TERMINULL

==========

VOLUME 1

2026-02-13 — 12 articles

The hacker e-zine for the modern underground

TABLE OF CONTENTS — VOLUME 1

00. Welcome to termin… Leland Flynn (@c0mmrade) EDITORIAL

01. React2Shell: Anatomy… Kai Yamamoto (@fl1ght) WRITEUP

02. The $1.5 Billion … Nora Ekström (@chainwatch) WRITEUP

03. EtherHiding: When th… Mara Okonkwo (@bl0ckd3ad) GUIDE

04. AirBorne: Zero-Cli… Dmitri Volkov (@rf_ghost) WRITEUP

05. Hunting the Giant C… Leland Flynn (@c0mmrade) WRITEUP

06. Ghost Tap: NFC Relay F… Reza Hajian (@tapjack) GUIDE

07. AI Breaks Crypto: Dee… Elena Kostrova (@d1ffeq) GUIDE

08. XBOW: The AI That B… Marcus Oduya (@autoexploit) TOOL

09. Security News Roun… The Editor (@r00t) SECURITY-NEWS

11. xibalba tainted ASCII-ART

12. shuratos heavenly BBS Savage ASCII-ART

WELCOME TO TERMINULL

by Leland Flynn (@c0mmrade) — 2026-02-13

tags: introduction, meta, editorial

WELCOME TO TERMINULL

The screen flickers to life. Another terminal, another connection. But this one is different.

You've connected to **terminull** -- the hacker e-zine with a classic style and modern content. Built for the terminal and the web alike, by people who still believe in the beauty of the baud. No tracking pixels. No newsletter popups. No sponsored content. Just pure, uncut signal.

> **[HACK]** terminull is a space for technical depth, original research, and the hacker ethos that built the internet. No SEO optimization. No engagement metrics. Just the work.

whomai

Actual Self-Portrait of the Author



My name is Leland but, you can call me `c0mmrade` if you'd like. I'm a Security Researcher, Hacker, and semi-professional weirdo. I suppose you could call me a *"Technologist"* if you wanted to get reductive with your labeling. I've worked in the tech space in varying capacities for over 24 years as of this publishing. Technology, Philosophy, Culture, and Justice are some of my deepest passions. *Radical Honesty* is an imperative for me; an undeniable signal.

I see beauty in the signal. And I intend to maintain it in whatever way I can.

WHY NOW?

We are digital citizens of the world. We keep us safe. We care about our world. We will make it better. We reject the global panopticon. We will fight it. We believe in truth. We will maintain the signal.

2025 was a year that demanded our attention. And 2026 has brought with it salvo after salvo of truth to power. Though many seem hell-bent on skewing that truth, many more are dedicated to maintaining the signal.

What better place than here? What better time than now?

This is the world we operate in now. It deserves a publication that takes it seriously -- and takes itself less seriously.

@mqudsi[1] - We see you, we hear you, and we thank you.

WHAT'S IN THIS ISSUE

A single deserialization flaw in React Server Components gave every nation-state APT on the planet unauthenticated RCE on half the web. North Korea stole $1.5 billion from a single exchange in an afternoon -- then hid their C2 infrastructure on the blockchain where nobody can take it down. Apple's AirPlay protocol turned out to be a wormable zero-click attack surface hiding in every living room. An AI agent climbed to #1 on HackerOne's global leaderboard, outperforming thousands of human hackers in a fraction of the time.

And somewhere in all of that, a ransomware gang left their credentials in a directory listing.

Welcome to Volume 01 of **terminull**.

* **React2Shell** -- anatomy of a CVSS 10.0 that broke the web's favorite framework
* **Lazarus vs. Bybit** -- inside the $1.5 billion crypto heist and the supply chain attack that made it possible

* **EtherHiding** -- when the blockchain becomes your C2 and nobody can pull the plug
* **AirBorne** -- 23 vulnerabilities in AirPlay, zero-click wormable RCE in your living room
* **Hunting the Giant Centipede** -- original Rhysida ransomware research, from IR to FBI coordination
* **Hunting the Giant Centipede: Phase 2** -- infrastructure genealogy, malware forensics, and attribution
* **Ghost Tap** -- NFC relay fraud at scale, from research tool to MaaS platform
* **AI Breaks Crypto** -- deep learning side-channel attacks on AES and post-quantum schemes
* **XBOW** -- the AI that beat every hacker on HackerOne
* **Security News Roundup** -- everything else that mattered in 2025

Each article is written for people who want to understand the *how*, not just the *what*. Code blocks have language tags. Exploit chains have diagrams. IOCs are included where appropriate.

HOW TO NAVIGATE

If you're reading this in a terminal -- as intended -- you have options:

```
guest@terminull:~$ help        # show available commands
guest@terminull:~$ read 01     # read article #01
guest@terminull:~$ ls          # list articles in current
volume
guest@terminull:~$ search rce  # search across all content
guest@terminull:~$ crt         # toggle CRT scanline effec
t
```

Keyboard shortcuts work too: `j`/`k` to move through items, number keys for direct selection, `Enter` to read, `?` for the full help overlay.

Over SSH? Connect with `ssh terminull.dev -p 2222` for the full BBS experience -- Bubble Tea TUI, Glamour-rendered markdown, the works. Thanks @charmcli[2] we truly love your work!

A NOTE ON ETHICS

Every technique in this zine is documented for **education and defense**. We practice *responsible disclosure* where possible. We do not coweer to bullies. We notify victims when we find them exposed. Article #5 in this issue describes an investigation where we proactively warned 14 organizations and *attempted* to work with the FBI to stop those responsible -- that's the standard.

> **[WARN]** Never use techniques described here against systems you don't own or have explicit, written authorization to test. Never use you knowledge or skills to harm an innocent. The line between research and evil is drawn by consent.

GET INVOLVED

terminull is an open project. If you have original research, a war story worth telling, art worth displaying, or a tool worth sharing -- we want to hear from you. The best zines were always community-driven. The source is on GitHub. The content is markdown. The bar for contribution is having something worth saying.

The cursor blinks. The prompt awaits.

Welcome aboard.

`-- c0mmrade`

LINKS

[1] https://x.com/mqudsi
[2] https://x.com/charmcli

REACT2SHELL: ANATOMY OF A 10.0

by Kai Yamamoto (@fl1ght) — 2026-02-13

tags: rce, deserialization, react, nextjs, cve, supply-chain

REACT2SHELL: ANATOMY OF A 10.0

On December 3, 2025, the React team published a security advisory that stopped the JavaScript ecosystem cold. CVE-2025-55182 -- a CVSS 10.0 unauthenticated remote code execution vulnerability in React Server Components -- was being actively exploited within hours of disclosure.

```
CVSS: 10.0 (Critical)
Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
Affected: react-server-dom-webpack, react-server-dom-turbo
pack, react-server-dom-parcel
Fixed: React 19.1.0
```

A single crafted POST request. Full server takeover. No authentication required.

They called it **React2Shell**.

React Server Components and the Flight Protocol

To understand how this happened, you need to understand React Server Components (RSC). Introduced with React 18 and matured in React 19, RSCs let you render components on the server and stream them to the client. The wire protocol that makes this work is called **Flight**.

Flight serializes the component tree -- props, references, and Server Function arguments -- into a binary-ish format that gets sent between client and server. When a user interacts with a form or calls a Server Action, the client sends a Flight-encoded payload back to the server, which deserializes it and executes the referenced function.

```
Client → Server: POST /_rsc (Flight-encoded payload)
Server: Deserialize → Resolve references → Execute Server Function
```

The problem? The deserialization was too powerful.

The Flaw

Security researcher Lachlan Davidson discovered that the Flight protocol's deserialization logic allowed attacker-controlled inputs to traverse the prototype chain and access the `Function` constructor. This meant an attacker could construct arbitrary JavaScript functions on the server side through nothing more than a crafted HTTP request.

> **[HACK]** The exploit works through prototype chain traversal during deserialization. The server processes RSC payloads without sufficient validation, and under specific conditions, a malformed payload causes the server to reach the Function constructor -- enabling arbitrary code execution.

The attack surface was enormous. Any application using React Server Components with Server Functions was vulnerable. But it was even worse than that -- Rapid7 confirmed that **applications without explicitly defined Server Functions were still vulnerable** if they used RSC at all. Standard Next.js production builds include RSC endpoints that can be reached without special routing or configuration.

Wiz reported that **39% of cloud environments** contained instances of Next.js or React in versions vulnerable to CVE-2025-55182.

The Exploit Chain

The exploitation is devastatingly simple:

1. **1.** Attacker identifies a target running React 19.x with Server Components (trivially fingerprintable)
2. **2.** Attacker sends a crafted POST request to any RSC endpoint
3. **3.** The Flight deserializer processes the malicious payload
4. **4.** Prototype chain traversal reaches the Function constructor
5. **5.** Attacker-supplied code executes in the server's Node.js process
6. **6.** Full RCE achieved -- reverse shell, data exfiltration, lateral movement

The exploit had a **near-100% success rate** against vulnerable targets. No memory corruption, no race conditions, no unreliable heap spraying -- just clean, deterministic deserialization-to-execution.

> **[WARN]** We are not publishing exploit code. The vulnerability is well-documented by Wiz, Rapid7, and others. If you need to test your systems, use the scanner at `github.com/Security-Phoenix-demo/react2shell-scanner`.

In the Wild

The timeline was brutal:

Date	Event
Nov 29, 2025	Lachlan Davidson discovers the vulnerability, reports to React team
Nov 30	Meta security confirms the flaw
Dec 1	Fix created, coordinated rollout to hosting providers begins
Dec 3	Fix published to npm, CVE-2025-55182 publicly disclosed
Dec 4	Davidson publishes PoC on GitHub
Dec 5	Exploitation detected in the wild by Amazon, GreyNoise, Rapid7

Within hours of disclosure, multiple China-nexus threat groups -- including **Earth Lamia** and **Jackpot Panda** -- were actively exploiting the vulnerability. Amazon's threat intelligence team observed active exploitation attempts almost immediately.

DPRK's **UNC5342** followed quickly, deploying React2Shell as an initial access vector for their EtherRAT campaign (see Article #3 in this issue).

By late December, GreyNoise had recorded over **8.1 million attack sessions**, with daily volumes stabilizing between 300,000-400,000 attempts. Their observation grid detected **8,163 unique source IPs** across 1,071 ASNs spanning 101 countries.

BleepingComputer reported that attackers had **breached over 30 organizations**, with 77,000 internet-exposed IP addresses remaining vulnerable.

Post-exploitation activity followed a familiar playbook: reconnaissance commands confirming RCE (PowerShell math operations as a canary), then Cobalt Strike beacon deployment for persistent access.

Next.js: A Companion CVE

The vulnerability also received a Next.js-specific tracking as **CVE-2025-66478**, though this was later rejected as a duplicate -- the root cause was in React's Flight deserializer, not in Next.js itself. Vercel published a one-command fix: `npx fix-react2shell-next` to automatically bump affected dependencies.

