cookie

An Instagram user's personal account information including cookies, email password, Instagram userID, saved password, phone number are revealed in this decrypted request..

Decrypted JSON body:

Figure 32: Decrypted request showing sensitive data stolen from Instagram.

Encrypted request:

Figure 33: Encrypted request showing stolen Instagram account information.

Also an inbound whitelisting rule in the Windows Firewall as shown below in the screenshot which requires administrative privileges.

Figure 34: Inbound Firewall rule added by the FFDroider malware which would further enable disallowed connections to the infected host.

Downloader Functionality:

After stealing and sending across the stolen details from the target browsers and websites to the Command & Control. The FFDroider Stealer further it tries to upgrade itself in a fixed interval of time by downloading other modules from an update server by sending across request to the following as mentioned - URL:http[:]//186[.]2[.]171[.]17/seemorebtu/poe.php?e=<filename> by calling wininet.dll APIs such as InternetOpenUrlW and InternetReadFile. The module is written onto the disk in the previously created “VlcpVideov1.01” directory as “install.exe”.

Figure 35: Malware sends request to the Update server to upgrade itself in a fixed interval of time.

Debugging Functionality:

During the process of reverse engineering the malware, we came across a functionality which was developed by the malware authors to debug the malware. If the filename at the time of execution is test.exe then the malware goes into its debug state and pops up messages on every loop where in, it prints out the stolen cookies and the final json body which is to be sent to the C&C from each and every browser for the target websites as shown in the screenshot below.

Figure 36: Debugging functionality implemented by the malware authors

Cloud Sandbox detection

Figure 37: The Zscaler Cloud Sandbox successfully detected the malware.

In addition to sandbox detections, Zscaler’s multilayered cloud security platform detects indicators at various levels

[Win32.PWS.FFDroider](#)

Conclusion Over the years, Stealer’s became one of the most commonly used malware in any cyber attack campaign. The Zscaler ThreatLabz team will continue to monitor this attack, as well as others, to help keep our customers safe.

Mitre table

T1055	Process Injection
T1027	Obfuscated Files or Information
T1027-002	Software Packing
T1003	OS Credential Dumping
T1016	System Network Configuration Discovery
T1018	Remote System Discovery
T1057	Process Discovery
T1082	System Information Discovery
T1083	File and Directory Discovery
T1005	Data from Local System

Indicators of Compromise:

Hash:

beb93a48eefd9be5e5664754e9c6f175

e8c629383fe4b2c0cbf57b0d335fc53f

6a235ccfd5dd5e47d299f664d03652b7

b11fd571c6cc4b8768f33a2da71fbb6e

URL:

download[.]studymathlive[.]com/normal/vinmall880[.]exe

download[.]studymathlive[.]com/normal/lilay[.]exe

download[.]studymathlive[.]com/install/vinmall1[.]exe?_sm_byp=iVVkm23V4sqBFtNM

download[.]studymathlive[.]com/install/vinmall1[.]exe?_sm_byp=iVVJWHH51nHRJTzP

Appendix:

1. FFDroider Stealer String Decryption Python Code -

- [Security Research](#)
- [Insights and Research](#)

Authors

[Avinash Kumar](#)

[Niraj Shivtarkar](#)

Recommended for You

[Analysis of Spring Cloud Framework Vulnerabilities](#)

[Analysis of BlackGuard - A New Info Stealer Malware Being Sold In A Russian Hacking Forum](#)

[Conti Ransomware Attacks Persist With an Updated Version Despite Leaks](#)

[Lapsus\\$ Attack on Okta: How to Evaluate the Impact to your Organization](#)