LayerZero.

LayerZero Labs KelpDAO Incident Report

May 18, 2026

Executive Summary

On April 18, 2026, the KelpDAO rsETH bridge, built on the LayerZero crosschain messaging protocol, was attacked, resulting in the loss of 116,500 rsETH (approximately $292 million). Mandiant, CrowdStrike, and independent security researchers all attribute the attack to DPRK threat actor TraderTraitor also known as UNC4899.

The breach began on March 6, 2026, when an attacker socially engineered a LayerZero Labs developer to harvest session keys, pivot into LayerZero's RPC cloud environment, and poison internal RPC nodes — RPC (Remote Procedure Call) nodes are servers that respond to queries about blockchain state. The attacker patched the running RPC memory with a program that returned correct responses to the LayerZero monitoring tools and tampered RPC responses to the LayerZero Labs DVN (Decentralized Verifier Networks). To further facilitate the attack, the attacker executed a Denial of Service (DoS) attack against an external RPC provider, forcing the LayerZero Labs DVN signing service to rely exclusively on two compromised internal nodes and produce a valid attestation for the forged crosschain message.

The impact was made possible by the affected OApp's single-verifier configuration. Because no second independent DVN was required to attest, the destination contract accepted the single valid attestation and unlocked rsETH. No other OApps, channels or transactions were compromised.

Following the breach, LayerZero Labs responded as follows:

First, a shift in operating stance. LayerZero Labs has historically been un-opinionated toward OApp builders using its infrastructure: whatever configuration the application's delegate chose, the LayerZero Labs DVN would sign for. That stance has changed. The LayerZero Labs DVN now enforces a baseline security configuration on every channel it participates in, and will refuse to sign as the sole required attestor on any channel. The on-chain protocol is unchanged and continues to enforce whatever each application configures; what changed is which configurations LayerZero Labs, as one worker among many, will participate in.

Second, a full rebuild of the operational infrastructure where the compromise lived. The affected cloud environment was replaced rather than patched, redeployed on hardened baselines with no legacy credentials, service accounts, or configurations carried over. Privileged access requires just-in-time elevation with short-lived credentials, multi-person review for IAM changes, and both device and session validation on every administrative request.

Third, LayerZero Labs has worked closely with partners across the ecosystem to review and harden security configurations, work that continues as a standing focus rather than ending at

incident response. We will continue to share updates on these efforts and the broader security posture of LayerZero infrastructure as the work progresses.

LayerZero Labs has been working with law enforcement and retained CrowdStrike and Mandiant, two leading cybersecurity firms, to investigate the incident, provide attribution, elevate LayerZero Lab's security posture, and contribute to this report. LayerZero Labs also retained zeroShadow, which provided corroborating attribution and assisted with token tracking and seizure efforts. LayerZero Labs is sharing these findings in a comprehensive manner so that participants across the ecosystem can understand the realities of these attack vectors and protect against increasingly prevalent state-sponsored attacks

1. How LayerZero Works

1.1 The architectural problem crosschain protocols solve
A blockchain is a fault-tolerant ledger of shared information between a set of nodes. Each node follows the network's consensus algorithm to decide on a canonical ordered list of transactions (the *ledger*). Different blockchains do not inherently have shared knowledge. A **crosschain transport layer** copies information between different blockchains. Crosschain transport layers are implemented by an off-chain service which observes source chain information through one or more **RPC providers** (a server running blockchain client software that responds to queries about chain state), waits for sufficient finality, then relays that information to a destination blockchain where its authenticity is established cryptographically (signed attestation, validity proof, optimistic challenge window).

Most crosschain bridges and interoperability protocols have one shared transport layer defined and configured by the bridge operator. The LayerZero protocol instead allows each application to define and configure their transport layer(s) independently. This means that each application owner can make trade-offs and design decisions based on their own security requirements while eliminating a single point of failure associated with early bridge designs.

The LayerZero protocol is built on three contract layers:

- An immutable **Endpoint** smart contract on each supported chain. Applications call it to send messages, and it delivers verified messages to applications. The Endpoint cannot be upgraded, paused, or modified by anyone, including LayerZero Labs. The Ethereum Endpoint is at `0x1a44076050125825900e736c501f859c50fE728c`. Source: `EndpointV2.sol`.
- A versioned, append-only immutable **Message Library** that handles encoding messages, managing transport layer security configurations, paying off-chain workers, verifying signatures, and committing verified messages to the Endpoint. The current EVM library is called ULN302 (send side, receive side). New Message Libraries can be added but deployed Message Libraries cannot be modified or removed.