Detection and Mitigation

Immediate actions:

```bash
# Check if you're running vulnerable React versions
npm ls react-server-dom-webpack react-server-dom-turbopack
react-server-dom-parcel

# Fix Next.js applications
npx fix-react2shell-next

# Or manually update React to 19.1.0+
npm install react@latest react-dom@latest
```

WAF signatures to detect exploitation attempts should look for:

* Malformed POST requests to RSC endpoints (`/_rsc`, Server Action paths)
* Flight protocol payloads containing prototype chain traversal patterns
* Unusual `Content-Type` headers on Server Function endpoints

Network indicators:

* POST requests to `/_rsc` endpoints from external sources
* Outbound connections to known Cobalt Strike infrastructure post-exploitation
* PowerShell execution from Node.js processes

The Bigger Picture

React2Shell wasn't just another CVE. It was a structural failure.

The Flight protocol was designed for performance, not security. Deserialization was trusted implicitly. The attack surface was enabled by default. And when the vulnerability dropped, the entire JavaScript ecosystem -- from startups to Fortune 500 -- was exposed simultaneously.

> **[INFO]** Comparisons to Log4Shell (CVE-2021-44228) are apt. Both were unauthenticated RCE in ubiquitous software. Both exploited deserialization trust boundaries. Both had near-instant exploitation by nation-state actors. The key difference: React2Shell had even broader reach because every Next.js App Router deployment was a target.

The fix was two lines of code. The damage was measured in months.

React2Shell is a reminder that "move fast" culture has consequences. When your serialization protocol becomes the most deployed attack surface on the internet, it better be hardened like one.

Patch your servers. Then read Article #3 to see what DPRK did with the access.

`-- fl1ght`

THE $1.5 BILLION HEIST: LAZARUS VS. BYBIT

by Nora Ekström (@chainwatch) — 2026-02-13

tags: dprk, lazarus, cryptocurrency, supply-chain, incident-response

THE $1.5 BILLION HEIST: LAZARUS VS. BYBIT

On February 21, 2025, at approximately 12:30 PM UTC, Bybit detected unauthorized activity in one of its Ethereum cold wallets. Within minutes, 401,347 ETH -- worth over **$1.5 billion** -- had been drained to an unknown address.

It was the largest cryptocurrency theft in history. And it started with a Docker project on a developer's laptop.

The Timeline

The attack didn't begin on February 21. It began 17 days earlier.

Date	Event
Feb 2	Domain `getstockprice[.]com` registered via Namecheap
Feb 4	Safe{Wallet} Developer1's macOS workstation compromised
Feb 5	Attacker begins accessing Safe{Wallet} AWS infrastructure
Feb 19	Malicious JavaScript injected into Safe{Wallet} S3 bucket
Feb 21, 12:30 UTC	Bybit initiates routine cold→warm wallet transfer
Feb 21, 12:31 UTC	Malicious JS intercepts transaction, redirects 401,347 ETH
Feb 21, 12:33 UTC	Attacker removes malicious JS from S3 bucket
Feb 26	FBI attributes attack to DPRK's TraderTraitor (Lazarus)

Two minutes. That's how long the malicious code was live in production before the attackers cleaned up after themselves.

The Supply Chain

Bybit didn't get hacked directly. **Safe{Wallet}** did -- the multisig wallet platform Bybit used to manage its cold storage.

Stage 1: Social Engineering

The attack began with a social engineering campaign targeting a Safe{Wallet} developer, referred to in forensic reports as "Developer1." The approach likely came through LinkedIn, Telegram, or Discord -- the standard playbook for DPRK's "Contagious Interview" campaigns.

Developer1 downloaded what appeared to be a legitimate crypto-themed application: a Docker project called `MC-Based-Stock-Invest-Simulator-main`, found in their `~/Downloads` folder. The project phoned home to `getstockprice[.]com` -- a domain registered just two days earlier.

> **[HACK]** The malicious Docker project exploited a remote code execution vulnerability in the PyYAML library, giving the attackers code execution on the developer's macOS workstation.

Stage 2: AWS Session Hijack

With control of Developer1's machine, the attackers extracted their active AWS session token. Safe{Wallet}'s configuration required MFA re-authentication every 12 hours for STS sessions, but the attackers didn't need to bypass MFA -- they simply hijacked an already-authenticated session.

Starting February 5, the attackers used the stolen token to access Safe{Wallet}'s AWS infrastructure, probing S3 buckets and CloudFront configurations.

Stage 3: JavaScript Injection

On February 19, the attackers modified JavaScript resources hosted on Safe{Wallet}'s AWS S3 bucket. The injected code was surgical -- it only activated for transactions from Bybit's specific cold wallet address (`0x1Db92e2EeBC8E0c075a02BeA49a2935BcD2dFCF4`).

When Bybit's signers opened Safe{Wallet}'s UI to approve the cold-to-warm transfer on February 21, they saw what appeared to be a legitimate transaction. The malicious JavaScript had altered the transaction's destination while keeping the UI display unchanged.

Stage 4: The Drain

The Bybit signers approved what they believed was a routine transfer. Instead, the transaction executed a `delegatecall` to a malicious contract, replacing the wallet's legitimate implementation and draining all 401,347 ETH to attacker-controlled address `0x47666fab8bd0ac7003bce3f5c3585383f09486e2`.

```
Wallet: 0x1Db92e...dFCF4 (Bybit Cold)
  └→ ExecTransaction (delegatecall)
     └→ Implementation swap → attacker contract
        └→ Drain: 401,347 ETH → 0x47666f...86e2
```

Why Multisig Wasn't Enough

Bybit used a Gnosis Safe multisig wallet -- the industry standard. Multiple signers had to approve every transaction. So how did a single malicious JavaScript injection bypass multisig?

Because the attack didn't target the *crypto*. It targeted the *UI*.

Every signer saw the same compromised Safe{Wallet} interface. Every signer approved what looked like a legitimate transaction. The multisig worked exactly as designed -- it just signed the wrong thing.

> **[WARN]** NCC Group's post-mortem was blunt: Bybit used a smart contract with a much larger attack surface than necessary. A purpose-built contract limited to native transfers and ERC-20 operations could have used human-readable EIP-712 signatures, making the malicious destination visible to signers. The `delegatecall` mechanism -- which allowed arbitrary contract replacement -- was an unnecessary feature that became the critical vulnerability.

Following the Money

Post-heist, the attackers moved fast. Within hours, the stolen ETH was being converted to Bitcoin and dispersed across thousands of blockchain addresses -- the standard Lazarus laundering playbook.

The FBI released a Public Service Announcement on February 26, attributing the theft to "TraderTraitor" -- their designation for the Lazarus Group / UNC4899 / Jade Sleet. They published 51 Ethereum addresses used in the initial laundering phase and called on exchanges, bridges, and DeFi services to block derived transactions.

Bybit launched a **$140 million bounty program**, offering up to 10% of recovered funds to anyone who could help trace and freeze the stolen assets. Blockchain analytics firms -- Chainalysis, Elliptic, TRM Labs -- raced to map the laundering infrastructure.

As of publication, approximately 88% of the stolen funds have been traced, though recovery remains partial. The Lazarus Group's laundering operation continues to evolve, using bridge protocols and decentralized exchanges to obscure the trail.

Aftermath

Safe{Wallet} rebuilt its entire infrastructure from scratch. All credentials rotated. New monitoring and validation layers added. The native Ledger integration was temporarily removed. A phased rollout restored the platform on Ethereum mainnet with additional security measures.

Mandiant's forensic investigation confirmed the 19-day attack timeline and the North Korean attribution. The sophistication of the operation -- patient, targeted, surgically precise -- was consistent with TraderTraitor's previous campaigns.

Lessons

This wasn't a crypto hack. It was a **supply chain attack** that happened to target crypto infrastructure. The lessons apply to any organization that depends on third-party software:

1. **UI integrity matters.** If your signing interface can be compromised, your cryptographic guarantees are meaningless.
2. **Minimize your attack surface.** Purpose-built contracts beat general-purpose ones. Don't ship `delegatecall` if you don't need it.
3. **Developer workstations are the front line.** A compromised laptop led to a $1.5 billion loss. Harden them accordingly.
4. **MFA isn't magic.** Session token theft bypasses MFA entirely. Monitor for anomalous session usage.
5. **Verify, then verify again.** Transaction verification should happen through independent channels, not through the same UI that could be compromised.

The largest heist in history wasn't executed with zero-days or quantum computers. It was a Docker project called "Stock-Invest-Simulator" and a developer who clicked download.

`-- chainwatch`

ETHERHIDING: WHEN THE BLOCKCHAIN BECOMES YOUR C2

by Mara Okonkwo (@bl0ckd3ad) — 2026-02-13

tags: dprk, blockchain, c2, etherhiding, malware, ethereum

ETHERHIDING: WHEN THE BLOCKCHAIN BECOMES YOUR C2

Traditional command-and-control infrastructure has a fundamental weakness: it's centralized. A domain can be sinkholed. A server can be seized. A CDN can be subpoenaed. Law enforcement and security researchers have spent decades perfecting the playbook for disrupting C2 channels.

North Korea's UNC5342 decided to make that playbook obsolete.

EtherHiding stores malware payloads and C2 instructions inside smart contracts on public blockchains -- Ethereum and BNB Smart Chain. The "server" is a distributed network of thousands of nodes. Nobody controls it. Nobody can take it down. The data is immutable.

> **[HACK]** Once C2 information is committed to the blockchain, it possesses a level of permanence and availability that traditional hosting cannot offer. There is no central server to seize, no registrar to compel, no hosting provider to contact.

How Traditional C2 Fails

The cat-and-mouse game of C2 infrastructure follows a predictable pattern:

```
Attacker registers domain → Malware beacons to domain
     ↓
Security team discovers domain → Contacts registrar
     ↓
Registrar suspends domain → C2 channel dies
     ↓
Attacker registers new domain → Cycle repeats
```

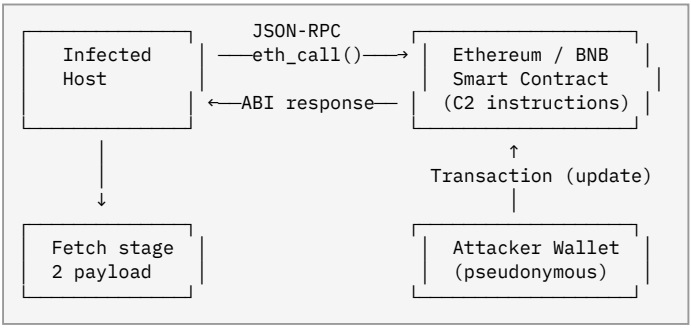
Variations exist -- fast-flux DNS, domain generation algorithms (DGAs), Tor hidden services -- but they all share a common trait: somewhere in the chain, there's an entity that can be compelled or compromised to disrupt the channel. Registrars comply with abuse reports. Hosting providers respond to court orders. Even bulletproof hosting in sympathetic jurisdictions eventually gets shut down.

EtherHiding breaks this model entirely.

The Architecture

The technique is elegant in its simplicity:

1. **Deploy a smart contract** on Ethereum or BNB Smart Chain. The contract contains encoded C2 instructions or payload URLs in its storage variables.
2. **Malware queries the blockchain** using standard JSON-RPC calls (e.g., `eth_call`) through public RPC endpoints like Infura, Alchemy, or any Ethereum node.
3. **The response is parsed** to extract the C2 address, payload URL, or configuration data.
4. **Update when needed.** If the contract is deployed with a mutable storage pattern, the attacker can update C2 instructions by sending a new transaction -- while all historical data remains immutable on-chain.



