New MultiloginBot Phishing Campaign

zscaler.com/blogs/security-research/new-multiloginbot-phishing-campaign

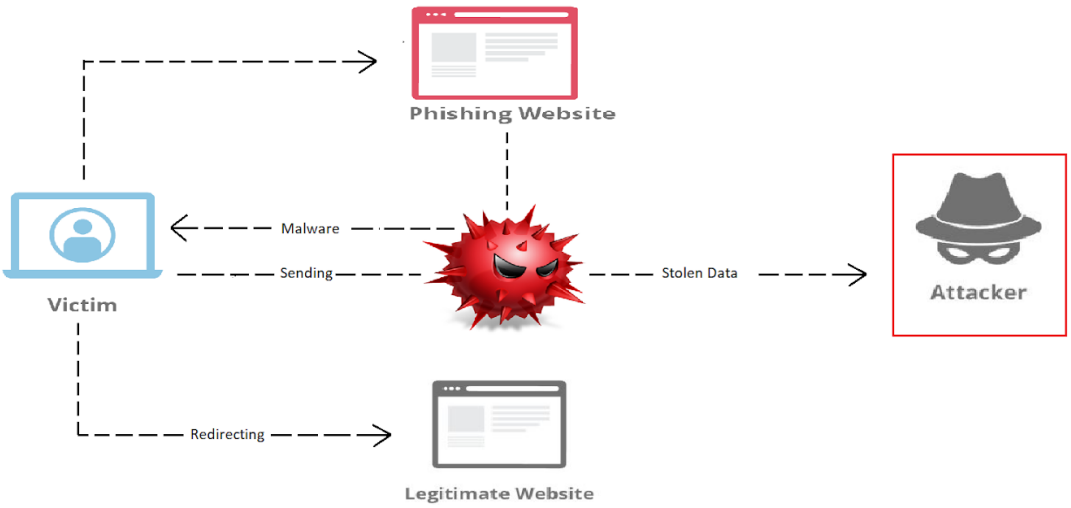
Multilogin is an application designed to make it easier to log into multiple accounts on a single website or platform simultaneously. Recently, Zscaler ThreatLabz has come across a live phishing campaign that is targeting genuine Multilogin users by tricking the users into downloading a malicious installer. The installer is hosted on newly registered websites "multilogin-uk[.]com" and "multilogin-us[.]com" (registered on September 2nd 2021) which are a lookalikes of the legitimate website "multilogin[.]com". The threat actor has taken great care to match every detail, starting from website layout to the url pattern used for downloading the application, in order to impersonate as the legitimate website.

The malicious installer installs a stealer (named as multilogin), written in Dotnet, on the compromised machine. This stealer gathers sensitive information from the victim's system and sends it to its telegrambot in a zip format.

This blog aims to describe the behavior of the installer and the main functionalities of the stealer.

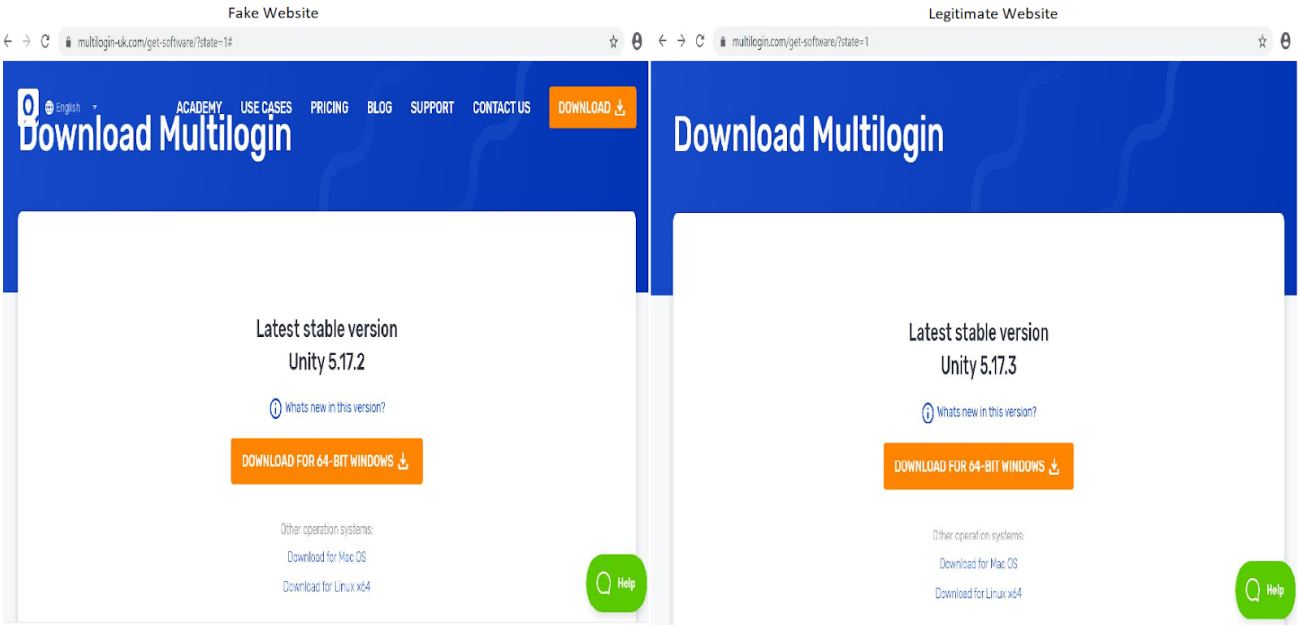
Attack Flow:

The below snapshot shows the delivery mechanism and attack chain of the malware.



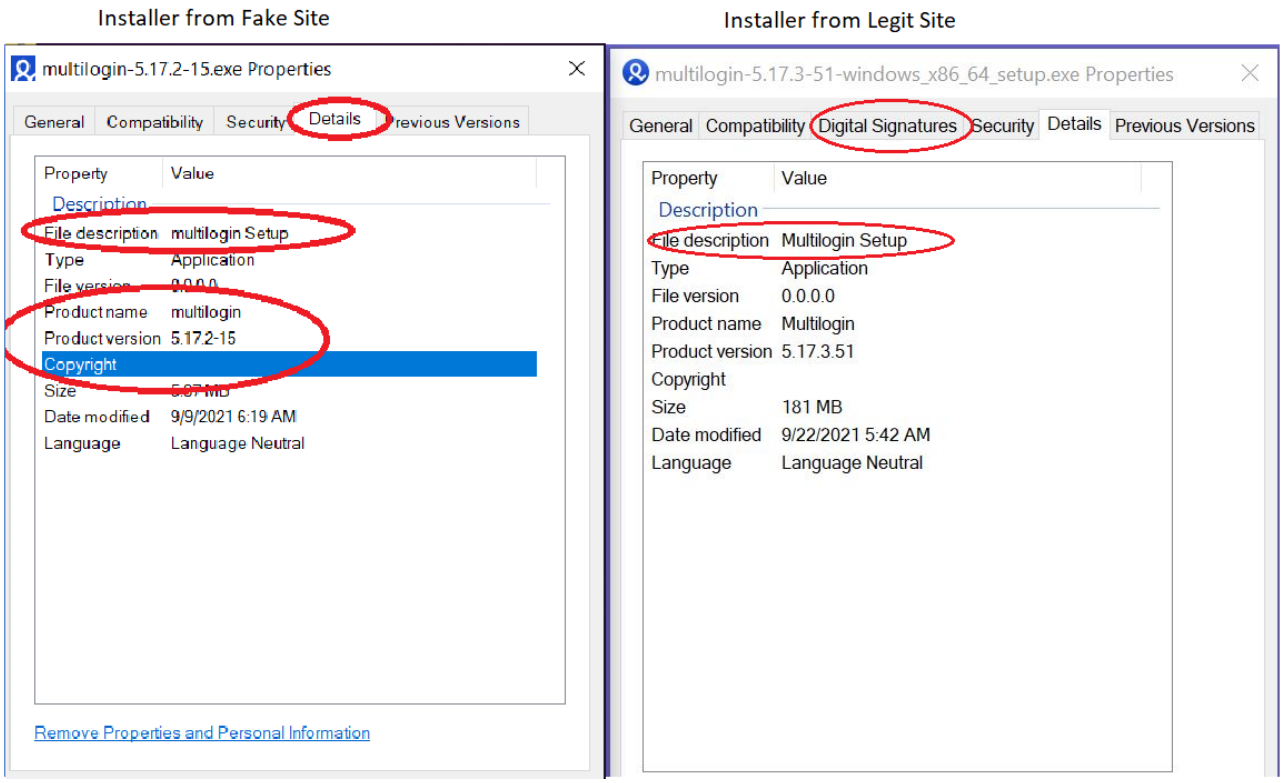
Technical Analysis

As a first step, the threat actors have cloned the legitimate website, giving it a similar domain name, in order to trick the user into visiting the phishing website and downloading the malicious installer hosted on it. The below snapshot shows the difference between the fake and legitimate sites.

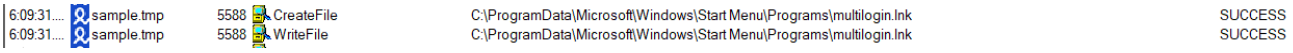


.