- **Workers** are services the **Message Library** pays at send-time to do the actual crosschain work. Two types matter for this document: DVNs (which produce verification attestations and compose the security stack) and Executors (which executes verified messages on the destination-chain). Each application picks which workers to use.

Applications built on top of LayerZero are called **OApps** (Omnichain Applications). KelpDAO's rsETH bridge is an OApp; specifically, an **OFT** (Omnichain Fungible Token), the LayerZero standard for tokens that move between chains.

A unique communication path between two specific OApp contracts is called a **channel**. KelpDAO's rsETH OApp on Ethereum and its counterpart on Arbitrum, taken together, form one channel. The OApp on Ethereum and its counterpart on Unichain are a different channel. Each channel maintains its own message ordering and independent security configuration. The configuration is set per-OApp and per-direction by an account designated as the OApp's **delegate**. For KelpDAO's rsETH OApp, the delegate is the EOA `0x1f7A03b70C5448DFd0a2C5a7865169253c2C769b`. The delegate is responsible for setting the channel's transport policy.

Channels are authenticated on-chain. The **owner** of each OApp registers its counterpart on the other chain by calling `setPeer` on the Endpoint. This registration gates both verification and delivery of messages. Before the Endpoint accepts a DVN-attested message on a new channel, it asks the receiving OApp whether the source matches the registered peer via `allowInitializePath`; without a peer, `verify()` reverts and the payload hash never enters the Endpoint. On every subsequent delivery, the OApp's `lzReceive` re-checks the same condition and rejects any message whose source is not the registered peer.

1.2 DVNs as Composable Transport Layers

The crosschain message verification work is performed by **DVNs** (Decentralized Verifier Networks). A DVN runs off-chain infrastructure to observe events (messages) on a source chain, decide whether they are valid and final, and submit cryptographic attestations of the message contents to the destination blockchain. The destination Message Library accepts a packet only after the channel's configured DVNs have produced matching attestations.

The most common DVN architecture, and the one the **LayerZero Labs DVN** uses, is a multisig signing committee: a set of off-chain servers observe the source chain through RPC providers and produce ECDSA secp256k1 signatures over the canonical hash of the message. The DVN infrastructure submits these signatures to the DVN smart contract which verifies them against a registered list of authorized signers before forwarding the verification to the Message Library. The contract is DVN.sol (with MultiSig.sol handling signature verification). The LayerZero Labs DVN on Ethereum is at 0x589dEDbD617e0CBcB916A9223F4d1300c294236b. Each DVN smart contract is owned by the entity which operates the DVN.

The OApp owner fully controls and owns the security configuration (by composing a security stack) and posture of the application.

1.3 Security Stack Semantics

Each security stack is defined by a set of ***required DVNs***, a set of ***optional DVNs***, and an ***optional DVN threshold***. Required and optional DVNs are represented as lists of unique addresses of DVN contracts.

All required DVNs *must* attest before a message is accepted. If a single required DVN does not attest or attests to different data, the message will be rejected by the protocol.

At a minimum, the number of optional DVNs who attest and agree to the content of a message must match or exceed the optional DVN threshold.

Two specific configurations matter for this post-mortem:

- A **2-of-2** stack: two required DVNs and zero optional DVNs. An attacker must compromise the signers, off-chain infrastructure, or RPC dependencies of both DVNs simultaneously to forge a packet.
- A **1-of-1** stack: one required DVN and no optional DVNs. An attacker can forge a packet by compromising the signer, offchain infrastructure, or RPC dependencies of one DVN.

1.4 The two on-chain roles inside the standard DVN contract

The standard DVN contract has two distinct sets of authorized addresses.

Signers are the addresses whose ECDSA signatures the contract checks when an attestation is submitted. They correspond to the off-chain servers that observe the source chain and produce signatures. The contract only accepts that attestation if the signers are in this list. The signer list can only be modified if a quorum of existing signers sign the change. The signers' private keys are held by the entity operating the DVN. For the LayerZero Labs DVN, those keys are held by LayerZero Labs.

Admins are addresses authorized to call the contract's relay function (`execute`). They submit pre-signed attestations on-chain. The admin only relays signed payloads; it does not produce signed attestations.The worst an attacker with admin access can do is delay messages, redirect collected DVN operator fees, or withdraw DVN operator fees. The admin cannot forge attestations or change the signer set.