From a network monitoring perspective, the malware's traffic looks like ordinary Web3 API calls -- the same requests that millions of DeFi applications, wallets, and dApps make every second.

The DPRK Toolchain

Google's Threat Intelligence Group (GTIG) first observed UNC5342 adopting EtherHiding in February 2025, marking the first time a nation-state actor embraced the technique. The campaign was tied to the "Contagious Interview" operation -- fake job postings targeting software developers.

JADESNOW (Stage 1 Downloader)

The initial JavaScript downloader interacts with a malicious BNB Smart Chain contract to retrieve the next-stage payload. JADESNOW queries the transaction history associated with a specific Ethereum address to fetch encrypted payload URLs.

INVISIBLEFERRET (Stage 2 Backdoor)

A Python-based backdoor with extensive capabilities:

* **Browser data theft** -- credentials, cookies, crypto wallet extensions
* **Cryptocurrency targeting** -- steals from 13+ different wallet types
* **Keylogging and clipboard monitoring**
* **File exfiltration** via FTP and Telegram bots
* **Persistent remote access** via AnyDesk installation
* **System reconnaissance** -- geolocation, system fingerprinting

EtherRAT (Post-Exploitation)

Documented by Sysdig, EtherRAT was deployed via React2Shell (CVE-2025-55182) exploitation. It represents the convergence of two DPRK capabilities: the EtherHiding C2 technique with the React2Shell initial access vector.

> **[INFO]** The full infection chain: **Social engineering** (fake job interview) → **JADESNOW** (JavaScript downloader, blockchain C2) → **INVISIBLEFERRET** (Python backdoor) → credential theft, crypto wallet drain, persistent access.

Why It's Hard to Stop

The advantages for attackers are significant:

Traditional C2	EtherHiding C2
Domain can be sinkholed	No domain to sinkhole
Server can be seized	"Server" is thousands of nodes
Hosting provider can comply with law enforcement	No single entity to compel
Traffic is anomalous	Traffic mimics legitimate Web3 calls
Historical infrastructure can be studied post-takedown	All data remains on-chain permanently

Blocking specific RPC endpoints (Infura, Alchemy) provides only temporary relief -- the malware can switch to any public Ethereum node, including self-hosted ones. The blockchain itself is the infrastructure, and blocking the blockchain means blocking all of Web3.

The pseudonymous nature of blockchain transactions makes attribution harder, too. While chain analysis can sometimes link wallet addresses to identities, deploying a smart contract from a freshly generated wallet leaves minimal traces.

Detection Approaches

EtherHiding isn't invisible. It has detectable characteristics if you know where to look:

Network monitoring:

* Watch for `eth_call` or `eth_getStorageAt` JSON-RPC requests from hosts that have no business interacting with blockchain infrastructure
* Monitor for connections to public RPC endpoints (Infura, Alchemy, QuickNode) from non-Web3 applications
* Flag processes that make blockchain API calls followed by downloading and executing payloads

Endpoint detection:

* Look for JavaScript or Python processes making Web3 RPC calls
* Monitor for ABI decoding libraries being loaded by unexpected processes
* Watch for the JADESNOW → INVISIBLEFERRET execution chain

Smart contract analysis:

* Contracts storing encoded URLs or IP addresses in their storage
* Contracts with update functions called from known-malicious wallets
* Unusual patterns in contract interaction frequency

```bash
# Example: Query a contract's storage slot for suspicious content
cast storage <contract_address> 0 --rpc-url https://eth.llamarpc.com
# Decode and check for URLs, IP addresses, or encoded payloads
```

[WARN] If you detect EtherHiding C2 traffic on your network, assume full compromise. The malware families deployed through this technique -- particularly INVISIBLEFERRET -- are designed for deep, persistent access and credential theft.

The Philosophical Problem

There's an irony that's hard to ignore: blockchain technology was built on the principles of decentralization, censorship resistance, and immutability. These were *features*, designed to protect against authoritarian control and corporate censorship.

North Korea is using those exact features to run untakedownable malware infrastructure.

The blockchain doesn't distinguish between a DeFi protocol storing collateral ratios and a nation-state storing C2 instructions. Both are just bytes in a smart contract. The censorship resistance that protects activists in authoritarian regimes also protects the attackers stealing billions in cryptocurrency to fund weapons programs.

This isn't a bug in the blockchain. It's a fundamental tension in the design philosophy. And it doesn't have a clean solution.

The server you can't take down is the one that belongs to everyone.

`-- bl0ckd3ad`

AIRBORNE: ZERO-CLICK WORMS IN YOUR LIVING ROOM

by Dmitri Volkov (@rf_ghost) — 2026-02-13
tags: apple, airplay, zero-click, wormable, rce, iot

AIRBORNE: ZERO-CLICK WORMS IN YOUR LIVING ROOM

In April 2025, researchers at Oligo Security disclosed something that made every Apple device owner reconsider their network trust model: **23 vulnerabilities** in Apple's AirPlay protocol, collectively dubbed "AirBorne." The most critical of these allow **zero-click, wormable remote code execution** -- meaning an attacker on your network can compromise your MacBook, then your HomePod, then your TV, then every AirPlay device on every network those devices subsequently connect to.

No user interaction required. No click. No prompt. No notification.

The Protocol

AirPlay is Apple's proprietary protocol for wireless streaming of audio, video, and screen mirroring between devices. It's been a core feature of the Apple ecosystem since 2010 and is embedded in:

* Every Mac, iPhone, iPad, Apple TV, and HomePod
* Hundreds of third-party speakers, receivers, TVs, and car infotainment systems via the **AirPlay SDK**
* CarPlay-equipped vehicles

The protocol operates over WiFi using a combination of mDNS for service discovery, HTTP/RTSP for session negotiation, and custom encryption for media streaming. Devices advertise themselves on the local network and accept connections based on configurable access control.

The Vulnerabilities

Of the 23 vulnerabilities Oligo reported, Apple assigned **17 CVEs**. The three most critical:

CVE-2025-24252: Use-After-Free (CVSS 9.8)

A use-after-free vulnerability in macOS's AirPlay service. When the service handles malformed network packets, a memory object is freed but a dangling pointer remains. The attacker triggers use of this pointer to achieve arbitrary code execution.

This is the **wormable** vulnerability. Combined with CVE-2025-24206, it enables zero-click propagation across devices.

CVE-2025-24206: Authentication Bypass (CVSS 7.7)

An authentication bypass that eliminates the user interaction requirement. AirPlay receivers have a configuration setting: "Allow AirPlay for" with options like "Current User," "Anyone on the Same Network," or "Everyone." CVE-2025-24206 bypasses the "Accept" click that normally appears even on permissive configurations.

This is the **zero-click enabler**. Without it, CVE-2025-24252 requires the victim to accept an AirPlay connection. With it, the exploit is fully automated.

CVE-2025-24132: Stack Buffer Overflow (CVSS 6.5)

A stack-based buffer overflow in the **AirPlay SDK** -- the library Apple provides to third-party manufacturers. This vulnerability achieves zero-click RCE on speakers, receivers, and other third-party devices **under all configurations**, regardless of access control settings.

[WARN] CVE-2025-24132 affects third-party devices running the AirPlay SDK. Apple can patch its own devices, but SDK-dependent hardware relies on individual manufacturers to push firmware updates. Many will never be patched.

The Worm Scenario

Here's what makes AirBorne existential: the exploit is **wormable**. Consider this attack chain:

```
1. Attacker compromises MacBook on coffee shop WiFi
   ↳ CVE-2025-24252 + CVE-2025-24206 (zero-click RCE)

2. Victim carries MacBook to corporate office
   ↳ Connects to corporate WiFi

3. Compromised MacBook propagates to office AirPlay device
s
   ↳ Conference room Apple TV, HomePod in the lobby

4. Compromised Apple TV propagates to other devices on net
work
   ↳ Other MacBooks, iPhones with AirPlay Receiver enable
d

5. Those devices move to other networks
   ↳ Home networks, other offices, hotels

6. Propagation continues indefinitely
```

No lateral movement tools needed. No second-stage payload. The worm uses AirPlay's own service discovery to find new targets and the same vulnerability chain to compromise them. The attack propagates silently, leaving no visible indication on the victim's device.

> **[HACK]** The default macOS configuration has AirPlay Receiver **enabled** and set to "Current User" -- which, combined with the ACL bypass vulnerabilities, still presents exploitable attack surface. Changing to a more restrictive setting reduces but doesn't eliminate the risk.

The Third-Party Problem

Apple patched its own devices on March 31, 2025:

- iOS / iPadOS 18.4
- macOS Ventura 13.7.5, Sonoma 14.7.5, Sequoia 15.4
- visionOS 2.4

The AirPlay SDK patches were released to the MFi (Made for iPhone/iPad) Program:

- AirPlay audio SDK 2.7.1 (March 31, 2025)
- AirPlay video SDK 3.6.0.126 (March 31, 2025)
- CarPlay Communication Plug-in R18.1 (April 4, 2025)

But here's the problem: **the MFi ecosystem includes hundreds of manufacturers** producing speakers, receivers, TVs, set-top boxes, and car infotainment systems. Each manufacturer must independently integrate the patched SDK and push firmware updates to their devices.

How many Sonos speakers, Bose receivers, and Samsung TVs will actually receive the update? How many hotel room Apple TVs? How many rental car CarPlay systems?

The honest answer: a fraction. Consumer electronics have notoriously poor patch lifecycles. Many AirPlay SDK devices will remain vulnerable for their entire operational lifetime.

Responsible Disclosure Timeline

Date	Event
Late 2024	Oligo begins AirPlay protocol research
Early 2025	23 vulnerabilities identified and reported to Apple
March 31, 2025	Apple releases patches for iOS, macOS, visionOS, and SDK
April 4, 2025	CarPlay plug-in patch released
April 28, 2025	Apple pushes additional fixes via Rapid Security Response
April 29, 2025	Oligo publishes AirBorne research

Mitigation

If you're reading this, here's what you should do right now:

For Apple devices:

```
Settings → General → Software Update → Install latest vers
ion
```

For AirPlay configuration:

```
Mac: System Settings → General → AirDrop & Handoff → AirPl
ay Receiver → Off
     (or set "Allow AirPlay for" to "Current User" at mini
mum)

iPhone/iPad: Settings → General → AirPlay & Continuity → A
irPlay Receiver → Off
```

For network administrators:

* Block AirPlay traffic (TCP/UDP port 7000) between untrusted network segments
* Isolate IoT and AV devices on separate VLANs
* Audit third-party AirPlay devices and contact manufacturers for firmware updates
* Consider disabling AirPlay on managed devices via MDM

For third-party device owners:

* Check manufacturer websites for firmware updates
* If no update is available, isolate the device on a dedicated IoT network
* Consider whether the device needs to be network-connected at all

[INFO] If you can't patch a third-party AirPlay device, network segmentation is your best defense. An unpatched speaker on an isolated VLAN can't be used as a pivot point even if it's compromised.

The Bigger Picture

AirBorne demonstrates a pattern we see repeatedly in IoT security: a protocol designed for convenience is implemented without adversarial threat modeling, embedded in billions of devices via an SDK, and then found to be fundamentally broken.

Apple responded quickly and patched their own ecosystem. But the SDK devices -- the speakers, the TVs, the car systems -- exist in a firmware update wasteland where patches arrive late, if ever. The attack surface isn't just Apple's to manage.

