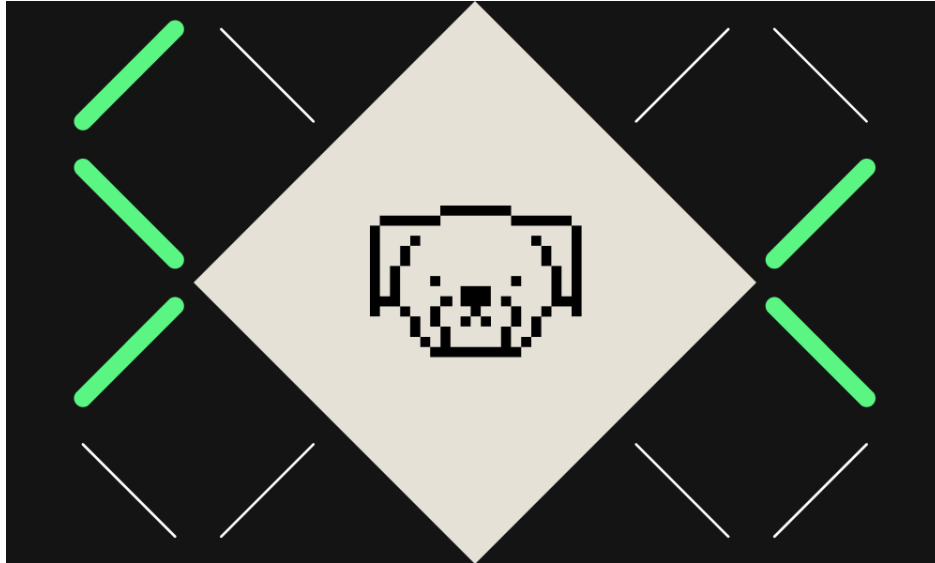


Introducing CylindricalCanine: The GoldenEyeDog subgroup responsible for the April DigiCert incident

By Aaron Walton

Published: 2026-07-15 · Archived: 2026-07-17 02:00:58 UTC



TL;DR

- Chinese cybercrime group, GoldenEyeDog, has been regularly updating their malware and tactics since 2015. We've observed them regularly leveraging code-signing certificates to bypass Windows's SmartScreen since 2024.
- They leverage several malware, including one we've seen primarily documented in Chinese language publications, and are calling it "Golden Gh0st RAT".
- In April 2026, GoldenEyeDog used their malware to access a support member's device at DigiCert, a code-signing certificate provider, and leveraged their access to steal certificates intended for DigiCert customers. This attack highlighted the capability of the malware and operators.

In this blog, we review the behavior and capabilities of a malware we call Golden Gh0st Loader and Golden Gh0st RAT. We believe these malware are used exclusively by a sub-group dubbed "[GoldenEyeDog](#)", a Chinese cybercrime group. In April 2026, the actors behind the malware were able to gain access to DigiCert to intercept code-signing certificates intended for DigiCert customers, and then used the certificates to sign their own malware. This piqued our interest in the malware, leading us to use DeceptionPro to monitor the malware over days in a controlled enterprise environment—and create a tool to decrypt the malware's network communications.

Expel-TA-0002 (AKA CylindricalCanine)

We've chosen to give a new name to this actor group to demarcate our visibility into their activity. We've identified the actors as a distinct part of a larger criminal group (GoldenEyeDog), which is often a distinction other organizations don't make.

CylindricalCanine are active users of a malware we're tracking as Golden Gh0st Loader and Golden Gh0st RAT. Golden Gh0st RAT is used primarily in phishing emails and/or submissions to support portals (these submissions may themselves be emails received by a ticketing system). These actors are using malware and targeting victims consistent with other Chinese cybercrime activity, including targeting finance organizations in the Asia Pacific region. The malware targets finance organizations in the Asia Pacific region.