Signers can replace the admin at any time through a function called `quorumChangeAdmin`. The contract is deployed without a super-admin role (`_roleAdmin = address(0x0)`), so there is no escape hatch.

1.5 The off-chain operational layer Gasolina and Essence

For the LayerZero Labs DVN, the off-chain side has two pieces, Gasolina and Essence, both operated by LayerZero Labs. These names are internal terminology, used here because the post-mortem refers to an attack on LayerZero Labs infrastructure these services rely upon.

- **Gasolina** is the signer infrastructure: the off-chain servers that read source-chain information through RPC providers, decide an event is real and final, and produce signed attestations with one of the DVN's signer keys. Gasolina is deployed as Infrastructure-as-Code (IaC) on cloud providers or on-prem. Gasolina holds the signer role on the DVN contract.

- **Essence** is the transaction-relay service. It orchestrates the signing process, aggregates signatures from Gasolina up to the DVN's required quorum, and submits them on-chain to the DVN contract via the contract's `execute` function. Essence holds the admin role on the DVN contract.

Essence cannot forge attestations because only Gasolina holds the private keys of the signers. This is enforced on-chain by the DVN contract, and cannot be bypassed by Essence through any interface.

The LayerZero Protocol relies exclusively on the attestations of DVNs to determine (1) the validity of information copied from the source blockchain to the destination blockchain, and (2) the confirmation depth (finality) of the source blockchain.

2. Unpacking the April 18, 2026 Incident

2.1 Incident Summary
On April 18, 2026, the KelpDAO rsETH bridge, built on the LayerZero crosschain messaging protocol, was attacked, resulting in the loss of 116,500 rsETH (~ $292M USD at the time of the attack). The attack was the result of a compromise to the internal LayerZero Labs RPCs and a simultaneous denial of service attack on a third-party RPC used by the sole LayerZero Labs DVN configured by the application owner.

For this incident to occur, the OApp had to be configured by its owner to use a single DVN which would allow a single signed attestation from that DVN to be sufficient to commit a message. In this case, a previous 2-of-2 configuration had been modified by the application owner to a 1-of-1 configuration which used only the LayerZero Labs DVN.

Wallet activity in the lead up to the incident, as well as the behavior that followed, points to these actions being carried out by the Democratic People's Republic of Korea (DPRK); Crowdstrike and Mandiant attribute this attack to DPRK with high confidence based on their forensic investigation. Both parties further assess with medium confidence that the attack is more specifically associated with **UNC4899** (aka TraderTraitor, Jade Sleet, Pressure Chollima).

Additionally, independent security researchers tanuki42 and tayvano attribute the attack to UNC4899. Their analysis, including the de-mixing of the initial funding for the wallet used to carry out the attack, traces back to addresses that have been previously linked to payments for TraderTraitor infrastructure. They found that the on-chain execution was highly deliberate

and consistent with how this group has been observed to operate in previous incidents, in ways that notably contrast with patterns attributed to other DPRK associated sub-groups.

UNC4899 is a DPRK actor assessed with high confidence to be aligned with the Reconnaissance General Bureau. While they possess the capacity to perform espionage activities, their primary focus is state-backed financial gain, which they achieve through the targeting and theft of financial assets within the cryptocurrency industry. Notably, UNC4899 was responsible for the attack on Safe{Wallet} in February 2025 that resulted in the $1.5B Bybit heist.

2.2 Attack Details

On March 6, 2026, a LayerZero Labs developer (Developer1) who was part of a team dedicated to running and maintaining RPCs was socially engineered to clone a malicious GitHub repo which dropped FLATROOF and ROOFDECK malware on Developer1's macOS system. The malware provided remote access to Developer1's computer and enabled the attacker to harvest session keys which were used to access LayerZero's RPC infrastructure. The EDR (Endpoint Detection and Response) running on the device did not detect the malware. LayerZero Labs utilizes Identity and Access Management via SSO and mandatory MFA across all its applications and cloud admin consoles, and these were enforced at the time of intrusion.

Beginning on March 30, 2026, the attacker utilized Developer1's session keys to access LayerZero's GCP and read GitHub environments using VPN services such as ExpressVPN, NordVPN and Mullvad VPN. The attacker gained access to the LayerZero RPC infrastructure, poisoned the `op-geth` running process on two separate GKE clusters in two separate regions by injecting an ELF Position Independent Executable ("PIE") into memory which returned a forged RPC response that manipulated the DVN into attesting to a crosschain message. On April 18, 2026, the attacker performed a Denial of Service (DoS) attack against external RPC providers. Together, these enabled an unlock of 116,500 rsETH on Ethereum; the affected RPC node evaded detection by not returning the forged RPC response to the LayerZero Labs monitoring tools.