For the purposes of analysis, we will look at the Installer with MD5 hash: **9986d6836e6b4456fd38e7d5b036c727**, which is an Inno package unsigned binary. The below snapshot shows the comparison between the installer downloaded from the fake site and the installer from the legitimate site.



Like the normal installer, the malicious installer creates a full environment, starting with registry changes, then creating required folders (explained later), for the effective execution of the malware. In order to achieve persistence on the compromised machine, the malicious installer creates a shortcut file in the ***All Users startup folder*** as can be seen in the below snapshot.



It is to be noted that the installer drops the malware at the user's selected installation path at the time of installing the application.

Required folders: The installer creates a folder named "**Item"** and the following sub-folders, which shall be checked and used by the malware later:

- **AutoFills**: Consists of text files containing browser's autofill information.
- **Cookies:** Consists of text files containing browser's cookies information.
- **IP:** Consists of a text file containing IP address information.
- **Passwords:** Consists of text files containing user's login information.

After successful installation, the final GUI (Graphical User Interface) comes up with an enabled check box to launch the application named as multilogin (hereinafter referred to as malware). After clicking the "**Finish**" button, the malware comes into play. Now, let's get into the code to understand the functionality of the malware.

Information Gathering

Before starting stealing activities, the malware checks the required folders in the system and then executes its functions to collect information from the compromised machine, explained below:

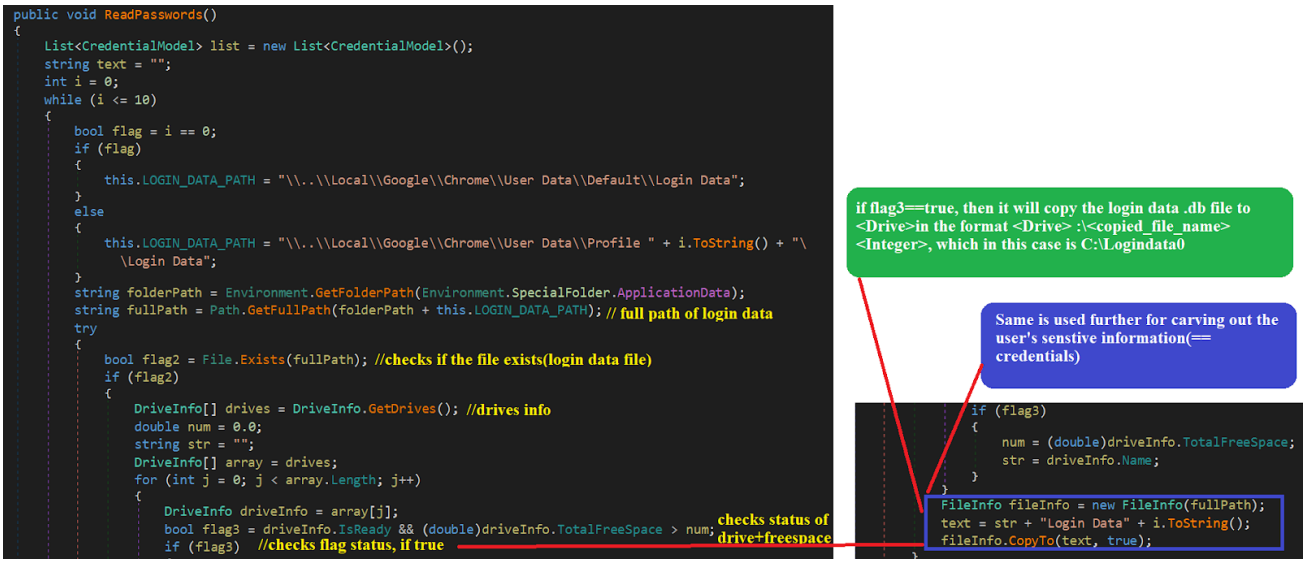
1.) IP Address Information:Firstly, the malware collects the IP address by making a web request to "*checkip.dyndns.org*"and stores the collected information at "**<Installation_Path>\Item\IPAddress\IPAddress.txt",** in the format **<IP_Address>: <Country_Name>,** asexplainedin the below snippet**.**

It is to be noted that the above code also acts as a checkpoint for an internet connection--that is, if the malware doesn't get a response, then the malware crashes.