Golden Gh0st RAT shares behavioral and tactical indicators with what Qi'anxin observed in [2020](#) and [2024](#). This malware also aligns with what Any.Run published as [Zhong Stealer](#) in February 2025, and the unnamed backdoor discussed [by Xiao Yi in an Aliyun community post](#) (December 2025), and also [by CyStack](#) (Feb 2026). Golden Gh0st RAT's main feature is remote access, and is a modified version of the [Gh0st RAT](#). We're choosing this name to recognize that connection. Gh0st RAT source code was originally published in 2008 and has been used in cyber attacks ever since. The Gh0st RAT payload is loaded by the first-stage malware we're calling Golden Gh0st Loader. Based on our analysis, we believe that CylindricalCanine is part of a larger group tracked by Qi'anxin as GoldenEyeDog (also known as APT-Q-27). This

conclusion is drawn from tactics, naming, and encryption techniques used by the actors in the malware which are consistent with Qi'anxin's publications.

A note on naming: In 2022, Qi'anxin stated that [GoldenEyeDog was rising to an APT \(advanced persistent threat\) group status](#), and has since labeled them an APT. This choice sometimes causes confusion since in American and European cybersecurity organizations, "APT" is often used for nation-state actors. However, Qi'anxin argues that GoldenEyeDog—with their use of zero-days, n-days, and modular attack chains—is a sophisticated threat actor similar to [FING](#) or [TA505](#). However, the designation of APT shouldn't be considered unusual: [Selena Larsen at Sleuthcon 2024](#) also noted cybercriminals have risen to APT sophistication levels, and failure to identify them as APT negatively impacts the response from law enforcement and the security industry.

The golden opportunity

We've tracked the malware over several years because of their regular use of code-signing certificates. That is, actors use code-signing certificates to keep Windows's SmartScreen from preventing unsigned EXE and MSI files from running. In the past, Extended Validation certificates helped ensure that SmartScreen wouldn't launch at all, but Microsoft has changed their system to give certificates less trust automatically, requiring files to build trust over time.

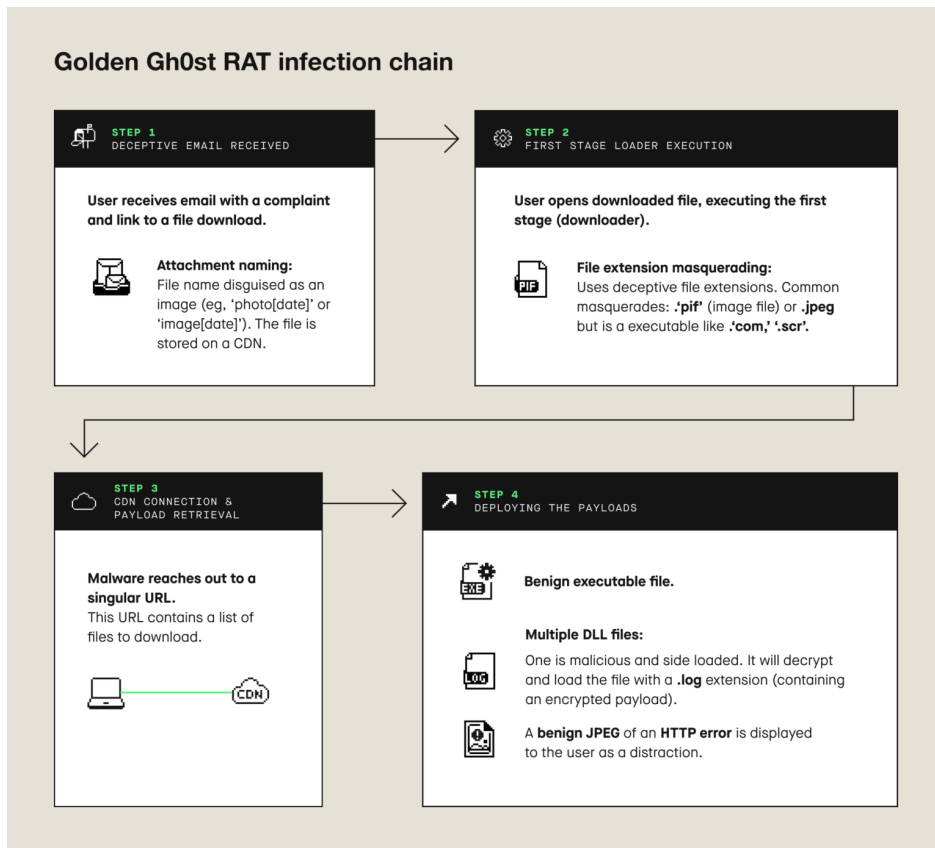
We've also written more about actors using code-signing certificates: how the [Black Basta ransomware gang used 100 or more unique certificates](#), how the [BaoLoader malware used certificates for their long running campaign](#), and the [Rhysida ransomware gang's use of certificates](#).