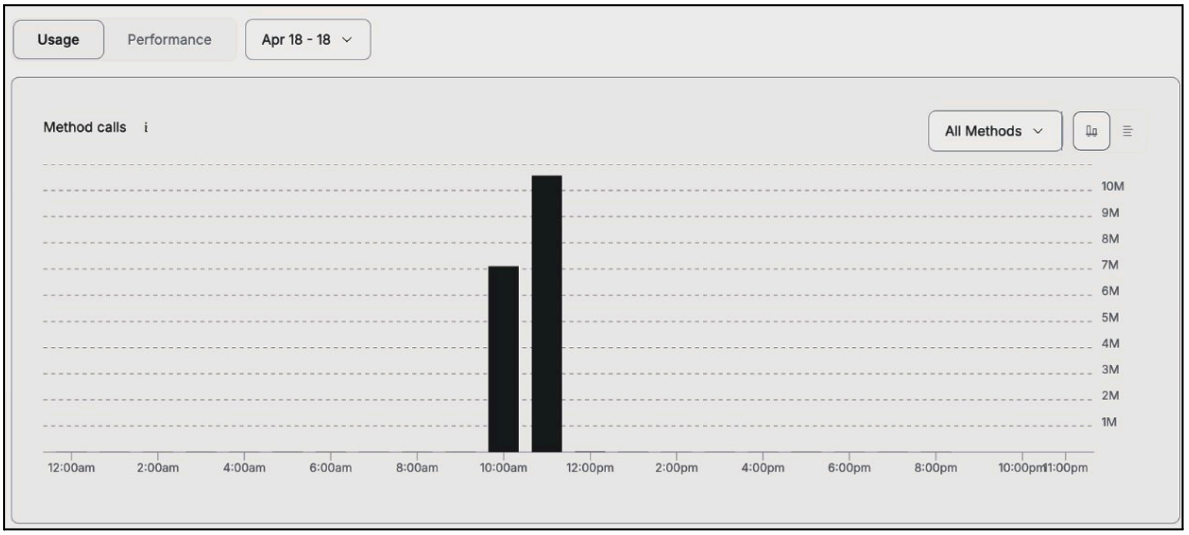

Figure 1 DoS attack on external RPC

On scope of compromise: a comprehensive analysis of LayerZero Labs' GitHub repositories shows no evidence of attempts to implant code, and forensic review found no evidence of compromise to any other LayerZero Labs infrastructure beyond the RPC environment described above. Out of an abundance of caution, LayerZero Labs further hardened its infrastructure across the organization and rotated all relevant keys and secrets.

2.3 Timeline

Date (UTC)	Description
2026-03-06 12:59:03 to 13:06:28	**Social Engineering** - Malicious GitHub repo cloned by Developer1. FLATROOF and ROOFDECK malware dropped on Developer1's Macbook.
2026-03-30 to 2026-04-16	**Cloud Infrastructure Reconnaissance** - GitHub & GCP accessed using Developer1 session keys; Attacker performed reconnaissance and established further persistence.
2026-04-16 to 2026-04-18	**Lateral Movement and RPC Poisoning** - GKE lateral movement; compromised `op-geth` running processes
2026-04-18 16:30	**DoS** - Attacker initiated DoS attack against external RPC providers which forced LayerZero's failover logic to rely exclusively on the poisoned internal RPC nodes.
2026-04-18 17:35	**Exploit** - Attacker unlocked 116,500 rsETH (~$292M USD)

2.4 Malware Used

2.4.1 github[.]com/pi2infra-can-4/gtn-candidate-repo.git (1.37GB)

Defines a malicious terraform provider registry
`registry.hashicorp-aws[.]com/hashicorp/awsbeta`

Downloads and executes `terraform-provider-awsbeta_v1.0.0`, likely when `terraform init` or `terraform plan` is run.

- **SHA256:**`f1df3737c972c5caf070d21a86f132648e6fe1ca07ba541c73eb30f1e4e96390`

Downloads and executes a file from:

- `https[:]//diagnose.hashicorp-aws[.]com/plugins/grpc/v6/schema/metrics/333afe63-c5a2-43f0-b046-7cbaa7797e8a`

2.4.2 FLATROOF

FLATROOF is a backdoor written in Rust and targets an ARM-based macOS environment. The backdoor uses Telegram for its command and control mechanism and is capable of command execution, file upload and download, and data theft.