23 vulnerabilities. 17 CVEs. Billions of devices. Zero clicks required.

Your living room is listening. Make sure it's patched.

If your AirPlay receiver is on, turn it off. Then update. Then turn it back on if you actually need it.

`-- rf_ghost`

HUNTING THE GIANT CENTIPEDE: RHYSIDA

by Leland Flynn (@c0mmrade) — 2026-02-13

tags: ransomware, rhysida, incident-response, threat-hunting, original-research

At the time of writing I still worked for CTS. Tense of the article has been maintained for this author's sanity.

HUNTING THE GIANT CENTIPEDE: RHYSIDA

Background

My name is Leland but, you can call me `c0mmrade` if you'd like. I'm a Security Researcher and Senior SecOps Engineer for CTS. I've worked in the tech space in varying capacities for over 24 years, and exclusively in Infosec the past 3 years. I've considered myself a hacker since the first time I encountered the word, though I was certainly hacking things around me well before that. Most of my knowledge has been self-earned with some formal training during my time in the U.S. Army and various structured resources like HackTheBox.

Recently I've had the opportunity to expand CTS' Security Operations Center both in capability and personnel. I'm looking forward to learning so much more as we grow.

Over the past 2 months I have been investigating the activity of the Rhysida ransomware operation. The following is an informal recounting of my experience.

About the Attacker

Rhysida is a ransomware operation that emerged in mid-2023, quickly establishing itself as a significant threat to the Internet. First identified in May 2023, the group is named after a centipede genus and features a distinctive centipede logo.

The group's sudden appearance suggests they may be experienced cybercriminals, potentially an evolution of the Vice Society ransomware operation. By June 2023, they had already set up a professional dark web data leak site employing double-extortion tactics—encrypting data and threatening to publish stolen information.

A notable attack against London-based talent agency The Agency[1] highlighted their capability to cause substantial financial and personal damage, leaking sensitive information like passports and other confidential documents.

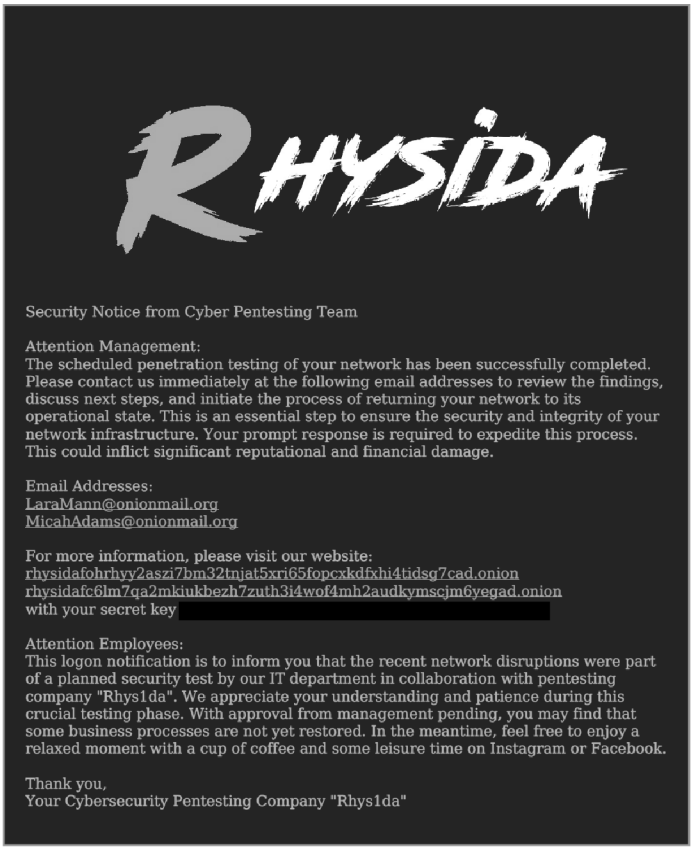
Despite their short history, Rhysida has distinguished itself through technical sophistication, strategic targeting, and a *typically* well-organized operational approach.

Their leak-site on TOR boasts 172 victims as of this writing but, I know for certain they've hit many more organizations during recent months. I've witnessed this firsthand, managing to gain a foothold in one of their previously unknown Command-and-Control (C2) servers, and trying my best to notify victims as I identified them.

Victim_Zero

Like most days I was up a little too early for my liking. I was just getting in my truck for my morning commute when a text came in from Alex, our CTO, "`Have an emergency can you call me when you get a second`". One of our unmanaged clients (not contracted for SOC monitoring, updates, etc.) was contacting us for help. They'd been hit with ransomware, most of their servers were unresponsive, and their business was completely offline.

By the time I got to the office, Alex had already shutdown most of their Hyper-V virtual machines and isolated the server. But it was clear that the damage was extensive. The attackers targeted the most common, high-value files; SQL databases, PDFs, Excel spreadsheets, etc. All appended with the extension `.rhysida`. Every directory was littered with PDF ransom notes identifying the attacker as the Rhysida ransomware operation and explaining how to contact them.



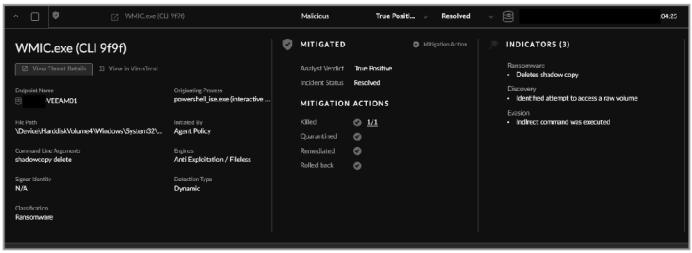
Rhysida ransom note with .rhysida file extension

Recovery

We immediately collected forensic memory and disk images of the Hypervisor in hopes of finding out more about our attackers and the malware they were using (thanks FTK!). If you've dealt with ransomware before you know the drill: isolate infected hosts, gather forensic images, and restore from backups. If you don't have backups then be ready to pony up the ransom or kiss your data goodbye. Thankfully, they had backups to restore from right!? Well, sort of. *Victim_Zero* was not following best practices. They were backing up all their VMs with Veeam, but they had no off-site replication. Further complicating things, our

client was not regularly updating their operating systems or software; so getting to the Veeam server's soft underbelly would have been trivial. I'm sure you can see where this is going.

Most ransomware groups make it a point to encrypt or delete obvious backups, increasing their chances of a payout. We checked their Veeam backups and confirmed that most of the datastores had been encrypted. So our next step was to check the Volume Shadow Service for copies of the data that we could restore from. One of the known Indicators of Compromise (IOCs) pertaining to Rhysida is an attempt to run `cmd.exe vssadmin delete shadows /all /quiet` (MITRE ATT&CK - T1490) to inhibit system recovery after they've exfiltrated the data. I mentioned earlier that this client was not under contract with our SOC, but thankfully they had bought SentinelOne through us for their servers, and this is the exact activity that our trusty EDR first identified and stopped.



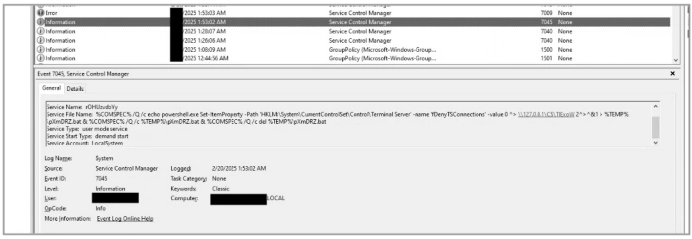
SentinelOne EDR alert showing blocked VSS deletion

So, good news! We still had shadow copies of VHDs that we could recover the environment from. The trouble was, they were living on an isolated machine potentially still infected with malware. We had our on-site contact pick up a few external drives large enough to store the recovered VHDs and got to work making copies, while Alex started spinning up a crisp, clean new infrastructure in the background. With a plan for recovery of their servers in action, I could turn my attention to threat hunting, and hopefully identify who to block.

Threat Hunting

I was unfamiliar with this group, so I took a moment to research them online. Like Rage said, "know your enemy". Honestly, they all start to blend together with their terrible names and ridiculous posturing. After reading some of the great work published by CISA[2], Sophos[3], and Picus[4] I was armed with some useful information about how Rhysida operates, common IOCs, and a few details about their history. I searched the firewall logs for any of the IOCs I had found but came up short. But just to be safe I added policies blocking related domains and IPs.

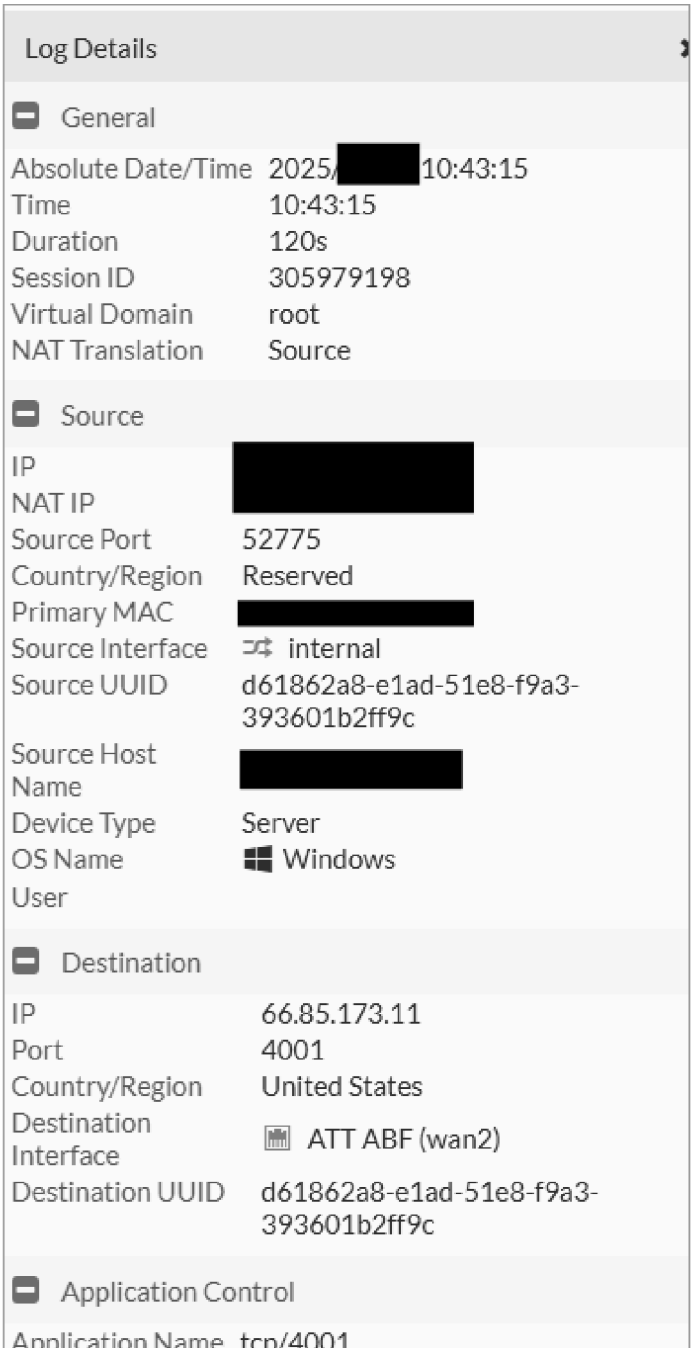
From the incidents reported by SentinelOne I could see that the final phase of their attack on our client began at roughly 2:04AM on 20FEB2025 from one of their Domain Controllers, putting the initial access phase somewhere between 3-7 days prior by rough estimation. Our client gave us the name of at least 1 workstation they were certain had been infected, so I decided to start there. Not long into digging I found the first signs of credential dumping, lateral movement, and C2 activity:



Evidence of credential dumping and lateral movement

Without access to the client's email logs and a proper EDR on their workstations, working out the initial access vector would be tough and consume time that we just didn't have. With our managed clients, I have access to a wealth of tools and resources like Barracuda for spam/phish filtering, Sentinel One and Huntress for EDR, and Blumira as our SIEM. These tools would have sped my work up considerably. But I did have some new timestamps, and a pathological inability to let go of anything.