Since 2024, the [Cert Graveyard](#) has documented 75 unique code-signing certificates used to sign Golden Gh0st Loader. However, in monitoring certificates over time, we observed an abnormal deviation in April 2025: they started using DigiCert certificates, *and* some of the certificates were issued to companies who previously had a valid code-signing certificate from DigiCert.

We reported this activity to DigiCert who investigated, identified the intrusion, and [published the incident report in great detail](#). The incident findings show that a support user's device was infected with malware after running a malicious file received through the support ticketing system. The compromised device allowed the actor to steal initialization codes for DigiCert customers who had open orders to renew their code-signing certificates. These initialization codes allow a customer to activate their physical hardware token for signing, and the sophistication and tactics made us take another look at the malware.

When communicating with DigiCert, we initially tracked the malware as "Zhong Stealer" because we were most familiar with the [AnyRun publication from 2024](#). However, AnyRun's report and name imply that Zhong Stealer is only a stealer, but the DigiCert incident suggested it was more than that. In our investigation and the analysis below, we found that the malware's main functionality is a remote access trojan (RAT).

Infecting a host



The mainstay tactic of this attacker is to send files disguised as screenshots in phishing emails. These files are delivered via a link included in the email, and the email is often picked up by the organization's ticketing system, delivering it directly to the support team, who unknowingly executes the malware when trying to download and view the file. This main tactic has been consistent since AnyRun's 2024 publication, as has the use of content delivery networks (CDN) to host the second stages. We'll briefly examine the malware discussed in AnyRun's publication to understand how it connects with current versions of the malware.

AnyRun's 2025 version

The malware in [AnyRun's report](#) was distributed in a ZIP file and was named “图片_20241224.exe” (translating to “photo_20241224.exe”). When executed by the a victim, it reaches out to get the contents of `hxxps://kkuu.oss-cn-hongkong.aliyuncs[.]com/ss/uu.txt`, which directs the malware to download the three items in the table below.

URL	Filename	SHA256
<code>hxxps://kkuu.oss-cn-hongkong.aliyuncs[.]com/ss/down.exe</code>	down.exe	4eaebd93e23be3427d4c1349d64bef4b5fc455c93aebb9b5b752981e9266488e
<code>hxxps://kkuu.oss-cn-hongkong.aliyuncs[.]com/ss/TASLoginBase.dll</code>	TASLoginBase.dll	1abffe97aafe9916b366da57458a78338598cab9742c2d9e03e4ad0ba11f29bf
<code>hxxps://kkuu.oss-cn-hongkong.aliyuncs[.]com/ss/TASLogin.log</code>	TASLogin.log	dd44dabff536a1aa9b845dd891ad483162d4f28913344c93e5d59f648a186098

The use of multiple files for this second stage is consistent over the duration of the campaigns. Within AnyRun's report, they don't discuss the relationship between these files. The EXE deployed is usually a legitimate application, but the DLL associated with that application is used for DLL sideloading/hijacking; the DLL is loaded by the EXE and is used to decrypt the file labeled as a .LOG.

In this case, the legitimate EXE is TASLogin by Tencent. Here's how it works:

- After pulling down all the files, the first stage executes the TASLogin application which loads the malicious DLL.
- The DLL uses PEB walking to identify Windows API `GetModuleFileNameA`, `CreateFileA`, `VirtualAlloc`, and others.

- The DLL uses these APIs to load the TASLogin.log.
- VirtualAlloc reads the encrypted blob into a read-write-execute (RWX) buffer.
- It then decrypts the buffer byte by byte using the following logic:

```
CIPHER_ADD = 0x77
```

```
CIPHER_XOR = 0x62
```

```
return bytes(((b + CIPHER_ADD) & 0xFF) ^ CIPHER_XOR for b in data)
```

The full decrypted “.log” connected to the AnyRun report [is on VirusTotal](#). The decrypted payload is the initial Golden Gh0st RAT implant discussed by [Qi'anxin](#) and [Xiao Yi](#). We've also provided a decryption script for this [file on GitHub](#).