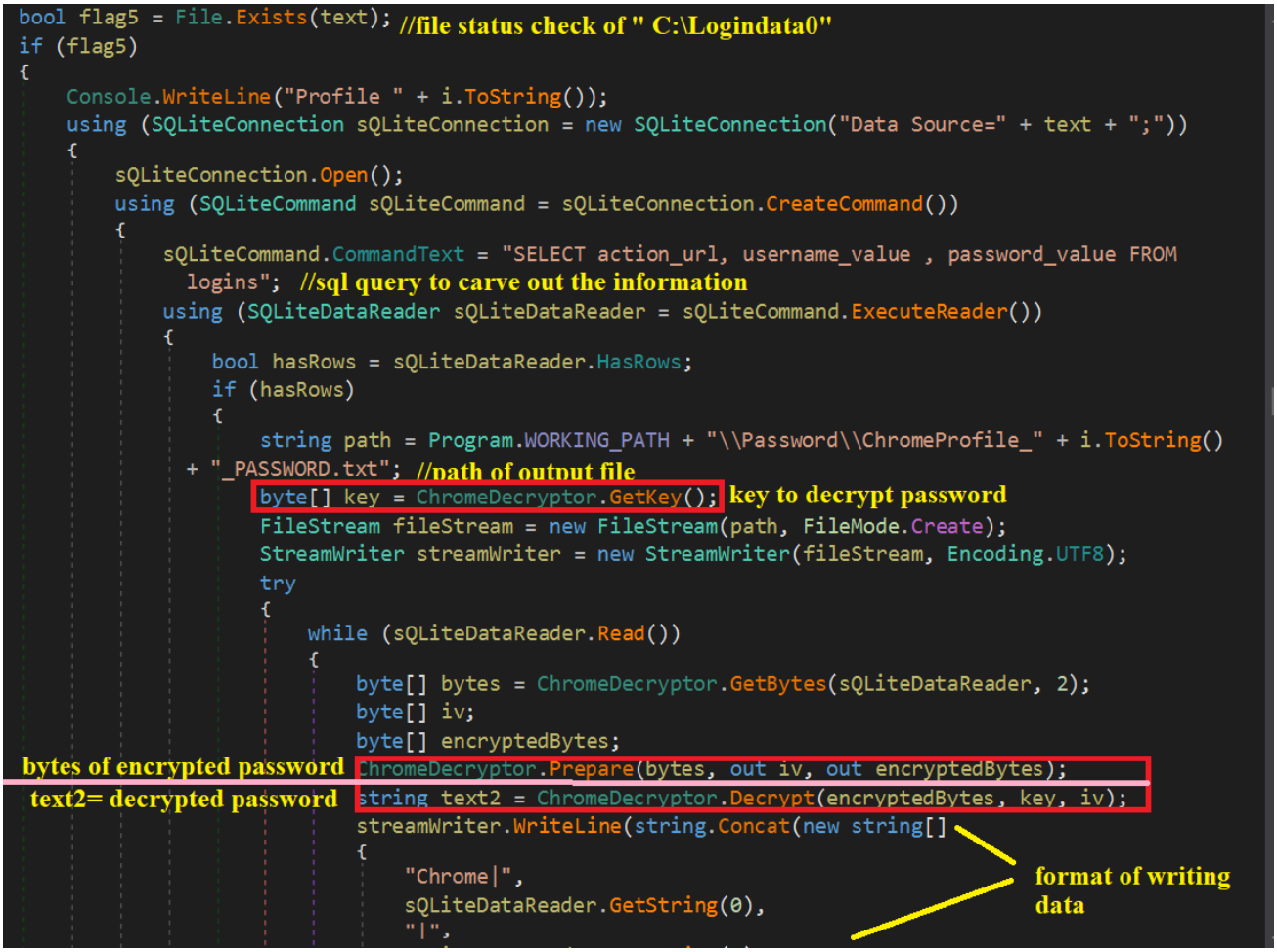
2.) User's Login data:

The steps to gather the login data of the user are as follows:

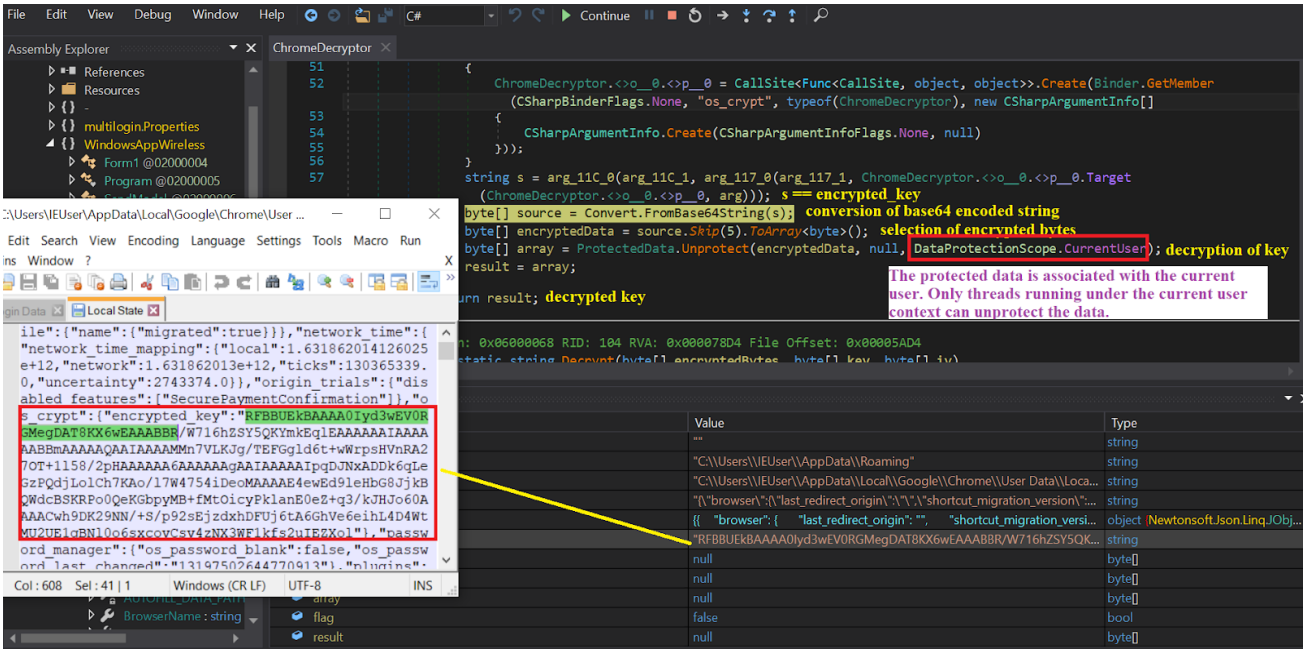
> Creates a copy of existing login data file to the destination file that is named as 'C:\LoginData0' in this case. The below snippet shows the detailed steps.



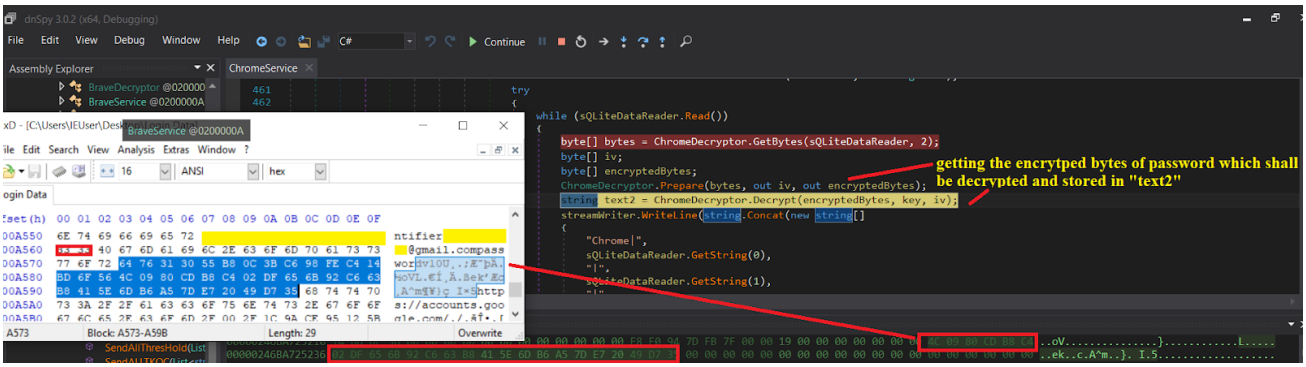
The below snippet shows how the SQL query is being executed against the newly created i.e **C:\LoginData0** file to carve out the login information. And then the information is decrypted and stored in a new file placed at **<Installation_Path>\Item\Password\ <Browser_name>Profile_<Integer>_PASSWORD.txt>**.



The login data contains passwords in an encrypted format, so the malware first gets the key, which shall be used further to decrypt the encrypted password. The below snapshots explain the same.



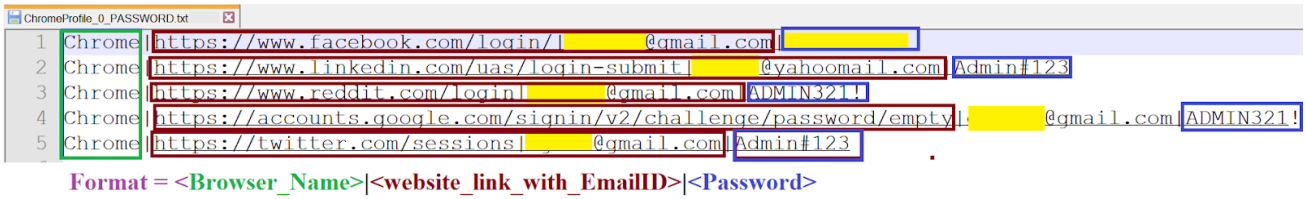
The next step is to get the encrypted data (password), which shall be passed to the next function named “Decrypt” explained in the next step.