So I turned my attention back to the firewall logs, looking for any abnormal traffic. After channeling my inner Angela Lansbury and cranking anything 173bpm or higher, I started seeing a pattern of activity. Thousands of log entries showing traffic from *Victim_Zero's* domain controller, over port 4001, to a public IP address of `66.85.173.11` and a related domain name of `patterson.pureskin.cloud`. Each session moving 50MB or more at a time. That looked an awful lot like data exfiltration to me.


Log Details
General
Absolute Date/Time 2025/[redacted]10:43:15
Time 10:43:15
Duration 120s
Session ID 305979198
Virtual Domain root
NAT Translation Source
Source
IP
NAT IP
[redacted]
Source Port 52775
Country/Region Reserved
Primary MAC [redacted]
Source Interface internal
Source UUID d61862a8-e1ad-51e8-f9a3-393601b2ff9c
Source Host Name [redacted]
Device Type Server
OS Name Windows
User
Destination
IP 66.85.173.11
Port 4001
Country/Region United States
Destination Interface ATT ABF (wan2)
Destination UUID d61862a8-e1ad-51e8-f9a3-393601b2ff9c
Application Control
Application Name tcp/4001

Category	unscanned
Risk	undefined
Protocol	6
Service	tcp/4001

Firewall logs showing exfiltration traffic to 66.85.173.11

Say what you will about Fortinet (and I have a lot to say) but they have a fantastic traffic log UI, and it makes work like this so much easier than some other vendors.

I scoured the logs multiple times over looking for other public IPs with similar behavior but found nothing. So, did we have a smoking gun? Was the city of Townsville once again safe? Hardly. But what I did have was an IOC that, as far as I can tell, has never been published before now.

Strap-in, this is where it gets silly.

Kick in the Door, Wavin' the .44

Rhysida, what's your deal? Why are you lashing out against non-profits, modelling agencies, and Police departments like an angry teenager? Who hurt you?

First I ran a search against WHOIS, expecting the IP to be registered in some former-Eastern Bloc territory, or maybe Southeast Asia. But Tempe, Arizona? The astute reader will no doubt expect this information might be actionable at a Federal level. And they would be right. *Oooooh, foreshadowing.*

Now I knew this IP related to our Rhysida attacker belonged to a hosting provider in the states, and I had contact info for them. Contacting the hosting company's abuse team to report the activity would likely cause the attackers to scatter and just move their activity to another server. And I much preferred taking this opportunity to learn some more about our little centipedes. It was time for some enumeration with Nmap!

```
Nmap scan report for 66.85.173.11
Host is up (0.028s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE    SERVICE    VERSION
22/tcp   open     ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu
0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 97:a2:d2:c3:e5:66:d6:f5:28:b0:51:f8:fe:04:57:fb (E
CDSA)
|_  256 39:1e:c3:e5:6f:bc:b7:47:a1:6a:89:f7:7f:e5:b6:9c (E
D25519)
80/tcp   open     http       Apache httpd 2.4.52
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2025-01-09 12:09  aafc820801b6c15b8ef17af5a833c54
7/
| -     2025-01-09 12:04  ed00852591680f1c3001d5ac0afc1d12
-1736421822/
|_
|_http-title: Index of /
|_http-server-header: Apache/2.4.52 (Ubuntu)
111/tcp  filtered rpcbind
4001/tcp open     newoak?
4321/tcp open     tcpwrapped
Aggressive OS guesses: Linux 2.6.32 - 3.13 (96%), Linux 5.
0 - 5.14 (96%), ...
No exact OS matches for host (test conditions non-ideal).
Network Distance: 11 hops
Service Info: Host: 66.85.173.11; OS: Linux; CPE: cpe:/o:l
inux:linux_kernel

Nmap done: 1 IP address (1 host up) scanned in 188.42 seco
nds
```

It was listening for SSH but, unfortunately just a version past a viable Authentication Bypass exploit. Apache was listening on port 80, and it looked like there were some interesting directories listed. There was our exfil port on 4001, and not much else. I was feeling a little dangerous and thought, "Let's try kicking in the door"!

I fired up a clean VM outside our environment and connected to the web server to take a look. The first directory rendered a password prompt, and the second gave a string of random characters. The random string in the second directory looked an awful lot like a password that was left exposed, but why would they just leave the key under the mat like that? I'm still unsure if this was a case of incompetence or apathy, but regardless of the

reason, it's terrible opsec. I pasted the password into the prompt and that was it, I had just logged into Rhysida's C2. *Surely it could not be that easy!?* I'm sure there are more that they can pivot to, which may be why they just don't seem to care about protecting it.

From here I could see a list of active connections between the C2 and their victims, with each row displaying:

* Session Port
* Build Number of the Implant Malware
* Domain\Hostname *and optional credentials for the session*
* The Public IP and Geolocation of the host
* Uptime of the session
* Other misc options

This wasn't a C2 I was familiar with, so I made a note to research it a later. But it was fairly apparent that these were SOCKS sessions, with the C2 acting as a proxy. But what I was witnessing now was multiple robberies in slow-motion. People needed to be notified. Some of the sessions were new enough that the attackers on the other end might only be in their initial access or enumeration phases. That meant if I reached the right people quickly enough, they could shut down servers and restore from backups before it got bad.

Identifying a Body

I might not know who the victims were, but I had enough info to do some basic Open-Source Intelligence gathering (OSINT), and narrow down who to contact. *I'll avoid discussing most of the victims I've spoken to in great detail to protect their privacy; omitting names where needed.*

One of the first victims (*victim_01* for those counting) I investigated had 2 hosts in the same domain, with the same public IP (65.255.81.224, US, Oklahoma) listed under the "Online" section of the C2 control panel. A quick WHOIS search for that IP confirmed the victim was Okeene Public Schools in Okeene, Oklahoma. *A representative of Okeene Public Schools was kind enough to approve my sharing of specific details around their incident.*

A quick web search turned up the homepage for the public school and a phone number for their Administration Office. I spoke to a very nice, if slightly confused receptionist, and tried to calmly explain the situation. I was trying not to be too alarmist, explaining that I was a security researcher investigating a ransomware attack on my client, and that led me to identifying more potential victims, I was sure that they we actively under threat, and asked to speak to their IT support staff. I'm sure she thought I was a scammer as she tried to brush me off at first. But persistence and a bit of concerned pleading paid off, she took my contact information, and a few hours later a representative from their MSP called me.

I gave their rep the technical details, showed him what I saw from the C2, and he confirmed for me that these were his client's servers. I walked him through collecting forensic images of one server's memory and disk before he shut down everything in hopes of restoring from backups. We even managed to collect a copy of a malicious DLL running on the server and the scheduled task that called it.

Was that a win!? Would my time and focus might really pay off!? I immediately got up from my desk and kissed my wife to celebrate.

But we were still left with the question, 'how did they get in in the first place'?

Initial Access – A Working Theory

By this point I had started running Nmap scans against every victim IP I saw, looking for obvious weak points in their attack surface, and thinking about how I would hack them. Most, but not all of the victims during this period seemed to be using Fortigate firewalls, running outdated firmware, with their management interfaces (web browser login prompt) exposed to the open Internet. I was aware of a recent Proof-of-Concept exploit[5] released by Sonny of WatchTowr Labs (*I cannot fanboy on these folks hard enough, you guys are awesome!*) that exploited `CVE-2024-55591` resulting in an authentication bypass on vulnerable Fortigate devices.

In fact, I had just spent a few days researching the vulnerability and the exploit about 2 weeks before this whole situation began. CTS has primarily been a Fortigate shop, so I try to stay on top of everything related to them. In my spare time I was playing around with Sonny's code in my lab to see how our SIEM responded to it. Being responsible hackers, the WatchTowr team release the code as PoC with a calling card built-in. Inserting a message into the logs which includes the IP/port combination `[13.37.13.37]:1337`. This would make it obvious to anyone checking their Firewall logs that some script kiddie downloaded the PoC and tried to use it to attack a real target. They also, intentionally I assume, included an infinite `while` loop, making the exploit hang after executing. So I made a few light modifications to make the script a bit more useful. I'm a mediocre Python programmer at best, so if I could identify and fix these with ease, I'm sure others have as well.

Using their PoC code as an example, I wrote a very basic vulnerability scanner and used it to identify any of our client's devices that might be vulnerable, and we started patching and mitigating immediately. *Victim_Zero* was on this list, but by this point Rhysida would have already been in the firewall and started their lateral movement phase. Unfortunately I didn't have access to Okeene's

firewall to check the logs, but when I ran my vulnerability scanner against it I got a positive result. And the same went for every other victim that I identified as using Fortigate. Every single one would give up vital information, change configurations, even add new administrator accounts. All we had to do was ask nicely.

Rhysida seems to have favored phishing for initial access in the past, but with a ton of vulnerable Fortigate devices on the Internet and some easy to run exploit code, I suspect this was too easy an opportunity to pass up. ***It's worth noting that I don't have enough proof to present this a verifiable fact currently, this is just an educated guess***.

Victim_Zero and Victim_02

I had contacted multiple victims by this point and left voicemails, sent emails, and even used a support ticket interface to alert one. *Side-note: Have I Been Pwned and similar services are an overlooked OSINT resource for vetting email addresses. Trying to verify an email address using only a person's name and their business name? If the organization is big enough the chances are high that someone's address has been cataloged by HIBP. Make a list of a few potential addresses using common conventions (first.last@company.com[6]) and you have a quick and dirty address verification.* The responses varied from silence to dismissal mostly. But it was time to turn my attention back to our client. Their infrastructure was recovered thanks to Alex' hard work. Another tech and I would have to make the long drive out to them to check each workstation by hand for signs of the malware, clean up what we could, and scrap/replace the rest. This process took the better part of 2 days, across multiple locations, but it was well worth it.

At night I spent some more time poking at Rhysida's C2 from the hotel, trying to find more information about it, and getting increasingly louder with my attempts to determine who might be on the other side. I ran dirbuster with a few common wordlists against it, trying to enumerate any other directories but, all I really found was a directory holding the GeoIP database the C2 used to determine where the victim IPs were located. Attempts to bruteforce SSH bore even less fruit. But I did finally identify the C2 as a variant of SystemBC thanks to this awesome write-up by the folks at Kroll[7].

