

# CrystalX: unpacking a Go RAT through three encrypted layers

 [derp.ca/research/crystalx-go-rat](https://derp.ca/research/crystalx-go-rat)

Kirk

May 18, 2026



CrystalX is delivered as a small native Windows loader wrapped around a much larger Go payload. The outer file, `NursultanCracked.exe`, is only 3,068,210 bytes. Almost all of that is resource data.

That layout is the useful starting point. The `.text` section is about 15 KB. The `.rsrc` section is 3,031,040 bytes. Resource 970 contains an encrypted payload, and the stub turns it into a 6.9 MB Go RAT through three transforms: position-dependent XOR, ChaCha20, then raw DEFLATE.

The payload adds another layer of encryption for its runtime strings. Once decrypted, those strings expose the C2 path, builder token, build ID, persistence names, anti-analysis lists, browser and messaging targets, and a command set covering remote desktop, webcam control, file management, keylogging, clipboard access, process execution, and credential theft.

Kaspersky [reported CrystalX \(opens in new tab\)](#) as a March 2026 malware-as-a-service RAT, originally seen as WebCrystal RAT and later rebranded. Their public write-up describes a Go-based RAT with a builder, WebSocket C2, Telegram/YouTube marketing, and payload protection built around zlib compression plus ChaCha20 encryption. That lines up with the mechanics in this sample: the loader unwraps a Go payload with ChaCha20 and raw DEFLATE, then the payload speaks WebSocket to its configured C2.

## Sample overview

---

| Field        | Value                                                            |
|--------------|------------------------------------------------------------------|
| Sample       | NursultanCracked.exe                                             |
| SHA256       | 34b84db8f10d34f711bb242b21bdf662ee489dcd0e9c23b9cc95240d324bb094 |
| Type         | PE32+ x64 GUI, unsigned                                          |
| Size         | 3,068,210 bytes                                                  |
| Compiler     | MSVC                                                             |
| PE timestamp | 2024-11-19T07:18:34Z                                             |
| C2           | wss://crystalxrat[.]net/api/ws                                   |

## The loader: three transforms from resource 970

---

The loader is a compact unpacker. Entrypoint flows through 0x140001190 to the loader state machine at 0x1400026d0. It loads RCDATA resource 970 via `FindResourceW(0, 970, RT_RCDATA)`, copies the buffer from offset +4 with `rep movsd` (0xb2188 DWORDs into a 0x2c8620-byte buffer), and runs three sequential transforms.

### Pass one: position-dependent XOR

---

```
for each byte offset:
    data[offset] ^= ((offset & 0xff) % 0x5f) + 0x20
```

The mask depends on the low byte of the offset. 0x5f is 95, so the formula cycles through 95 unique XOR bytes starting at 0x20 (space).

### Pass two: ChaCha20

---

```
Key:    d01526bdaad75c24f94b80a6fde12b958078fa82beb4741e1ccdd8eb15564470
Nonce:  598a7eeda372bc6d9992e03c
Initial counter: 0
```

The implementation at 0x14000328c-0x14000355b uses a ChaCha20 quarterround routine at 0x140004550 with rotation constants 16/12/8/7 and the standard expand 32-byte k constant. The key is embedded at 0x1400067c0, nonce at 0x140006690.

## Pass three: raw DEFLATE

---

The ChaCha20 output feeds into a raw DEFLATE inflater at 0x140003a30 that handles both fixed and dynamic Huffman blocks. The inflated buffer starts with an MZ header.

## Manual PE mapping

---

The loader does full PE loading from scratch. It copies headers and sections, applies relocations, resolves imports through an embedded import descriptor, sets section memory protections, runs TLS callbacks if present, then calls the payload entrypoint. The unpacked payload matches the mapped memory region from the behavioural memory dump at base 0x140000000, which ties the static unpacking path back to the runtime payload.

## The Go payload

---

The decoded PE is 6,908,416 bytes. PE32+ x64 GUI. SHA256  
a9340c46243f5d2b00e30ea649bd14fc146ebbb42e43dbe45f5ee0cc9fc9227a.

The Go runtime is present but the build info is stripped. `go version -m` returns unknown. GoReSym parsed the pcintab metadata as Go 1.20-era metadata, but it cannot recover the original package paths. The type names have been replaced with random 8-12 character strings.

The binary has roughly 3,000 obfuscated type names. Examples from the GoReSym output: EtCmeRH4, GKBqIpMv, YhleiAVa8oj, Sd2mofbx. The original package structure is gone. Recovering function semantics requires following xrefs from known library functions such as `crypto/aes`, `crypto/rsa`, and `crypto/cipher.XORKeyStream` into the obfuscated user code that calls them.