After getting the key and the encrypted data, the malware will decrypt them by using the below code.

```
public static string Decrypt(byte[] encryptedBytes, byte[] key, byte[] iv)
{
    string result = string.Empty;
    try
    {
        GcmBlockCipher gcmBlockCipher = new GcmBlockCipher(new AesEngine());
        AeadParameters aeadParameters = new AeadParameters(new KeyParameter(key), 128, iv, null);
        gcmBlockCipher.Init(false, aeadParameters);
        byte[] array = new byte[gcmBlockCipher.GetOutputSize(encryptedBytes.Length)];
        int num = gcmBlockCipher.ProcessBytes(encryptedBytes, 0, encryptedBytes.Length, array, 0);
        gcmBlockCipher.DoFinal(array, num);
        result = Encoding.UTF8.GetString(array).TrimEnd("\r\n\0".ToCharArray());
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
    }
    return result;
}
```

The below snippet shows the format in which malware will store the user login data information.

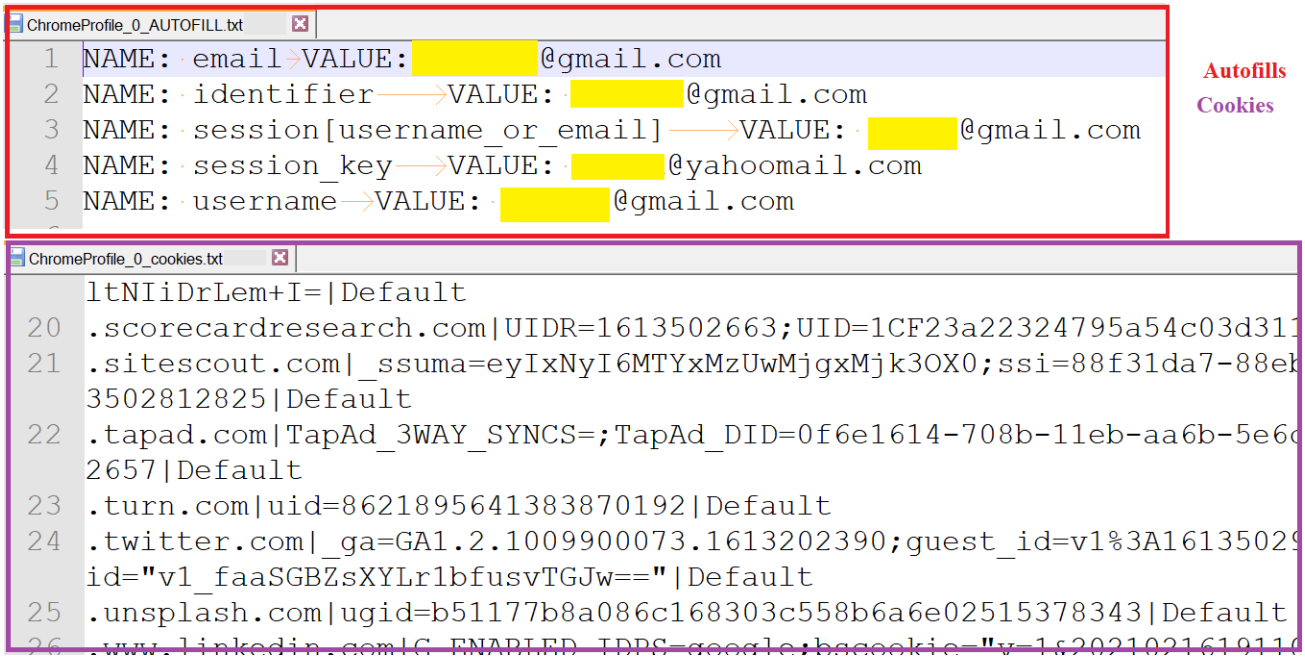


3.) Autofill and Cookies:

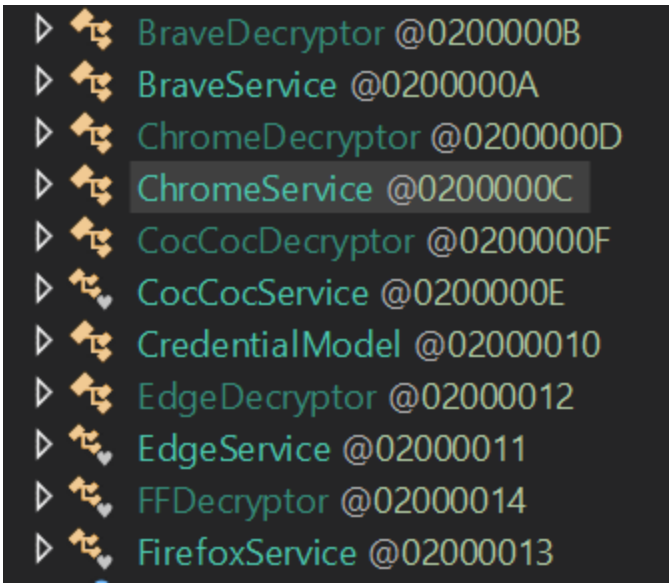
The malware uses a similar mechanism to the one explained above to collect autofill and cookies information and stores the respective data in a file placed at **<Installation_Path>\Item\Autofill\<Browser_name>Profile_<Integer>_AUTOFILL.txt>** and **<Installation_Path>\Item\cookies\ <Browser_name>Profile_<Integer>_cookies.txt>,** respectively. The below table depicts the targeted file and the sql queries used to extract the information.

Type	Targeted file	Newly Created file Path	Sql Query	Decryption of bytes?
Autofills	Webdata	C:\Webdata0	**Select name, value FROM autofills**	False
Cookies	cookies	C:\cookies0	**SELECT host_key, is_httponly, path, is_secure, expires_utc, name, encrypted_value, creation_utc FROM cookies order by host_key,creation_utc desc**	True

The below snippet shows the format in which malware will store the related information.



Note:- *Similar code and logic is there for stealing information from other browsers (except Firefox). The below snapshot shows the list of browsers targeted by the malware author.*



***<Browser>*Service** = Responsible for checking the targeted file location and then writing of the decrypted data.

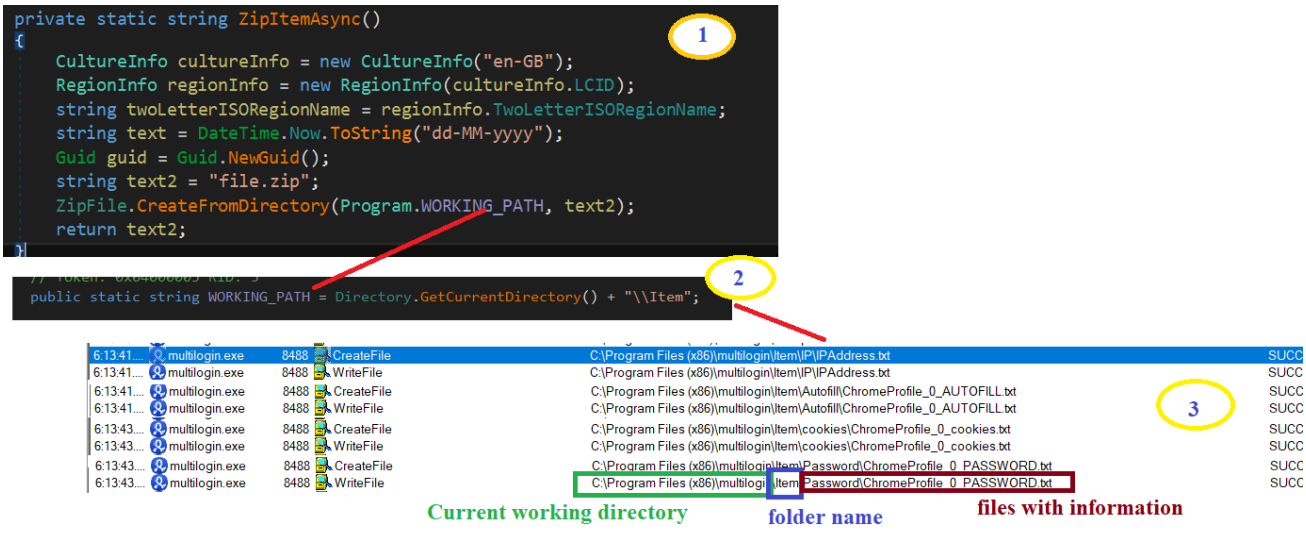
***<Browser>*Decryptor=** Responsible for decryption of the encrypted data

In the case of Firefox, the malware targets *cookies.sqlite, signons.sqlite* and *logins.json* files to carve out the sensitive information and stores the data in **<Installation_Path>\Item\cookies\FFProfile_Cookies.txt** and **<Installation_Path>\Item\Password\FFProfile_PASSWORD_.txt** respectively. It is to be noted that for decryption of encrypted data, the malware uses **PK11SDR_Decrypt**" API of ***nss3.dll***.