I was already 30 minutes into my long drive home, when I received an email back from one of the victims (we'll call them *Victim_02*) who was conveniently located in a nearby city. I had emailed with their staff after initially identifying them but, they felt they had the situation under control. Still driving I jumped into a call with the CEO of *Victim_02* and explained

who I was, how I found them, and what I've seen with the attacks so far. It turned out they had responded to the threat too late; Rhysida had encrypted most of their environment and most of their business was offline. Just like *Victim_Zero*, the attackers exfiltrated important data, deleted VSS copies, encrypted everything including Veeam backups, and dropped a ransom note. The CEO asked for any help we could offer, he'd already spent a sleepless night or two in their datacenter trying to recover things.

45 minutes later I was sitting across from *Victim_02's* CEO in a DataCenter conference room, walking them through what I saw from the C2, and demonstrating an exploit against their vulnerable Fortigate firewall. Since that meeting they've become one of our newest clients. *I can't stress enough that my vigilante activities against Rhysida were borne out of an intense, near-pathological hatred of bullies and cowards. I was not then, and am not now, seeking business from this. I just want to make Rhysida hurt, if even just a little bit.*

The Call is Coming from Inside the House

Shortly after getting home I was working with *Victim_02's* team to get a forensic image of one of their infected servers. This proved a little difficult since the memory alone was about 2TB. With no viable options for a disk image we opted for just the RAM and got that copied to a clean external disk. Unfortunately they had already factory reset their Fortigate, so any logs we might have had were lost. *Part of me wishes I had hacked their firewall when I first identified them and locked Rhysida out. The thought certainly crossed my mind.*

They were recovering data from cold storage tape at this point but it was certainly a better position than some victims were in. Over the next few days we began bringing their infrastructure back online, slowly, making sure each host had our remote access, SIEM, and EDR agents deployed before they were allowed to join the network. At least, that was the expectation.

I had already created policies blocking the Rhysida C2 and disabled the management interface on their WAN port, I just couldn't upgrade the firewall to a later firmware release since the model was end-of-life. But, it came as a surprise when our SIEM alerted the SOC team about 'external IP' `13.37.13.37` successfully logging in to the Fortigate. That's the artifact that WatchTowr's PoC code generates, remember? And all external access to the firewall was being denied. The attack had come from inside *Victim_02's* LAN.

Predictably whatever device ran the script deleted all the traffic logs, leaving little to go on. But I finally had some direct evidence of the attacker using WatchTowr's Fortigate exploit in the wild.

Between the C2 access, forensic data, logs, malicious DLLs, and anecdotal evidence I had collected from other victims I felt like I might have something useful to bring to the authorities.

Snitching to the Feds

Victim_02 had already started a dialog with the FBI about their situation, and we had made it clear that we were happy to share whatever information we could. After a nice, sobering talk with our attorney, I soon found myself on a conference call in the middle of the day, from my home office, showing Federal Agents around the C2 of an international ransomware operation. This whole situation still feels a little surreal. But it was reassuring to have some of my suspicions confirmed, and to exchange information with experts.

When I searched the internet for mentions of the C2's IP I couldn't find anything. It wasn't in any IoC lists or malicious IP search I could find. And it turned out the FBI hadn't seen it either. So I packaged up all my notes, evidence, etc. and sent it to them. As of this writing the FBI agents I've worked with intend to get a forensic copy of the server, and hopefully get closer to identifying Rhysida members.

It's worth noting that the server is still online as of this writing. I'm still monitoring it and contacting new victims as I see them. I estimate that we've successfully shut down around 14 attacks so far.

Conclusion

Rhysida's ransomware attack on our client was a stark reminder of the ever-evolving threat landscape in Infosec. It also reinforced the value of curiosity for me, both personal and professional. We could have just blocked their C2 for all our clients, and left it at that. Instead my team continues to help me see this through as far as I can take it. I owe a debt of gratitude and respect to the team I work with for their dedication to our clients, their incredibly hard work, and their support when I go off on tangents like this.

I could never have predicted where that first text message from Alex would lead. From a standard ransomware cleanup came a complex investigation spanning multiple organizations, and the infiltration of a command and control server providing rare insight into the inner workings of one of the most prolific ransomware operations in recent memory.

In the end, the battle against ransomware will go on for as long as it remains profitable, but with the right tools, knowledge, and collaboration, we can mitigate risks and safeguard our digital assets. Stay safe out there, and know your enemy.

Indicators of Compromise

C2 IP: `66.85.173.11` **Domain:** `patterson.pureskin.cloud`

Activity: Continuous, high-volume traffic over port 4001 to single IP

File(s): `driver64.dll` **SHA-1:** `b031ea666f4679f512b7e135aae31af755278b82` **SHA-256:** `a8659f60b63b4ef781439de7676fded950a238ebd926997c70174e5d7ef43529`

Malicious Scheduled Task Name: `{system}.xml` **Contents:** Attempts to load malicious DLL via rundll32 `"%AppData%\Local\driver64.dll rundll"`

The investigation continues in Phase 2[8].

`-- c0mmrade`

LINKS

[1] https://www.theregister.com/2025/02/19/london_celebrity_talent_agency_reports/
[2] https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-319a
[3] https://github.com/sophoslabs/IoCs/blob/master/2311%20Vice%20Society%20-%20Rhysida%20IoCs.csv
[4] https://www.picussecurity.com/resource/blog/rhysida-ransomware-explained
[5] https://labs.watchtowr.com/get-fortirekt-i-am-the-super_admin-now-fortios-authentication-bypass-cve-2024-55591/
[6] mailto:first.last@company.com
[7] https://www.kroll.com/en/insights/publications/cyber/inside-the-systembc-malware-server
[8] /vol/1/10-rhysida-phase-2

GHOST TAP: NFC RELAY FRAUD AT SCALE

by Reza Hajian (@tapjack) — 2026-02-13

tags: nfc, fraud, mobile, malware, android, fintech

GHOST TAP: NFC RELAY FRAUD AT SCALE

A research tool built by a German university to study NFC security has been weaponized into a criminal malware-as-a-service platform. Victims install an app, tap their credit card, and their payment credentials are relayed in real-time to a mule's device thousands of miles away -- who walks into a store and buys luxury goods on the victim's dime.

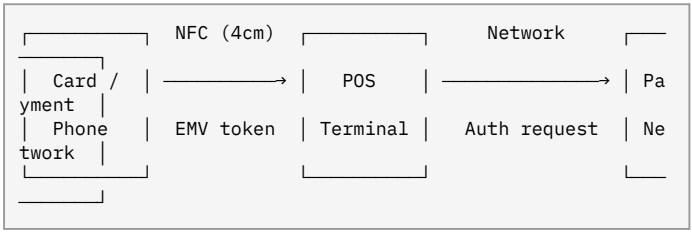
ESET's telemetry shows NFC-based attacks surged **35x in H1 2025** compared to the previous six months. In Russia alone, losses from NFCGate variants hit 432 million rubles (~$5.5M) in Q1 2025, with an average of 40 successful attacks per day.

This is Ghost Tap. And it's just getting started.

How NFC Payments Work

Near Field Communication (NFC) enables contactless payments at distances under 4cm. When you tap your phone or card at a terminal:

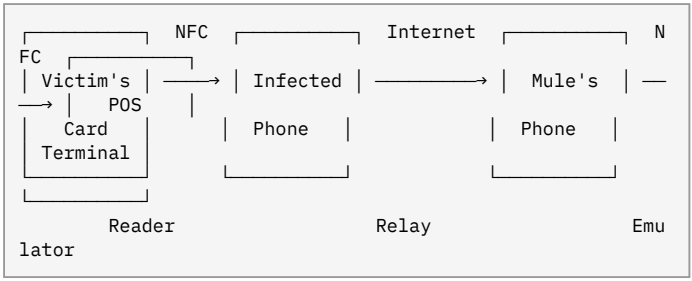
1. The terminal generates an electromagnetic field
2. Your card/phone responds with a cryptographic token (EMV)
3. The token is validated against the payment network
4. Transaction approved (typically under $100 without PIN)



The security model assumes **physical proximity**. If your card generates a valid token, the terminal assumes the cardholder is standing in front of it. NFC relay attacks break this assumption.

The Relay Attack

An NFC relay attack inserts a proxy between the card and the terminal, extending the effective range from centimeters to continents:



The victim's phone reads the card's NFC data and streams it over the internet to the mule's phone, which emulates the card to the POS terminal. The entire relay happens in real-time -- the EMV cryptographic challenge-response passes through the proxy as if the card were physically present.

From Research Tool to Weapon

NFCGate (2015 -- TU Darmstadt)

NFCGate was developed at the Technical University of Darmstadt as a legitimate security research tool to study NFC relay attacks. Open-source, academic, designed for controlled lab environments. It demonstrated that NFC relay was practically feasible, not just theoretical.

NGate (2024 -- Czech Republic)

ESET discovered NGate in the Czech Republic -- the first in-the-wild malware built on NFCGate's relay functionality. Attackers used phishing campaigns to trick victims into installing a malicious app, then instructed them to tap their bank card. The relayed NFC data was used for fraudulent ATM withdrawals.

> **[HACK]** NGate was a proof of concept in the criminal underground. It showed that NFCGate's relay architecture could be weaponized at scale with minimal modification.

SuperCard X (2025 -- Chinese-Speaking Groups)

Cleafy Labs documented SuperCard X, a full MaaS (Malware-as-a-Service) platform operated by Chinese-speaking cybercrime groups. SuperCard X professionalized the NFCGate relay attack into a turnkey fraud operation:

* **Recruitment:** Mules hired through Telegram platforms (Huione Guarantee, Xinbi Guarantee, Tudou Guarantee)
* **Tooling:** Custom Android apps for both victim (Reader) and mule (Tapper) sides
* **Infrastructure:** Relay servers managed by the MaaS operator
* **Monetization:** Mules purchase luxury goods (jewelry, electronics) at retail stores using relayed credentials, then resell for cash

The Attack Flow

A typical SuperCard X operation:

1. **Social engineering:** Victim receives an SMS or call impersonating their bank, claiming suspicious activity
2. **App installation:** Victim is directed to install a "security verification" app (the Reader component)
3. **Card tap:** Victim is instructed to tap their credit/debit card to their phone for "verification"
4. **Real-time relay:** NFC data streams to the mule's device in another city or country
5. **POS fraud:** Mule taps their phone at a POS terminal, completing a contactless purchase
6. **Cashout:** Purchased goods are resold or returned for cash

The entire attack -- from social engineering to purchase -- can happen in under 10 minutes.

Scale of Operations

The numbers are staggering:

Metric	Value
NFCGate-infected devices (Russia, Q1 2025)	175,000+
Daily successful attacks (Russia, Q1 2025)	~40
Financial losses (Russia, Q1 2025)	432M rubles (~$5.5M)
NFC attack growth (ESET, H1 2025 vs H2 2024)	35x increase
Huione Guarantee lifetime transaction volume	$27 billion

The Chinese cybercrime ecosystem supporting these operations is massive. Huione Guarantee -- a Telegram-based criminal marketplace that processed over $27 billion in crypto before its nominal shutdown in May 2025 -- served as a one-stop shop for recruiting mules, buying stolen card data, and laundering proceeds through luxury goods resale.

> **[INFO]** Even after Telegram banned Huione Guarantee and Xinbi Guarantee channels, the operations quickly migrated to Tudou Guarantee and other successor platforms. The decentralized, Telegram-based model makes permanent disruption extremely difficult.