## The string obfuscation layer

---

The Go binary encrypts every operational string at rest. The string decryptor at 0x1402da180 is called at runtime whenever the RAT needs a string. The flow:

1. `main.(*dSRZL80U0z).Execute` calls the decryptor.
2. `RMdVNutIO.(*Uxm_0m).DecodeString` at 0x140127b20 Base64-decodes the blob.
3. `DZs79I.IS5J3lcHT` at 0x140123680 validates the AES key length (16, 24, or 32 bytes) and constructs an AES block state.
4. `FlzRlavBxpyH.JVIiHaAiMgQ` at 0x140121cc0 sets up GCM mode: nonce size 12 bytes, tag size 16 bytes.

The static key lives in a Go string tuple at 0x140656b30:

Hk4f0CLbqKFbbAxwyAcFKUKXK4iqVaMD

Encrypted strings are stored as Base64. Decoded, the first 12 bytes are the AES-GCM nonce. The remainder is the ciphertext plus 16-byte authentication tag.

Decrypting the string table recovered 265 plaintexts.

## What the strings contained

---

- **C2 endpoints:** crystalxrat[.]net, /api/ws, wss
- **Authentication:** X-Builder-Token, zenc0rn
- **Build tracking:** YBFZUW1U32T
- **Persistence names:** SecurityHealthSystray.exe, Windows Security Health Service, NvContainerTask\_YBFZUW1U32, Global\WinSecMutex\_YBFZUW1U32
- **Anti-analysis:** VM MAC OUIs, sandbox hostnames and usernames, analysis tool process names, proxy/MITM tool names
- **Browser paths:** Chrome, Edge, Opera, Yandex, Login Data, Cookies, Web Data, Local Storage, leveldb
- **Messaging paths:** Discord (discord, discordcanary, discordptb), Telegram Desktop, AyuGram, 64Gram, Kotatogram
- **Gaming paths:** Steam (loginusers.vdf, ssfn), Roblox (ROBLOSECURITY)
- **System API names:** CreateProcessW, VirtualAllocEx, WriteProcessMemory, NtTraceEvent, AmsiScanBuffer, EtwEventWrite

The build ID YBFZUW1U32T seeds all persistence artifact names. The mutex is Global\WinSecMutex\_YBFZUW1U32. The scheduled task is NvContainerTask\_YBFZUW1U32. The lock file lands as %ProgramData%\GoogleUpdate\YBFZUW1U32T.lock.

That AES-GCM path is separate from the loader's ChaCha20 pass. GoReSym also surfaced AES-CTR, RC4, and ChaCha20 XORKeyStream wrappers inside the payload. The confirmed string decryptor for this sample is AES-GCM.

## C2 protocol

---

The C2 orchestrator at 0x14031b940 decrypts the five required endpoint components from the string table at runtime, then calls Qeomb2.NGG9X0j.Dial and .Upgrade for WebSocket handshake over TLS.

```
Endpoint: wss://crystalxrat[.]net/api/ws
Header:   X-Builder-Token: zenc0rn
Build:    YBFZUW1U32T
```

The WebSocket frame writer at 0x1402cf860 uses standard framing: FIN/opcode byte, 7/16/64-bit payload length encoding, mask bit with 4-byte mask key. No additional post-TLS crypto. After WebSocket decode, the payload body is JSON.

## Inbound message dispatcher

---

The inbound dispatcher at 0x14031d120 reads incoming WebSocket frames, handles control opcodes (close, ping, pong), parses the JSON body, and routes on a type field.

| type                       | Handler                                    |
|----------------------------|--------------------------------------------|
| command                    | Routes to direct command dispatcher        |
| rd_start / rd_stop         | Remote desktop stream control              |
| rd_input                   | Remote desktop mouse/keyboard input events |
| rd_list_monitors           | Enumerate connected monitors               |
| rd_block_mouse             | Block mouse input                          |
| rd_block_keyboard          | Block keyboard input                       |
| rd_block_display           | Blank or lock the display                  |
| webcam_list                | Enumerate webcam devices                   |
| webcam_start / webcam_stop | Webcam streaming control                   |
| chat_to_victim             | Display a message box to the victim        |

Message parameters include monitor\_index, quality, device\_index, message, and block.