Zip file creation code

After stealing all the data from different browsers, the malware then creates a zip file which shall be sent to the command and control (C2) server. The below code explains the same.

```
finally
{
    string text = Program.ZipPath + "\\" + Program.ZipItemAsync();
```



Telegram Bot

After zip creation, in order to transfer the zip file, the malware initiates C2 communication to its Telegram bot, using a hard-coded token. The snippets of the code are shown below.

- Setting up of required protocols + Reading of zip file + Sending data (labeled 1)
- Full telegram bot url (labeled 2)

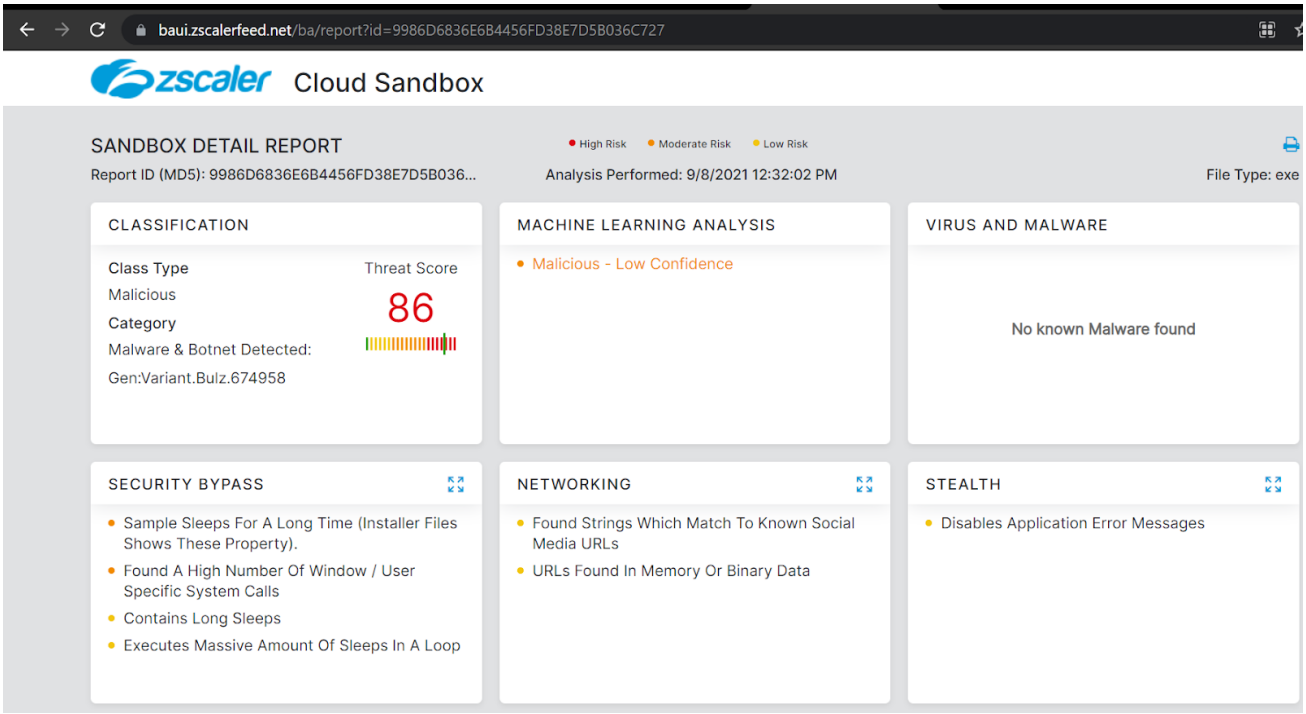
Post-Stealing Activities

After successfully performing the stealing activities, in order to leave no trace, the malware displays a pop-up with a false message to update the application and asks the user for confirmation to proceed ahead. Once the user responds, then the malware opens a legitimate link in the foreground and deletes the created zip file in the background, irrespective of the option chosen by the user. It means that even if the user selects "No," the code will execute in the same pattern.

How to Defend Against This Attack

Zscaler's multilayered cloud security platform detects these indicators at various levels: Win32.Backdoor.MultiloginBot.

The Zscaler sandbox coverage is below:



MITRE Att&ck Table

T1584	Compromise Infrastructure
T1547	Boot or Autostart Execution
T1555	Credentials from Password Stores
T1567	Exfiltration over Web Services
T1059	Command and Scripting Interpreter
T1005	Data from Local System
T1114	Email Collection
T1560	Archive Collected Data

T1606	Forge Web Credentials

IOCS

Below are information on IOCs, including MD5 hashes and URLs, that should be blocked.

MD5s

9986d6836e6b4456fd38e7d5b036c727

f991573756dbc778b52b84212c7a36c5

Phishing domains:

multilogin-uk[.]com

multilogin-us[.]com