File Characteristics

File Name	File Type	Size (bytes)	MD5	SHA256
SystemUpdate	MACH-O64	2,346,048	c586e6be49105a23af8f306b560e35e6	6328567511d88fdc2ae0939c5ef17b7a63d2a833881900de018a4f12f4982525

- *C2:*
  - *api.telegram[.]org*
  - *hxxps://io.caiai[.]net/staticscandatav15/upload*
  - *hxxps://technicais.sytes[.]net/statics/v11/a83f7fua937/dg0041*
- *Functionality:*
  - *Command execution, file upload/download, and data theft*
- *File Paths:*
  - *~/Library/com.apple.iTunesCloud/SystemUpdate*
  - */temp/collected_data/**
  - */temp/collected_data.zip*

2.4.3 ROOFDECK

ROOFDECK is a backdoor written in Rust targeting macOS ARM64 environments.

The backdoor utilizes the Nostr protocol for decentralized Command and Control (C2) communications. The backdoor is capable of system reconnaissance, file manipulation, remote shell access, and establishing persistence via Launch Agents.

File Characteristics

File Name	File Type	Size (bytes)	MD5	SHA256
iSync	MACH-O64	4,929,424	c22a69c45ec74af11a9f87f195ebd392	61a110681a70af3dc21634558e12b1c00964f0cf48e90c89

File Name	File Type	Size (bytes)	MD5	SHA256
				896eb0fda1e60b2d

- *C2:*
  - *hxxps://api.nostr[.]watch:443/v1/online*
  - *wss://relay.damus[.]io:443*
  - *wss://nos[.]lol:443*
  - *wss://nostr[.]mom:443*
  - *wss://relay.snort[.]social:443*
  - *wss://offchain[.]pub:443*
  - *wss://relay.nostr[.]band:443*
  - *wss://nostr.oxtr[.]dev:443*
  - *commsouthindia[.]com*
- *Functionality:*
  - *Command execution, file upload, and data theft*
- *File Paths:*
  - *~/Library/com.apple.internal.ck/iSync*
  - *~/Library/Spelling/words-en.dat*
- *Xprotect Behavioral Rule Violations:*
  - *macOS.Browser.Generic*
  - *macOS.Network.Outgoing*
  - *macOS.Persistence.ShellProfileFiles*

2.5 RPC Tampering

While the attacker employed efforts to cover their tracks by deleting certain files associated with their malicious activity, the LayerZero Labs Incident Response (IR) team was able to recover two Core Dump files associated with `op-geth` crashing. OP-Geth (Optimism go Ethereum) is a Go-based implementation of the Optimism Ethereum protocol, serving as a core execution client for running nodes, managing smart contracts, and handling transactions. It allows users to interact with Optimism Ethereum-equivalent networks, verify data, and supports JSON-RPC APIs for developers. A Core Dump is a file containing the recorded state of the working memory of a computer program at a specific moment in time, usually when that program has crashed or terminated abnormally.

Mandiant performed file carving on the Core Dumps to extract malicious code used to poison the `op-geth` running process. Two malicious files were recovered from the Core Dumps (`CARVED1` and `CARVED2`).

2.5.1 `CARVED1` - The Launcher

Name	CARVED1
Role	Launcher

File path on disk	(unknown)
Filename in header	(unknown)
Type	ELF position-independent executable
SHA256[1]	`ce8a08a888457dd6f44041acbef5db011edf5c6cda29fe603f653087d62f4a5f`
Size	810,992

`CARVED1` is a launcher that uses manual techniques to allocate memory for two external shared libraries, copy them into memory, apply relocations, resolve symbols, and execute the constructors from the shared libraries. Two shared library paths are hard-coded into the executable: `/usr/lib/librpcd.so` (the attacker's next stage of malware, `CARVED2`) and `/lib/ld-musl-x86_64.so.1` (the standard C library). The addresses of certain global variables (`rpcm_static_mode`, `rpcm_exe_buf`, `rpcm_exe_len`) from the libraries are resolved and values are written to those variables, indicating a tight coupling between `CARVED1` and the external libraries that it launches. An excerpt of `CARVED1` is shown in Figure 2.