The Evolution Continues

The threat landscape is evolving rapidly. F6 (a Russian security firm) documented a "reverse" NFCGate scheme where attackers use CmaxsRAT to intercept SMS notifications and PINs, then initiate and relay NFC transactions to accomplices waiting at ATMs.

Each new iteration adds capabilities:

* **NGate** -- Basic relay, ATM withdrawal focus
* **SuperCard X** -- MaaS platform, POS fraud focus
* **Ghost Tap** -- Enhanced relay with reduced latency
* **RatOn** -- Combines RAT capabilities with NFC relay in a single package

Defense

Defending against NFC relay is challenging because the EMV protocol wasn't designed to verify physical proximity beyond the NFC range itself.

For consumers:

* Never install apps from links received via SMS or phone calls
* Your bank will **never** ask you to tap your card to a third-party app
* Disable NFC when not actively using contactless payments
* Set low limits for contactless transactions

For financial institutions:

* Implement velocity checks for contactless transactions
* Flag geographic anomalies (same card, different cities, short timeframe)
* Monitor for contactless-MSD transactions from EMV-capable cards
* Deploy behavioral biometrics to detect relay latency patterns

For the industry:

* Distance bounding protocols (like the proposed L1RP) can detect relay by measuring round-trip time at the physical layer
* Device attestation to verify the integrity of the NFC-capable device
* Enhanced transaction verification for high-value contactless payments

> **[WARN]** Distance bounding -- measuring the physical round-trip time of NFC communications -- is the only robust technical countermeasure against relay attacks. It's specified in EMVCo standards but not yet widely deployed. Until it is, NFC relay remains fundamentally viable.

The tap that pays for your coffee could be paying for someone else's jewelry. Check your statements.

```
-- tapjack
```

AI BREAKS CRYPTO: DEEP LEARNING SIDE-CHANNEL ATTACKS

by Elena Kostrova (@d1ffeq) — 2026-02-13

tags: side-channel, deep-learning, cryptography, aes, post-quantum, hardware

AI BREAKS CRYPTO: DEEP LEARNING SIDE-CHANNEL ATTACKS

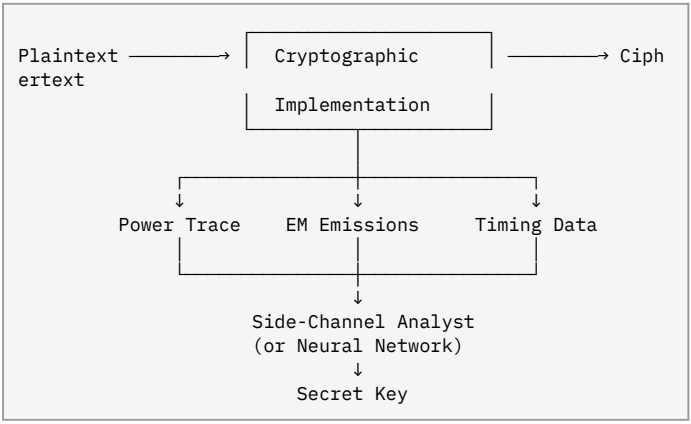
Your encryption algorithm might be mathematically perfect. Its *implementation* is not.

Every processor leaks information when it computes. Power consumption fluctuates as transistors switch. Electromagnetic radiation escapes the chip. Execution timing varies with the data being processed. For decades, side-channel analysts have exploited these physical leakages to extract secret keys -- but it required deep expertise, careful signal processing, and statistical intuition.

Deep learning is automating all of it.

Side-Channel Attacks 101

A side-channel attack extracts secret information by observing the *physical behavior* of a device during cryptographic operations, rather than attacking the mathematics of the algorithm itself.



The most studied technique is **Differential Power Analysis (DPA)**, introduced by Paul Kocher in 1999. DPA works by:

1. Collecting thousands of power traces during encryption operations
2. Formulating hypotheses about key byte values
3. Computing statistical correlations between hypothesized intermediate values and measured power consumption
4. The correct key hypothesis shows the highest correlation

DPA is powerful but has limitations: it requires careful trace alignment, knowledge of the implementation, and struggles against countermeasures like masking and desynchronization.

Enter Deep Learning

Starting around 2018, researchers began applying convolutional neural networks (CNNs) to side-channel analysis -- and the results were striking.

Instead of hand-crafting statistical tests and manually aligning traces, the approach became:

1. **Collect** power/EM traces during cryptographic operations
2. **Label** each trace with the known key byte (profiling phase)
3. **Train** a CNN to predict the key byte from the trace
4. **Attack** new traces from the target device with the trained model

The neural network learns to locate the informative time samples, compensate for noise and misalignment, and extract key-dependent features -- all automatically.

> **[HACK]** The key insight: CNNs are **shift-invariant**. A convolutional filter detects the same pattern regardless of where it appears in the trace. This means deep learning attacks are naturally resistant to desynchronization countermeasures -- one of the most common protections in real hardware.

Attacking AES

The ASCAD (ANSSI SCA Database) has become the standard benchmark for evaluating deep learning side-channel attacks on AES-128. It provides power traces from a protected AES implementation on an ATMega8515 microcontroller, with both fixed-key and variable-key datasets.

Recent results from 2024-2025 research:

Model	Target	Countermeasures	Traces to Recover Key
VGG-16 variant	ASCAD (fixed key)	Masking + desync (50 pts)	~50
VGG-16 variant	ASCAD (fixed key)	Masking + desync (200 pts)	~50
Optimized CNN	ASCAD (variable key)	Masking	~40
HACNN	Custom AES-128	Synchronous	48
CNPP	Custom AES-128	None	<100

For context, traditional DPA might require tens of thousands of traces against unprotected implementations and struggle entirely against masked ones.

A 2025 paper published in ScienceDirect -- "Beyond encryption: How deep learning can break microcontroller security through power analysis" -- demonstrated CNN-based key recovery from STM32 microcontrollers, a platform widely used in IoT and embedded systems. The models achieved successful key extraction with significantly fewer traces than classical techniques.

Breaking Post-Quantum Crypto

Perhaps the most alarming development: deep learning side-channel attacks are being turned against **post-quantum cryptographic implementations**.

The IACR ePrint 2025/1754 survey -- "Machine Learning and Side-Channel Attacks on Post-Quantum Cryptography" -- catalogs attacks against the algorithms selected by NIST's Post-Quantum Cryptography standardization:

*

CRYSTALS-Kyber (ML-KEM): Power analysis attacks on lattice-based key encapsulation. Research demonstrated a 99.98% attack success rate, recovering a 32-byte encapsulation key from **a single power trace** on an STM32F415 microcontroller.

*

CRYSTALS-Dilithium (ML-DSA): Signature scheme vulnerable to EM side-channel analysis targeting the polynomial multiplication operations.

*

Saber KEM: A study by Ngo achieved the first successful deep learning attack on a **masked** Saber implementation using only **16 power traces** -- bypassing the masking countermeasure entirely through end-to-end learning.

> **[WARN]** Post-quantum cryptographic *algorithms* are designed to resist quantum computers. They are not inherently resistant to side-channel attacks. The mathematical security and the implementation security are separate problems, and deep learning is rapidly closing gaps in the latter.

Defending with Deep Learning

The same technology being used to attack can also defend.

A 2025 paper in Nature/Scientific Reports described a hybrid framework using **WGAN-GP (Wasserstein Generative Adversarial Networks with Gradient Penalty)** combined with genetic algorithms to optimize cryptographic protocol implementations against side-channel leakage. The approach:

1. WGAN-GP generates realistic synthetic power traces
2. Genetic algorithms evolve protocol parameters to minimize side-channel leakage
3. The optimized implementation is tested against deep learning attacks

The framework evaluated AES-ECB, AES-GCM, ChaCha20, RSA, and ECC implementations, with AES-GCM showing the best balance of efficiency and side-channel resistance.

Other defensive applications include:

* **Leakage detection:** Training models to identify *where* in the execution trace information leaks, enabling targeted countermeasure placement
* **Countermeasure evaluation:** Using attack models as automated testers for new masking and shuffling schemes
* **Dynamic reconfiguration:** FPGA-based systems that alter their power profile in response to detected analysis attempts

The Hardware Security Implications

The trajectory is clear. Deep learning is:

* **Reducing expertise requirements.** Classical DPA required years of domain knowledge. A CNN model can be trained by anyone with a GPU and a dataset.
* **Defeating countermeasures.** Masking, desynchronization, and noise injection -- the standard defenses -- are being bypassed by models that learn to filter noise and compensate for randomization.
* **Attacking post-quantum schemes.** The algorithms we're deploying to resist quantum computers are vulnerable to the neural networks we already have.
* **Shrinking trace requirements.** From tens of thousands to dozens to single traces. Less data means shorter physical access, smaller attack windows, harder detection.

The endgame isn't theoretical. If a CNN can extract a key from a single power trace captured by an EM probe held near a device, the security model for any hardware cryptographic implementation needs to be reconsidered.

> **[INFO]** Certification bodies like Common Criteria and FIPS are beginning to require deep learning-based evaluations for cryptographic modules. If your hardware was certified before ~2023, it may not have been tested against these attacks.

Your algorithm is sound. Your implementation is an antenna. Train accordingly.

`-- d1ffeq`

XBOW: THE AI THAT BEAT EVERY HACKER ON HACKERONE

by Marcus Oduya (@autoexploit) — 2026-02-13

tags: ai, pentesting, bug-bounty, automation, hackerone

XBOW: THE AI THAT BEAT EVERY HACKER ON HACKERONE

In August 2025, for the first time in bug bounty history, the top-ranked vulnerability researcher on HackerOne's US leaderboard wasn't human.

XBOW -- an autonomous AI pentesting agent built by a team including Diego Jurado Pallarés (himself a top-ranked HackerOne hunter) and Nico Waisman (former CISO of Lyft) -- submitted over 1,000 vulnerability reports in a few months, validated each with a working proof-of-concept exploit, and climbed to the #1 spot on the platform's leaderboard.

The question everyone is asking: is this the beginning of the end for human bug bounty hunters?

What XBOW Is

XBOW is a purpose-built, multi-agent AI system designed specifically for web application penetration testing. Unlike general-purpose LLMs applied to security tasks, XBOW was engineered from the ground up for autonomous vulnerability discovery.

The key architectural decisions:

* **Multi-agent architecture:** Specialized agents collaborate on different aspects of the assessment -- reconnaissance, injection testing, authentication analysis, etc.
* **Deterministic validation:** Findings are verified using traditional security tooling and PoC execution, not LLM judgment. This is critical for maintaining accuracy.
* **Safety layer:** An automatic safety system ensures PoC exploits are harmless -- confirming vulnerabilities without causing damage.
* **Human review pre-submission:** While discovery is fully automated, the security team reviews all findings before they're submitted to HackerOne, per platform policy.

The Numbers

XBOW's performance on HackerOne:

Metric	Value
Total vulnerabilities reported	~1,060
Verified vulnerabilities on profile	285
Time to reach #1 US leaderboard	~3 months
Zero-days discovered (total)	1,000+

On XBOW's own open-source benchmark -- 104 web security challenges sourced from PortSwigger, PentesterLab, and CTF competitions -- the system achieved a **76.9% overall success rate**, with perfect scores in some categories:

Vulnerability Class	Success Rate
SSRF	100%
Security Misconfiguration	100%
Server-Side Template Injection	85%
SQL Injection	83%
Command Injection	75%
Cross-Site Scripting	57%
Blind SQL Injection	0%

The failure cases are informative. XBOW struggles with vulnerabilities requiring timing-based analysis (blind SQLi) and complex payload crafting (certain XSS variants) -- areas where human intuition and creative problem-solving still dominate.