## Direct command dispatcher

---

The direct command dispatcher at 0x1402feaa0 compares inbound command strings against a table of 40+ known commands and dispatches to handler closures:

| Category        | Commands                                                                                                                                                                                     |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Shell execution | cmd:, bg:, start, .exe, .bat, .cmd                                                                                                                                                           |
| System          | reboot, bsod, voltage_drop, lock_desktop, close_all_windows                                                                                                                                  |
| Display         | shake_on/off, block_input_on/off, rotate_* (0/90/180/270), taskbar_hide/show, hide_desktop_icons/show_desktop_icons, hide_cursor/show_cursor, invert_colors_on/off, monitor_on/off, visible: |

| Category          | Commands                                                                                                                                                          |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Admin toggles     | <code>disable_taskmgr/enable_taskmgr, disable_cmd/enable_cmd</code>                                                                                               |
| File manager      | <code>fm:drives, fm:ls:, fm:del:, fm:rename:</code>                                                                                                               |
| Data theft        | <code>clipboard:get, clipboard:set:, clipboard:start, clipboard:stop, keylogger:start, keylogger:stop, software:list, software:uninstall:, steal&gt;manual</code> |
| UI tricks         | <code>`msgbox</code>                                                                                                                                              |
| Network downloads | <code>bg: downloads a background image to bg_image.jpg and applies it</code>                                                                                      |

The `cmd:` prefix runs commands through `CreateProcessW`. The `bg:` prefix downloads and applies a remote image as wallpaper. The `/remote/` path fragment is used by URL-related helpers for downloading and executing files from remote URLs.

## Persistence

The sample copies itself to

`%LOCALAPPDATA%\Microsoft\DeviceMetadataStore\SecurityHealthSystray.exe`. The installed copy is byte-for-byte identical to the original and has the same SHA256. The build ID appears in every persistence name.

| Mechanism      | Detail                                                                                                                                                                        |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Scheduled task | <code>NvContainerTask_YBFZUW1U32</code> runs <code>SecurityHealthSystray.exe</code> worker on logon, highest privileges                                                       |
| Startup folder | <code>Windows Security Health Service.lnk</code>                                                                                                                              |
| WMI filter     | <code>Windows Security Health Service_Filter</code> monitors <code>__InstanceModificationEvent</code> on <code>Win32_PerfFormattedData_Perf0S_System</code> every 300 seconds |
| WMI consumer   | <code>Windows Security Health Service_Consumer (CommandLineEventConsumer)</code>                                                                                              |

| Mechanism | Detail                        |
|-----------|-------------------------------|
| Mutex     | Global\WinSecMutex_YBFZUW1U32 |

Defender exclusions are added for C:\Windows\System32\svchost.exe and the entire System32 directory. The TaskScheduler Operational and Services Diagnostic event logs are disabled via wevtutil. Inbound firewall notification is turned off. Inbound and outbound firewall rules named System Network Service are created.

## Anti-analysis

The payload runs a multi-category environment check before any C2 activity. The decrypted string set contains the full detection lists.

| Category              | Checks                                                                                                           |
|-----------------------|------------------------------------------------------------------------------------------------------------------|
| MAC prefixes          | 00:05:69, 00:0c:29, 00:50:56, 08:00:27, 52:54:00 (VMware, VirtualBox, QEMU)                                      |
| Hostname blacklist    | sandbox, malware, virus, analysis, cuckoo, vmware, vbox, qemu                                                    |
| Username blacklist    | sandbox, virus, malware, maltest, test, john, user, currentuser                                                  |
| Registry VM artifacts | VMware Tools, VirtualBox Guest Additions, VBoxGuest/Mouse/Service/SF/Video, vmci, vmhgfs, VMTools, VMMEMCTL, Xen |
| VM process detection  | VBoxService, VBoxTray, vmttoolsd, VMwareTray, VMwareUser, prl_tools, qemu-ga, xenservice, vboxservice            |
| Analysis tools        | wireshark, procmon, processhacker, x64dbg, ollydbg, ida, windbg, dnSpy, fiddler, burp, Sysmon                    |
| Proxy/MITM tools      | charles, burp, mitmproxy, portswigger, zap proxy, Proxifier, HTTPDebugger                                        |

If detection triggers, the sample runs PowerShell to disable Defender (Stop-Service WinDefend, disable Defender scheduled tasks, set registry tamper flags), disable Sysmon, purge all event logs with wevtutil el | ForEach-Object { wevtutil cl \$\_ }, and apply restrictive ACLs on its installation directory.

## Network behavior

---

Two behavioural runs connected to `crystalxrat[.]net:443` over TLS. The resolved addresses sit behind Cloudflare AS13335, making them CDN edge observations.

The first PCAP shows `crystalxrat[.]net` in the TLS SNI with 56,862 bytes transmitted and 8,701 bytes received. The second PCAP shows similar volumes: 56,640 bytes transmitted and 8,639 bytes received. The CrystalX TLS session client random (5a3a660ff209398a8a81bd8dc6ca8b9880c0456879456a4f930531eba0557a9d) is absent from the keylog, so the captured WebSocket payload bodies remain opaque. The keylog contains 12 CLIENT\_RANDOM entries, all from a different session prefix.

The domain is young. WHOIS shows `crystalxrat[.]net` registered through Tucows on 2026-04-14 at 10:07:57Z, updated at 10:11:21Z, and pointed at `braden.ns.cloudflare.com` and `teresa.ns.cloudflare.com`. SecurityTrails only lists one subdomain, `www`.