[1] File carved from crash dump, so hash and size of the actual original file may vary.

```
v19 = "/usr/lib/librpcd.so";
v18 = "/lib/ld-musl-x86_64.so.1";
sub_126B("# loading module\n");
sub_126B("# step 1: reading .so\n");
v7 = 0LL;
v8 = 0LL;
*(_QWORD *)&v7 = sub_1363("/usr/lib/librpcd.so", (char *)&v7 + 8);
if ( !(_QWORD)v7 )
{
  sub_126B("# ERROR: cannot read ");
  sub_126B(v19);
  sub_126B(&unk_3000);
  sub_112A(1LL);
}
*((_QWORD *)&v8 + 1) = v7;
sub_126B("# step 2: .so read OK\n");
v17 = sub_145A(&v7);
sub_126B("# step 3: reading libc\n");
v5 = 0LL;
v6 = 0LL;
*(_QWORD *)&v5 = sub_1363(v18, (char *)&v5 + 8);
if ( !(_QWORD)v5 )
{
  v18 = "/lib/libc.musl-x86_64.so.1";
  *(_QWORD *)&v5 = sub_1363("/lib/libc.musl-x86_64.so.1", (char *)&v5 + 8);
}
v20 = 0LL;
if ( (_QWORD)v5 )
{
  *((_QWORD *)&v6 + 1) = v5;
  v20 = sub_145A(&v5);
}
v16 = v17 + v20;
v15 = sub_150A(v17 + v20);
if ( !v15 )
{
  sub_126B("# ERROR: mmap reserve failed\n");
  sub_112A(1LL);
}
sub_126B("# step 4: loading segments\n");
if ( (int)sub_1560(&v7, v15) < 0 )
{
  sub_126B("# ERROR: load .so segments failed\n");
  sub_112A(1LL);
}
sub_126B("# step 5: applying relocs\n");
```

Figure 2 - Excerpt from CARVED1 (Launcher)

`CARVED1` injects the two libraries into the currently running process. As `CARVED1` is a PIE, it can be executed standalone for the attacker's testing. Alternatively, an external tool (unrecovered) could be used to also inject `CARVED1` into the `op-geth` process and then

execute it. Normally, PIEs are built to support ASLR on the main program, but in this case, having the program relocatable makes it easier to inject it into running processes.

2.5.2 CARVED2 -The Monitor

Name	CARVED2
Role	Monitor
File path on disk	/usr/lib/librpcd.so
Filename in header	librpcmon.so
Type	ELF shared object
SHA256[2]	5a90ae020ae40965848b7a344cd6e1439b63ab6c56489d4a16c2cfb97ecb034b
Size	15,424,136

CARVED2 is a monitor that hooks system calls made by its containing process. Mandiant performed memory carving to recover the segments of CARVED2 from a core dump and rebuild plausible ELF headers around them. CARVED2 uses funchook, an open source library (https://github.com/kubo/funchook), to hook the Go standard library functions syscall.Syscall and syscall.Syscall6 in order to intercept calls to read, write, close, and sendto. Intercepting these calls affects reads and writes to regular files and other objects on which these calls are used (pipes, network sockets, etc.). When its containing process makes one of these calls, CARVED2 can view and parse HTTP and JSON.

CARVED2 loads another external library, librpcw.so (“Worker”), and defers the decision of whether to modify HTTP and JSON traffic to that library. Specifically, CARVED2 expects librpcw.so to export an rpc_handler function for CARVED2 to call and determine how to modify the traffic. Mandiant did not recover a copy of librpcw.so.

[2] File carved from crash dump, so hash and size of the actual original file may vary.