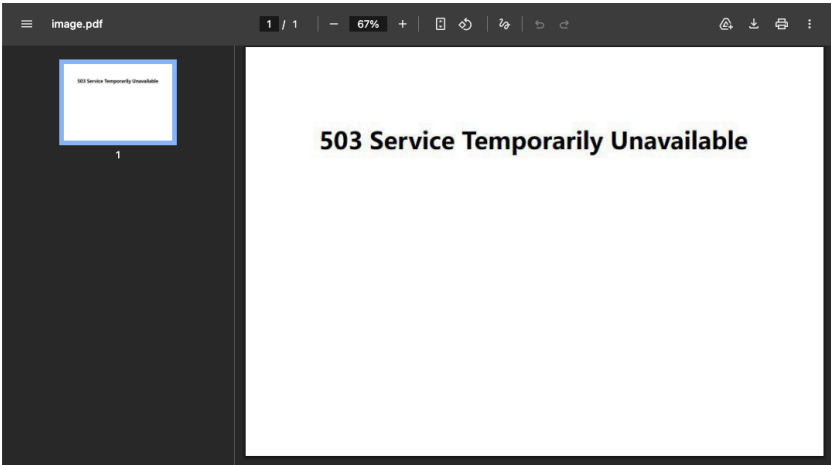
AnyRun's report doesn't cover the decryption of the LOG file or the capability of the decrypted payload. Instead, they focused on credential stealing capabilities observed by the sandbox, without discussing the Gh0st RAT payload and its capabilities.

Hard to teach an old dog new tricks

Over the last two years, we've observed that the tactics used by this malware family have remained mostly the same with remarkably little variation. Variations typically include what legitimate EXEs are used, the inclusion of an additional DLL required by the EXE, and the addition of decoys.

To show the similarity over campaigns, we'll look at a campaign from June 2, 2026. The first stage Golden Gh0st Loader malware reaches out to a text file on the CDN, which contains the URL for each of the other payloads, just as before. The URL this time was `hxxps://storage[.]googleapis[.]com/kiki001/as.txt` and contained the items in the table below.

URL	Purpose	SHA256
<code>hxxps://storage[.]googleapis[.]com/kiki001/updat[.]jexe</code>	Legitimate EXE for sideloading	2b0071007c3f5fa8e949a8de53be03e97901dd505694ca939b575a49e
<code>hxxps://storage[.]googleapis[.]com/kiki001/vcruntime140[.]dll</code>	Legitimate Microsoft runtime	8e08575492175e042f093f325b07a5c14ca71e7c581474838db3d48f5
<code>hxxps://storage[.]googleapis[.]com/kiki001/msvc140[.]dll</code>	Legitimate Microsoft runtime	e4c71980dbb4a1e1a86816687afdaea043b639b531135fc4516fb2429
<code>hxxps://storage[.]googleapis[.]com/kiki001/crashreport[.]dll</code>	Malicious DLL to sideload	27b722c66f69e360c4da106daacf3b9eeaabd20634d7e5eff45a28bd70
<code>hxxps://storage[.]googleapis[.]com/kiki001/updat[.]log</code>	LOG file to be decrypted and loaded into memory	3313f347e83aaf48ea31fb1d49fc37452f48f81d20a1b93009e2e78385
<code>hxxps://storage[.]googleapis[.]com/kiki001/image[.]jpg</code>	Decoy JPEG that shows 503 error	(Not present at time of analysis)
<code>hxxps://storage[.]googleapis[.]com/pkgnew/image[.]pdf</code>	Decoy PDF with JPEG of 503 error	f67de637fca127212dc60b9a02f74e66dbd602b3b9f6f6e4f2b75614c1