Black Hat 2025

At Black Hat USA 2025 (August 2-7) and DEF CON 33 (August 7-10), XBOW ran live against real HackerOne targets from the conference booth. They discovered dozens of vulnerabilities during the event, producing a working PoC for every single one.

The live demo format was a statement: this isn't a research prototype. It works against production targets, in real-time, at conference WiFi speeds.

> **[HACK]** XBOW's benchmark is open-source on GitHub (`xbow-engineering/validation-benchmarks`). The 104 challenges were kept confidential until release to ensure they hadn't been used in any model training. If you want to evaluate AI pentesting tools, it's the current standard.

The Debate

The security community's reaction has been... complicated.

The optimistic view: XBOW handles the "scan-and-exploit" tier of bug bounty -- the repetitive, pattern-matching work that human hunters do at scale. This frees human researchers to focus on complex logic bugs, architectural vulnerabilities, and creative attack chains that AI can't replicate. The overall volume of vulnerability discovery increases. The internet gets safer.

The pessimistic view: Bug bounty programs are the primary income source for thousands of security researchers worldwide. If AI can submit 1,000 reports in the time a human submits 10, the economic model collapses. Platform leaderboards become AI competitions. New researchers can't compete and never develop skills that would let them tackle the harder problems.

The pragmatic view (CyberScoop's take): "Not yet." XBOW's 0% success rate on blind SQL injection and 57% on XSS shows significant gaps. The most valuable vulnerability classes -- business logic flaws, authentication bypasses, race conditions -- require contextual understanding that current AI lacks. Human hunters aren't obsolete. But the floor is rising.

> **[INFO]** HackerOne has publicly addressed XBOW's participation, noting that their policies allow AI-assisted submissions with human oversight. The platform sees AI as a force multiplier for the security ecosystem, not a threat to human researchers.

What It Means for Defense

XBOW's implications extend beyond bug bounty. If an AI agent can find this many vulnerabilities in web applications, defenders need to reconsider their assessment cadence.

Traditional pentesting follows an annual or quarterly cycle: schedule a firm, wait 35-100 days from scoping to final report, pay $10,000-$35,000+, get a PDF. By the time you read the report, your codebase has changed.

XBOW's commercial product -- **Pentest On-Demand** -- delivers complete pentest results within five business days through a self-service interface. No scoping calls. No kickoff meetings. Reports meet SOC2, ISO 27001, and 40+ compliance frameworks.

The model shift: pentesting as a **continuous, automated process** rather than a periodic, human-led engagement. Ship code on Monday, get pentest results by Friday. Every sprint, not every quarter.

What It Means for Offense

If XBOW can find hundreds of vulnerabilities autonomously, so can attackers. The democratization of offensive capability is accelerating. The barrier to entry for vulnerability discovery is dropping from "years of training" to "API access."

This isn't speculative. The architecture is known, the benchmark is public, and the approach is replicable. The question isn't whether adversaries will build similar systems -- it's whether they already have.

The Future

XBOW's founders are focused on scaling in 2026. The benchmark will evolve. The agent's capabilities will expand. Other companies -- and open-source projects -- are building competing systems.

The security industry is watching a familiar pattern play out at accelerated speed: a capability that once required years of human expertise is being compressed into software that runs in minutes. We've seen it with vulnerability scanning, with malware analysis, with threat intelligence.

Penetration testing is next.

Whether that's cause for celebration or concern depends on which side of the keyboard you're on.

The #1 hacker on HackerOne wrote zero lines of code. Think about what that means for your career -- and your defenses.

`-- autoexploit`

SECURITY NEWS ROUNDUP: 2025 IN REVIEW

by The Editor (@r00t) — 2026-02-13

tags: news, vulnerabilities, industry, year-in-review

SECURITY NEWS ROUNDUP: 2025 IN REVIEW

2025 was the year security stopped being theoretical. Nation-state exploitation timelines compressed to hours. Cryptocurrency theft hit unprecedented scale. AI went from research curiosity to leaderboard champion. And contactless payment fraud went industrial.

No sponsored content. No vendor hype. Just the stories that mattered.

Critical Vulnerabilities

React2Shell (CVE-2025-55182) -- CVSS 10.0

The web's Log4Shell moment. Unauthenticated RCE in React Server Components via unsafe deserialization of the Flight protocol. Discovered November 29, disclosed December 3, exploited within hours. 39% of cloud environments affected. 8.1 million attack sessions recorded by GreyNoise. Over 30 organizations breached.

Read the full analysis in Article #1 of this issue.

Ivanti Connect Secure (CVE-2025-22457) -- CVSS 9.0

Stack-based buffer overflow in Ivanti Connect Secure VPN gateways, exploited by China-nexus UNC5221 before patches were available. Initially assessed as a low-risk bug -- later proven to be a remotely exploitable RCE. Affected thousands of enterprise VPN concentrators.

> **[WARN]** If you run Ivanti Connect Secure, verify you're on a patched version immediately. This was actively exploited as a zero-day before disclosure.

Langflow AI RCE (CVE-2025-3248) -- CVSS 9.8

A code injection vulnerability in Langflow, a popular visual framework for building AI/LLM applications. Unauthenticated attackers could execute arbitrary code on the server. Added to CISA's Known Exploited Vulnerabilities (KEV) catalog. A stark reminder that the rush to deploy AI tooling is creating new attack surfaces faster than they can be secured.

SharePoint ToolShell (CVE-2025-53770) -- CVSS 9.8

Unauthenticated RCE in Microsoft SharePoint through insecure deserialization. Actively exploited against government and financial sector targets.

Yet another deserialization-to-RCE pattern -- 2025 was the year deserialization vulnerabilities went mainstream.

AirBorne -- 23 AirPlay Vulnerabilities

Oligo Security disclosed 23 flaws in Apple's AirPlay protocol, including zero-click wormable RCE (CVE-2025-24252 + CVE-2025-24206). Billions of devices affected, including third-party speakers and car systems that may never receive patches.

Read the full analysis in Article #4 of this issue.

Major Incidents

Bybit Heist: $1.5 Billion Stolen

DPRK's Lazarus Group (TraderTraitor) stole 401,347 ETH from Bybit on February 21 via a supply chain attack on Safe{Wallet}. A developer's laptop was compromised through social engineering, AWS session tokens were hijacked, and malicious JavaScript was injected into the wallet UI. The FBI attributed the attack within five days.

Read the full analysis in Article #2 of this issue.

AISURU 31.4 Tbps DDoS Record

In February 2026, Cloudflare mitigated a 31.4 Tbps distributed denial-of-service attack -- the largest ever recorded. The AISURU botnet, composed of compromised IoT devices, launched the attack against a major gaming platform. The previous record (5.6 Tbps, also by AISURU) had stood for less than a year.

APT42 Deepfake Phishing

Iran's APT42 (Charming Kitten) deployed Google Gemini to generate deepfake video and audio for phishing campaigns targeting journalists, diplomats, and human rights activists. The AI-generated content was used in video calls to impersonate known contacts. Google's Threat Analysis Group documented the campaign and published IOCs.

Threat Trends

EtherHiding: Blockchain C2 Goes Nation-State

DPRK's UNC5342 became the first nation-state actor to adopt EtherHiding -- embedding C2 instructions in Ethereum and BNB Smart Chain smart contracts. The technique makes malware infrastructure effectively untakedownable: no domain to sinkhole, no server to seize, no hosting provider to contact.

Read the full analysis in Article #3 of this issue.

NFC Relay Fraud Explodes

NFC-based payment fraud surged 35x in H1 2025 (ESET telemetry). The weaponization of TU Darmstadt's NFCGate research tool into criminal MaaS platforms (NGate, SuperCard X, Ghost Tap) has industrialized contactless payment fraud. Russian losses alone: 432 million rubles in Q1 2025.

Read the full analysis in Article #6 of this issue.

Rhysida Ransomware: Original Research

c0mmrade's firsthand investigation into a Rhysida ransomware incident led to the discovery of exposed C2 infrastructure, identification of 14 additional victims through SOCKS proxy logs, and coordination with the FBI for forensic acquisition. The initial access vector: CVE-2024-55591 (Fortigate auth bypass).

Read the full analysis in Article #5 and the Phase 2 investigation in Article #10 of this issue.

AI in Security

XBOW Hits #1 on HackerOne

An autonomous AI pentesting agent submitted over 1,000 vulnerability reports and climbed to #1 on HackerOne's US leaderboard in approximately three months. At Black Hat 2025, XBOW demonstrated live vulnerability discovery against production targets. The benchmark (104 challenges) is now open-source.

Read the full analysis in Article #8 of this issue.

AI Breaks Side-Channel Crypto

Deep learning models now recover AES keys from fewer than 50 power traces -- even against masked implementations. Attacks on post-quantum cryptography (CRYSTALS-Kyber, Saber) achieved key recovery from as few as 16 traces, bypassing masking countermeasures through end-to-end learning.

Read the full analysis in Article #7 of this issue.

AI as Attack Surface

The proliferation of LLM-integrated applications created new vulnerability classes: prompt injection, training data extraction, and tool misuse. The Langflow RCE (CVE-2025-3248) exemplified the pattern -- AI tooling deployed rapidly with insufficient security review.

By the Numbers

Metric	Value
Largest crypto theft	$1.5B (Bybit)
Largest DDoS attack	31.4 Tbps (AISURU)
React2Shell attack sessions	8.1 million
AirPlay CVEs disclosed	17
NFC attack growth (H1 2025)	35x
XBOW vulnerabilities submitted	~1,060
Chrome zero-day market price	$3M+
Full iOS chain market price	$5M+

Looking Ahead to 2026

The trends are clear:

* **Exploitation timelines are compressing.** Hours between disclosure and exploitation is the new normal. "Patch Tuesday, Exploit Wednesday" has become "Patch, Exploit Same Day."
* **Supply chain attacks are maturing.** The Bybit heist showed that targeting a developer's laptop is more effective than targeting the platform directly.
* **AI is reshaping both offense and defense.** XBOW on offense, deep learning countermeasure evaluation on defense. The tools are available to both sides.
* **Physical-layer attacks are going mainstream.** NFC relay fraud went from research paper to MaaS platform in under two years. Side-channel attacks are following a similar trajectory.
* **Blockchain infrastructure is being weaponized.** Untakedownable C2, immutable malware hosting, pseudonymous operations. The tools of decentralization serve attackers as well as activists.

Stay informed. Stay paranoid. Stay patched.

See you in Volume 2.

`-- r00t`

XIBALBA

by tainted — 2026-07-07

tags: ansi

"xibalba" by tainted / FiRE · 2026-07-07 — 80×185






tnt[fire]
x i b a l b a . b b s
+o nuskooler
xibalba.l33t.codes:44510

SHURATOS HEAVENLY BBS

by Savage — 2026-03-03

tags: ansi

"shuratos heavenly BBS" by Savage / Fire Graphics · 2026-03-03 — 80×147






Shuratos Heavenly Sphere
Savage Of Fire Graphics 2026

terminull

Volume 1 — 2026-11-11

no tracking // no ads // just text

https://terminull.c0mmrade.com