2.5.3 How They Fit Together

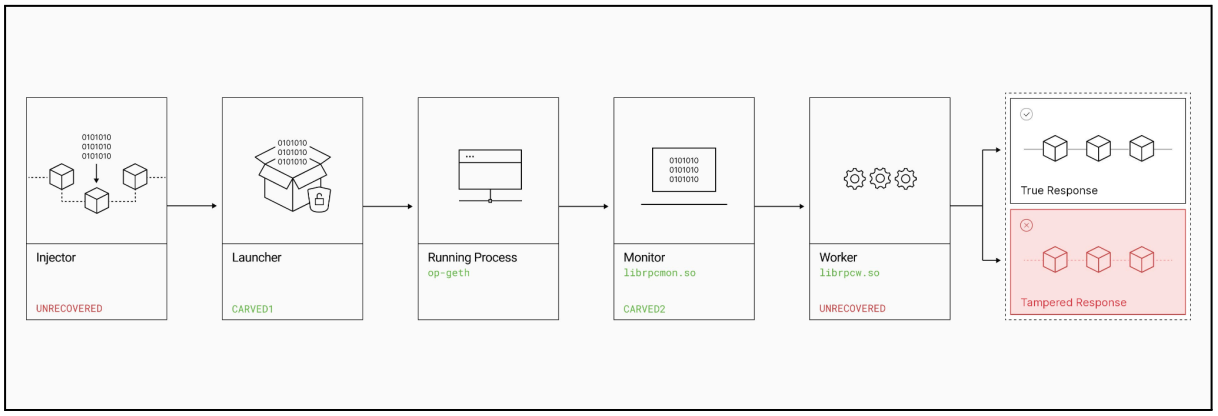

Figure 3 - RPC poisoning

– When `CARVED1` (Launcher) executes (via unrecovered Injector), it loads `CARVED2` (Monitor) into the current process (`op-geth`).

– `CARVED2` (Monitor) uses `funchook` to hook the `read`, `write`, `close`, and `sendto` system calls when the `op-geth` process uses Go's `syscall` functions.

– `CARVED2` (Monitor) uses the hooks to read the HTTP and JSON traffic.

– If the traffic is a request or response for Ethereum, `CARVED2` (Monitor) calls `librpcw.so` (Worker) which reports back whether to modify the request or response.

3. Evaluating Points of Failure

Each layer in the stack outlined below names what it was supposed to do, who was responsible for it, and how it behaved on April 18.

Application Configuration. The 1-of-1 security stack set up by the OApp owner meant the channel had no second DVN whose disagreement would have rejected the forged packet. One additional independent required DVN (2-of-2) would have required the attacker to coordinate simultaneous attacks on multiple independent DVN signers, off-chain infrastructure or RPC dependencies.

Protocol Layer (Endpoint, Message Library, Channel State). The protocol enforced the policy the application's delegate had configured. The receive library's threshold check passed because the configuration required one signed attestation from one specified DVN, and one signed attestation from that address arrived. There was no protocol-level failure.

Signer Keys. The DVN's private signing keys were not compromised. Forensic analysis has found no evidence of key exfiltration. The signatures the contract verified were produced by the legitimately-held keys, signing the hash they were given.

DVN Contract Layer (`DVN.sol`, `MultiSig.sol`). The contract verified the signer-quorum signatures against the registered signer set, which is the only on-chain check it makes. The signatures were valid because they were produced by the registered keys.

Off-Chain RPC Infrastructure. Two internal RPC nodes were patched in memory, external RPC providers were forced offline by DoS, and the signing service was left reading source-chain state through compromised data sources only. The signing service's design assumed its RPC layer was honest, and the assumption was violated by an attacker with sufficient operational capability to violate it.

The same five layers summarized in a table format:

#	Layer	Responsibility	April 18 outcome
1	Number of independent signing services in the required set	Application	Failed: set to one, no second DVN to disagree
2	Source-chain confirmation depth	DVN operator (off-chain)	Not directly bypassed, but rendered ineffective by compromised source data: attacker fabricated a confirmed event but did not bypass the depth gate
3	Signer key custody (HSM, IAM, etc.)	DVN operator	Held: keys not compromised
4	What stands between attacker and signing service (RPC quorum, IAM, payload validation)	DVN operator	Failed: internal nodes patched, external nodes DoS'd into failover
5	On-chain quorum check (signature validity)	Contract	Held: signatures valid against the registered signer set

4. Hardening: What Changed?

In the time since the incident, we've updated practices related to our DVN, RPC infrastructure, and general operating procedures. We've also updated our guidance to applications on how to harden security, and have worked with partners to help them implement these changes.

4.1 LayerZero Labs DVN

These are the constraints the LayerZero Labs DVN now imposes on the configurations it will participate in. They operate above the protocol; the protocol's enforcement of application-configured policy is unchanged.

- **Single-DVN Refusal.** The LayerZero Labs DVN no longer signs attestations for any channel that has it as the only required DVN. The signing service inspects the destination channel's UlnConfig and refuses to sign any messages until the channel is updated.

- **Multi-Source RPC Quorum**. The signing service now requires multiple independent RPC sources quorum when reading security related data from all blockchains it supports, with explicit diversity requirements across providers (internal and external), hosting environments, and geographies.

- **Client Diversity:** LayerZero Labs is developing a new Rust client and will operate its own DVN with the two individual clients being required to attest to protocol messages.