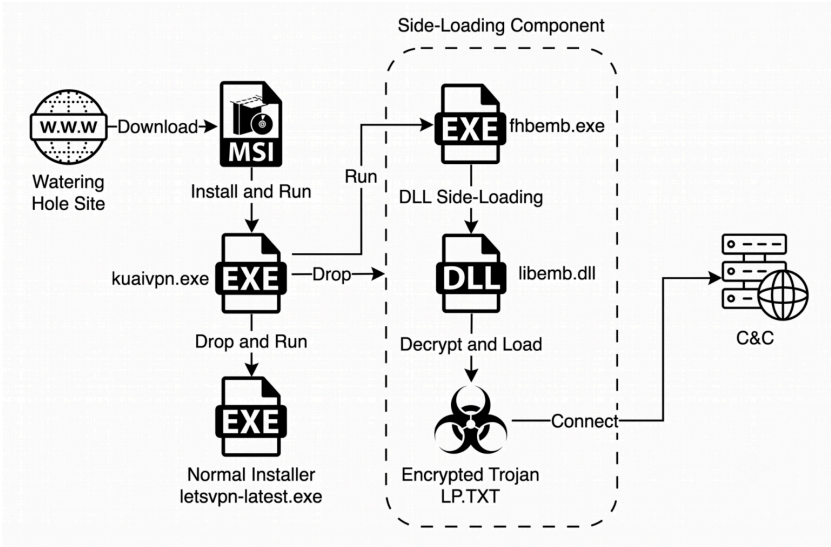


Decoy PDF of a 503 error.

Similar to the 2024 payload, the recent 2026 version uses a legitimate binary to load a DLL, and the DLL loads the update.log and decrypts it. We’ve also provided a decryption tool for this payload. This decryption routine uses a nearly identical decryption routine to the 2024 version, with the only variation: where the 0xFF byte-mask sits. This variation causes no difference in functionality, resulting in an identical result regardless of the byte-mask location.

```
CIPHER_ADD = 0x77
CIPHER_XOR = 0x62
return bytes(((b + CIPHER_ADD) ^ CIPHER_XOR) & 0xFF for b in data)
```

The tactic of using a benign file to sideload a DLL and decrypt a third file is consistent with what’s described by [Qi’anxin in 2024](#) for loading the second stage implant. However, the initial delivery and lure are different: the malware was distributed through a [watering hole](#) website offering a VPN download. The download then retrieves the second stage payload, which uses DLL side-loading to decrypt the second stage implant in a manner consistent with the Golden Gh0st Loader.



Qi’anxin’s depiction of the infection chain associated with GoldenEyeDog’s watering hole lures.

Golden Gh0st RAT 2026

When the infection starts, the malware loads the Golden Gh0st RAT implant.

Capabilities

As with all Gh0st RAT variants, the capability of the malware is handled through plugins and an internal module dispatcher. The following analysis is based on an implant with the SHA256 81e276aaa3eb9b3f595663c316b3c6414cc3dde5e6cc3a82856b7276acabb7de, observed on April 12, 2026.

Handlers will change between samples, and the command IDs are also known to change between samples.

The following table describes the modules and their functionality:

Cmd ID	Handler	Plugin name	Description
0x0991	sub_1001d433	onBootup	Boot-time config/notification
0x0c91	sub_1001c877	fnProxy	SOCKS/TCP proxy tunnel start
0x154c	sub_1001c74b	DllScreenHide	Hide victim's screen/suppress display output
0x1574	sub_1001c773	DllKeyboard	Keylogger module start
0x1579	sub_1001c81d	DllSystem	Process enumeration
0x1587	sub_1001c7eb	DllMsgBox	Display message box on victim screen
0x1589	sub_1001c79b	DllShell	Shell execution mode
0x1593	sub_1001cce0	DllOpenURLHIDE	Open URL in hidden browser window
0x1594	sub_1001cca3	DllOpenURLSHOW	Open URL in hidden browser window
0x1597	sub_1001c944	DllSerStart	Delivers plug32.dll payload to enable an RDP backdoor

In addition to the plugins, the malware contains the following dispatch table and the command functions:

Cmd ID	Handler	Category	Description
0x098e	sub_1001c89f	Config	C2 SOCKS port update—live re-target without reinfection
0x098f	sub_1001c9c3	Credential	Browser credential collection via CDllManager module cmd 0x987 (trigger alt 1)
0x0990	sub_1001c9c3	Credential	Browser credential collection via CDllManager module cmd 0x987 (trigger alt 2)
0x0c8a	sub_1001c9c3	Credential	Browser credential collection via CDllManager module cmd 0x987 (primary trigger)
0x0c8f	sub_1001c92d	Privilege	UAC bypass—re-launches self via ShellExecuteExA(“runas”) then exits
0x0c90	sub_1001c916	Disruption	Kill explorer.exe
0x0c95	sub_1001cd1d	Execution	Kill process by name (from payload); reports kill count to C2
0x0c96	sub_1001d368	Anti-forensic	Recursive file/directory delete (path from payload)
0x0c97	sub_1001cbcb	Persistence	Creates a .BAT script if it doesn't exist. The .BAT script checks for the malware running, if not, it will start it
0x0c98	0x1001cc0c	Unknown	Kills CMD.exe if it is running, CMD.exe is used to run the randomly named .BAT script created by 0x0c97
0x0c99	sub_1001d0a8	Recon	Process listing with full image paths via GetProcessImageFileNameA
0x0c9b	0x1001cc16	Anti-forensic	Uses “RunDll32.exe InetCpl.cpl, ClearMyTracksByProcess 255” to clear Internet Explorer user data (cookies, history, cache, etc)
0x0c9c	0x1001cc20	Credential	Skype —reads Local Storage login tokens from %APPDATA%; kills skype.exe first, then reads files

0x0c9d	0x1001cc2a	Credential	Chrome —reads User Data profile (Login Data) from %LOCALAPPDATA%; kills Chrome.exe first
0x0c9e	0x1001cc34	Credential	Firefox —reads Firefox profile from %APPDATA%; kills firefox.exe first
0x0c9f	0x1001cc3e	Credential	360 Secure Browser (360安全浏览器) —reads Default\apps profile from %APPDATA%; kills 360se.exe first
0x0ca0	0x1001cc48	Credential	360 Speed Browser/360Chrome (360极速浏览器) —reads profile from %LOCALAPPDATA%; kills 360chrome.exe first
0x0ca1	0x1001cc52	Credential	Multi-app sweep —eight profile paths targeting avBackup.dat, FormData3.dat, historyUrl3.db, and history3.db (Tencent QQ/Chinese apps)
0x0ca2	0x1001cc5c	Credential	Tencent QQ Browser (QQ浏览器) —reads Default profile from %LOCALAPPDATA%; kills QQBrowser.exe first
0x1536	sub_1001bee8	C2	Keepalive ACK
0x1537	sub_1001c6fb	File	Loads decryption routine for the plugin32.dll plugin
0x1565	sub_1001c723	Recon	Screenshot capture
0x158a	sub_1001ca5f	Unknown	Remote shutdown/reboot/logoff
0x158b	sub_1001ca02	Anti-forensic	Self-destruct—deletes payload files, deletes “Windos Defende” scheduled task, stops and deletes Windows service persistence, kills running tasks, exits
0x158c	sub_1001cb7a	Anti-forensic	Windows Event Log wiper—clears Application, Security, System logs
0x158f	sub_1001cc66	Execution	Execute command via CreateProcessA on WinSta0\Default
0x1588	sub_1001cb2a	Config	Routing/session group update
0x1590	sub_1001cb2a	Execution	Drop and exec EXE
0x1591	sub_1001c989	Execution	Drop and exec EXE (identical to 0x1590)
0x1592	sub_1001c96c	File	Drop log file
0x1595	sub_1001cae8	Config	C2 host update—live re-target without reinfection

Traffic

The Golden Gh0st RAT uses custom communication methods. The malware uses plain (not TLS/SSL encrypted) WebSocket traffic to communicate with the command and control server. The traffic itself is encrypted with keys that are hardcoded in the implant. In our tracking of Golden Gh0st RAT, we observed that the WebSocket most frequently communicates with the C2 on a low numbered port, primarily 5188 and 5198. This communication method differs from the original [2008 Gh0st RAT, which uses a custom TCP protocol](#).