Certificate history is also narrow. CertSpotter shows two public certificates for `crystalxrat[.]net` / `*.crystalxrat[.]net` on 2026-04-14: one Let's Encrypt E8 certificate and one Sectigo DV E36 certificate. The live TLS certificate is the Let's Encrypt leaf, valid from 2026-04-14 to 2026-07-13, with SHA1 fingerprint 087d077990ecbc7ac2b09e261df80ffc49475a61. SSLBL has no entry for that certificate.

ThreatFox lists `crystalxrat[.]net` as a `botnet_cc` domain tagged CrystalX and RAT, reported by abuse.ch. OTX has public botnet C2 pulses for the domain and URL observations for `hxxp://crystalxrat[.]net`, `hxxps://crystalxrat[.]net`, and `hxxps://crystalxrat[.]net/login/`, all returning HTTP 403. URLhaus has no host entry.

Public Triage results show a small visible CrystalX cluster: two May 11 submissions of this NursultanCracked.exe hash, two May 1 samples using `crystalxrat[.]net`, and one May 1 sample using `crystalxrat[.]top`. The `.top` domain was registered through Spaceship on 2026-03-12, used the same Cloudflare nameservers, and had apex/wildcard certificates issued the same day; RDAP currently shows serverHold. Public URL submissions on Apr 8 also touched `webcrystal[.]lol` and `webcrystal[.]sbs`, echoing the WebCrystal branding Kaspersky described.

## Detection

---

## IOC summary

---

## Hashes

---

| Artifact                      | SHA256                                                           |
|-------------------------------|------------------------------------------------------------------|
| NursultanCracked.exe (loader) | 34b84db8f10d34f711bb242b21bdf662ee489dcd0e9c23b9cc95240d324bb094 |
| Unpacked Go payload           | a9340c46243f5d2b00e30ea649bd14fc146ebbb42e43dbe45f5ee0cc9fc9227a |
| RCDATA 970 (encrypted)        | 8a6f8ef99384152df63a39b6ba9f08f0a1e9cc33b14319e8a1b184beb4a06cf7 |
| Memory dump (mapped)          | 2497e0aa88af681872194966bfc2bd67013ea75c96f4b5717abe4a4f43e69394 |

## Network

---

| Indicator      | Value                                                       |
|----------------|-------------------------------------------------------------|
| C2             | crystalxrat[.]net:443                                       |
| Related C2     | crystalxrat[.]top                                           |
| WebSocket path | wss://crystalxrat[.]net/api/ws                              |
| Nameservers    | braden.ns.cloudflare.com, teresa.ns.cloudflare.com          |
| Registrar      | Tucows Domains Inc.                                         |
| Created        | 2026-04-14T10:07:57Z                                        |
| CT history     | Let's Encrypt E8 and Sectigo DV E36 certs for apex/wildcard |
| ThreatFox      | botnet_cc, tags: CrystalX, RAT                              |
| Builder token  | X-Builder-Token: zenc0rn                                    |
| Build ID       | YBFZUW1U32T                                                 |

## Host

| Indicator        | Value                                                                  |
|------------------|------------------------------------------------------------------------|
| Persistence path | %LOCALAPPDATA%\Microsoft\DeviceMetadataStore\SecurityHealthSystray.exe |
| Scheduled task   | NvContainerTask_YBFZUW1U32                                             |
| Startup shortcut | Windows Security Health Service.lnk                                    |
| WMI filter       | Windows Security Health Service_Filter                                 |
| WMI consumer     | Windows Security Health Service_Consumer                               |
| Firewall rules   | System Network Service, System Network Service Out                     |
| Mutex            | Global\WinSecMutex_YBFZUW1U32                                          |
| Lock file        | %ProgramData%\GoogleUpdate\YBFZUW1U32T.lock                            |

## Static keys

| Key                                                              | Use                    |
|------------------------------------------------------------------|------------------------|
| Hk4f0CLbqKFbbAxwyAcFKUKXK4iqVaMD                                 | AES-GCM string decrypt |
| d01526bdaad75c24f94b80a6fde12b958078fa82beb4741e1ccdd8eb15564470 | ChaCha20 loader key    |
| 598a7eeda372bc6d9992e03c                                         | ChaCha20 loader nonce  |

CrystalX makes you earn the view. The loader burns three transforms to hide a Go payload, the payload hides its strings behind AES-GCM, and the useful shape only appears after both layers are peeled back: a builder-tagged WebSocket RAT with durable host artifacts, a loud command surface, and C2 infrastructure that already shows a small public trail from WebCrystal-era domains into the current CrystalX naming.

Kirk

I like the internet. Want to get in touch?