These changes make it so that even if an attacker were to repeat the same off-chain compromise tomorrow, no application using LayerZero Labs as a sole required DVN would have its forged attestation accepted as a complete security stack, because LayerZero Labs would not sign in the first place.

These are above-protocol controls. They do not change the protocol's enforcement of application-configured policy; they limit the configurations the LayerZero Labs DVN will sign for. An OApp that wants to configure a 1-of-1 stack with a different DVN could technically still do so; the protocol will enforce that policy. What it cannot do is have LayerZero Labs as the sole signer of that stack.

4.2 RPC Infrastructure

- **Cloud Environment Rebuild.** The cloud environment hosting the affected RPC infrastructure was fully replaced. All systems in scope were redeployed on new infrastructure with hardened baselines. No credentials or service accounts were carried over from the compromised environment, eliminating the possibility of residual access: backdoors or dormant credentials. In addition, an XDR (Extended Detection and Response) was deployed throughout the cloud environment and the

EDR on all LayerZero Labs was upgraded and configured for a more aggressive enforcement.

- **Access Control and Credential Lifecycle.** Developer credentials no longer hold administrator-equivalent access within the RPC infrastructure. Privileged operations now require just-in-time elevation with per-session authentication. Long-lived service account keys have been replaced with short-lived rotating credentials. Multi-person review is required for IAM changes that would expand a credential's scope of permissions. Alerts have been set up for any deviation.

- **Session Token Binding.** Token validity windows have been shortened materially, reducing the operational lifetime of any token an attacker could harvest. And device context, which was previously evaluated at the point of authentication, is now re-evaluated on every administrative request rather than treated as a one-time check at login.

- **Monitoring.** Real-time alerting is deployed on infrastructure changes, particularly privileged-pod execution and configuration drift. Log-integrity monitoring detects future attempts to degrade the visibility of an active attacker. We're currently developing RPC anomaly detection to cross-validate between independent observers, so divergence between the DVN's view of source-chain state and other observers' views surfaces as an alert rather than passing silently.

- **RPC Reliability Standards.** Internal RPC nodes have minimum requirements for performance, redundancy, and operational separation. Trust assumptions are documented per chain.

4.3 Application-Level Configuration Guidance

These are the configuration changes recommended for any OApp operator. The protocol gives the application sovereignty over its security stack.

- **A minimum of two required DVNs per channel.** With multiple required DVNs configured an attacker must compromise the off-chain infrastructure of multiple independent DVNs simultaneously to forge a packet.

- **Pin libraries and configuration explicitly.** Calling `setConfig` with the chosen values, even when the chosen values match the current default, makes the configuration durable against future default changes and makes the application's chosen policy auditable on-chain as a deliberate decision.

- **Multisig delegate / owner.** Configuration changes (DVN selection, library version, ownership transfer, peering, etc.) peering should require a multisig with multiple

independent signers. A single compromised delegate or owner key should never be sufficient to weaken an OApp's security configuration.

- **Confirmation thresholds aligned to source-chain finality.** Higher confirmation depths make re-org attacks and short-window data manipulation harder. The threshold should reflect the security characteristics of the source chain.

4.4 Protocol defaults

LayerZero Labs will apply changes to the protocol defaults to ensure all channels use no less than a 3-of-3 DVN configuration. More details on these changes will be shared in the coming days.

5. Looking Ahead

5.1 LayerZero Console

LayerZero Labs is developing a Console product that will include in-depth monitoring and analytics for asset issuers, chain operators, DVN operators, and risk managers. Real-time monitoring will consist of configuration checking across libraries, DVN setups, rate limits, versioning, changelog and more.

5.2 DVN Ecosystem

LayerZero Labs has always encouraged applications to operate their own DVNs, which many do today. We will work with 3rd party DVNs to further harden the ecosystem for application use, from enforcing RPC diversity and quorum, to helping stand up redundant global infrastructure, to developing net-new clients that enable applications to have more diversity as part of their security posture.

5.3 A more opinionated LayerZero Labs

While the protocol remains unchanged, we are changing how we operate as a service provider. LayerZero Labs' DVN will no longer act as a configuration agnostic signing service. Our DVN will now refuse to be the sole signer (1-of-1) for any channel. We will only provide attestation for applications that maintain a multi-DVN stack and will enforce a policy for blockchains on providing multi-RPC infrastructure. Listing of OFT assets on first party offerings, like the Value Transfer API and Stargate Finance, will require meeting these requirements.