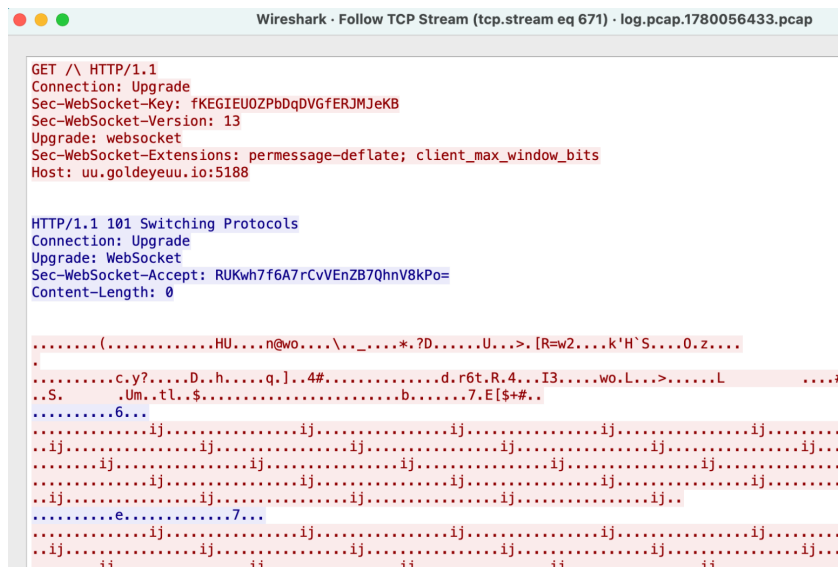
The implants reach out to these command and control servers:

Domain and port	Timeframe
uu.goldeyeuu.io:5188	December 1, 2025–June 9, 2026
wk.goldeyeuu.io:5188	January 18, 2026–June 7, 2026
api.keensie.com:5198	June 6, 2026–current

The Golden Gh0st RAT’s WebSocket uses a custom protocol inside the WebSocket frame, specifying a 12-byte header. These 12 bytes are also used as the heartbeat frames, carrying a size of 0, sent at regular intervals from the client and server to keep the connection alive.

Offset	Size	Field	Function
--------	------	-------	----------

0	4	total_length	Specifies the total byte length of the message (uint32)
4	4	orig_size	Uncompressed size of the plaintext; 0 is used for heartbeats that contain no payload
8	4	session_id	Command/session identifier
12	...	payload	Encrypted and compressed data bytes



The WebSocket setup and heartbeat as seen through a packet capture.

During our analysis, we found that the implants in April/May/June consistently contained the following two hardcoded keys:

1. 'REGISTER_KEY' = '8A913610E905C3DD1F65781EA3B1933471B230F88E1C155616099A03AB0ABC0'
2. 'MODULE_KEY' = '2031A71C399563ADAF1572E10ABB395387EB132208A001C5E140496D7A3E0B26'

The registry key is used to encrypt both client and server communication. The module key is used to decrypt modules/plugins that are pushed via the implant. With these hardcoded keys, we were able to decrypt the network traffic to and from the bot using a custom tool. In preparation for this blog, we confirmed that it can still decrypt traffic as recent as July 6 without any modification. The tool looks for specific server commands to determine how to handle and decrypt the data.

We created a custom tool to decrypt network traffic to and from the bot, and you can find it [here](#).

During our investigation, we monitored an infected host in a simulated domain joined environment using [DeceptionPro](#), which allowed us to monitor an infection for days. With the hardcoded keys and the tool we created, we were able to decrypt the information sent from the victim machine to the command and control server (C2). Our script hasn't been tested against all of the capabilities listed above, but it extracts the following items:

1. Victim fingerprint sent to the C2
2. File listings viewed through the implant
3. Screenshots sent to the C2
4. Payloads sent to the victim device

Days after the infection started, we were able to extract a persistence plugin pushed to the victim host. Unlike the other routine commands, this payload carries an UPX packed executable encrypted using the hardcoded `MODULE_KEY`. The internal name of this module name of this executable is `plugin32.dll`.

plugin32.dll: persistence plugin

The plugin32.dll plugin creates a persistent backdoor. It's passed to the victim host in three chunks, decrypted with the REGISTER_KEY, ZLIB decompressed, assembled, decrypted with the MODULE_KEY, and then loaded into memory. The payload is packed with UPX. The decrypted and unpacked payload has the SHA256 hash

[d1b1938963037aa332591a4c999523a05886d1f62d80e03f0adc22630b8671c4](#), which we've made public alongside this blog.

The plugin creates additional persistence by opening an RDP backdoor in two stages:

1. Create a backdoor account:
 1. The account is created with a username and password hardcoded into the payload.
 2. The new user is added to the administrator group.
2. Modify the `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon` registry key to allow for automatic log-on and suppress credential prompts for the backdoor account.

Defensive opportunities

Suricata rules

The custom communication protocol used by Golden Gh0st RAT gives defenders the opportunity to build detections for this traffic. The following Suricata rules were created in collaboration with Proofpoint, and are now part of Proofpoints ETPro and ETOpen network detection rules.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET MALWARE GoldenGh0stLoader Directory Listing Command From C2"; flow:established,to_client; content:"|00 00 00 00 00 38 15 00 00|"; fast_pattern; offset:4; byte_jump:2,2,from_beginning,little,post_offset 1; isdataat:!2,relative; classtype:trojan-activity; sid:1; rev:1;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET MALWARE GoldenGh0stLoader Websocket Checkin - C2 Response"; flow:established,to_client; content:"|00 00 00 00 00 00 8b 0c|"; fast_pattern; offset:4; byte_jump:2,2,from_beginning,little,post_offset 1; isdataat:!2,relative; classtype:trojan-activity; sid:1; rev:1;)
```

Indicators of compromise (IOCs):

Domains

The following table shows the primary domains used by Golden Gh0st RAT.

Domain and port	Timeframe
uu.goldeyeuu.io:5188	December 1, 2025–June 9, 2026
wk.goldeyeuu.io:5188	January 18, 2026–June 7, 2026
api.keensie.com:5198	June 6, 2026–current

Files

We've provided a CSV on GitHub, which includes 1,926 files. These files use code-signing certificates leveraged by GoldenEyeDog for Golden Gh0st Loader and Golden Gh0st RAT.

https://github.com/expel-io/expel-intel/blob/main/2026/07/GoldenEyeDog-IOC/Golden_Gh0st_IOCs.csv

The signed files are not exclusive to CylindricalCanine. In part two of this blog, we'll dig deeper into what files are signed with these certificates and uncover this threat actor group's relations to other groups.

References

<https://cystack.net/research/malware-linked-apt-q-27>

<https://xz.aliyun.com/news/90793> / <https://www.ctfiot.com/286193.html>

<https://medium.com/@anyrun/zhong-stealer-analysis-new-malware-targeting-fintech-and-cryptocurrency-71d4a3cce42c>

<https://sect.ijj.ad.jp/blog/2026/02/plugin-executed-via-staticplugin/>

<https://bluecyber.hashnode.dev/mustang-panda-x-plugin-analysis-of-the-january-2026-sample-a-multi-layer-execution-chain>

<https://ti.qianxin.com/apt/hacker-detail/5fbb87ef7579b60020f466c7>

[https://mp.weixin.qq.com/s?](https://mp.weixin.qq.com/s?biz=MzI2MDc2MDA4OA==&mid=2247490831&idx=1&sn=54def291b6bd573186734895b7ed03b6&scene=21&poc_token=HAnxLmqj6fzzusOy8Jl)

[biz=MzI2MDc2MDA4OA==&mid=2247490831&idx=1&sn=54def291b6bd573186734895b7ed03b6&scene=21&poc_token=HAnxLmqj6fzzusOy8Jl](https://mp.weixin.qq.com/s?biz=MzI2MDc2MDA4OA==&mid=2247490831&idx=1&sn=54def291b6bd573186734895b7ed03b6&scene=21&poc_token=HAnxLmqj6fzzusOy8Jl)

[zWt](#)

<https://mp.weixin.qq.com/s/gdb8KStXDxY2eOLLbiPFCQ>

<https://asec.ahnlab.com/en/57185/>

Source: <https://expel.com/blog/introducing-cylindricalcanine/>