Zellic

Prepared for

LayerZero Labs

Prepared by

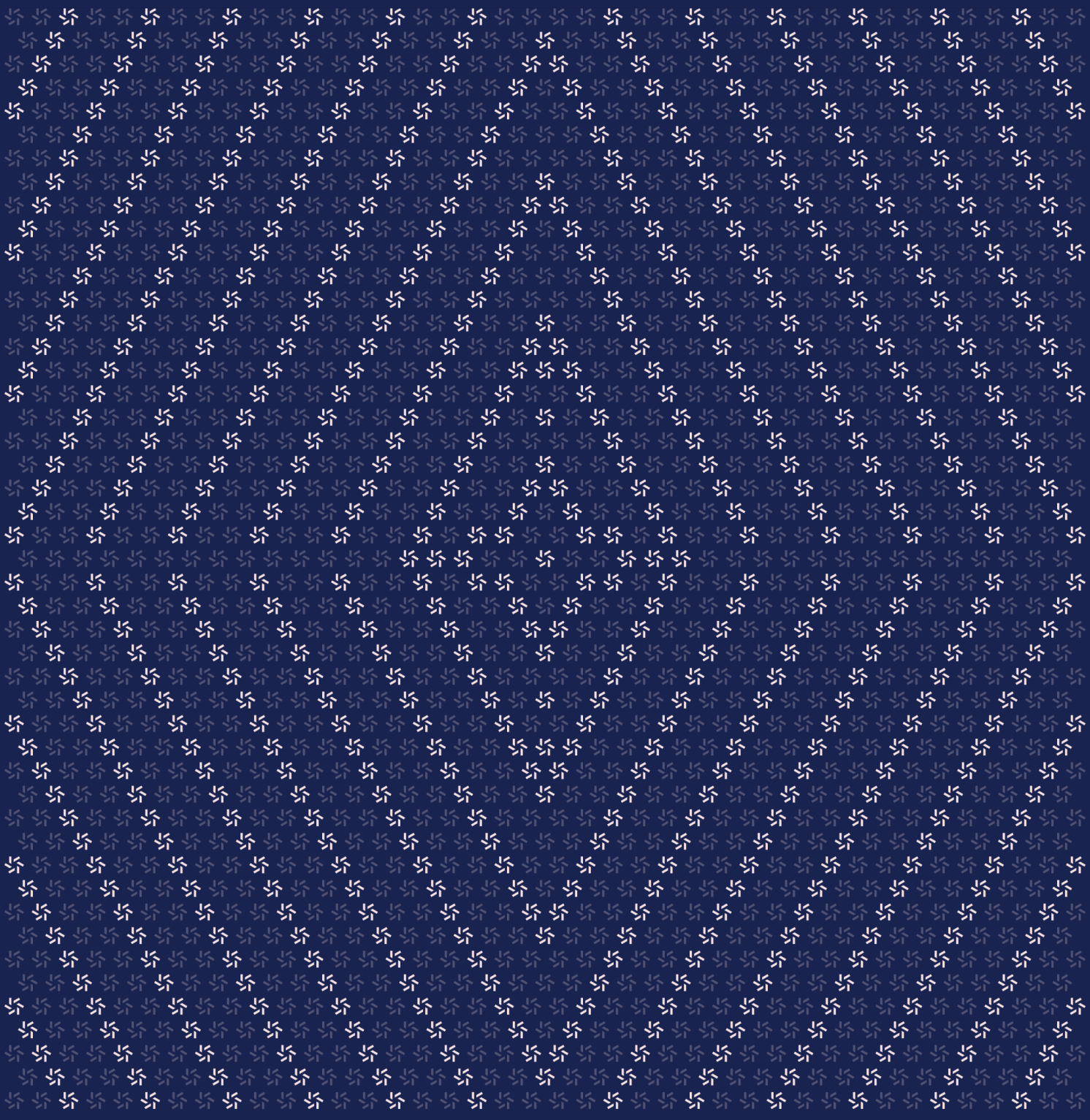
Jasraj Bedi

Zellic

March 14, 2024

DVN

Smart Contract Security Assessment



Contents

About Zellic **4**

1. Overview **4**

1.1. Executive Summary 5

1.2. Goals of the Assessment 5

1.3. Non-goals and Limitations 5

1.4. Results 5

2. Introduction **6**

2.1. About DVN 7

2.2. Methodology 7

2.3. Scope 9

2.4. Project Overview 9

2.5. Project Timeline 10

3. Detailed Findings **10**

3.1. Arbitrary message forging when supporting new chains 11

3.2. Centralization risk with modifiable router 14

4. Discussion **14**

4.1. Assessment Focus 15

5. Assessment Results **15**

5.1. Disclaimer 16

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for LayerZero Labs from March 10th to March 14th, 2024. During this engagement, Zellic reviewed DVN's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Is it possible for the DVN admin to modify in-flight messages?
- Is it possible for the DVN admin send fake messages?
- Is it possible for the DVN admin to forge the source eid, to deliver arbitrary messages on behalf of other chains?

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- The general connectivity, reliability and functionality of the delivery mechanism.
- Centralization risks that can only lead to DoS. It is a known risk factor that the DVN admin can cause DoS

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

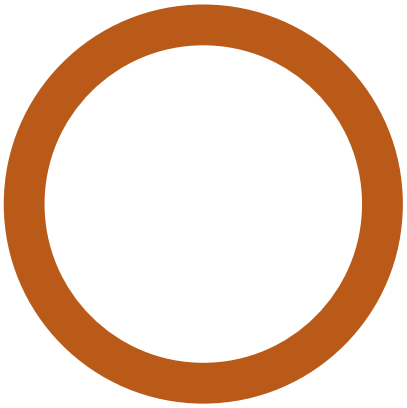
1.4. Results

During our assessment on the scoped DVN contracts, we discovered two findings, both of which were high impact.

Additionally, Zellic recorded its notes and observations from the assessment for LayerZero Labs's benefit in the Discussion section (4. ↗) at the end of the document.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	2
Medium	0
Low	0
Informational	0



2. Introduction

2.1. About DVN

LayerZero Labs contributed the following description of DVN:

> LayerZero is an interoperability protocol that connects blockchains (50+ and counting), allowing developers to build seamless omnichain applications, tokens, and experiences

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and

Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (4. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved a review of the following targets:

DVN Contracts

Repository	https://github.com/LayerZero-Labs/monorepo ↗
Version	monorepo: `8feea92f6d7b89b269be4dfc5f0c0a2eb80d55ed`
Programs	• evm/messagelib/contracts/uln/dvn/adapters/axelar/AxelarDVNAdapter.sol • evm/messagelib/contracts/uln/dvn/adapters/CCIP/CCIPDVNAdapter.sol
Type	Solidity
Platform	EVM-compatible

2.4. Project Overview

Zellic was contracted to perform a security assessment with one consultant for a total of 0.8 person-weeks. The assessment was conducted over the course of 1 calendar week.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald
Engagement Manager
chad@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Jasraj Bedi
CTO
jazzy@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

March 10, 2024	Start of primary review period
March 14, 2024	End of primary review period

3. Detailed Findings

3.1. Arbitrary message forging when supporting new chains

Target	CCIPDVNAdapter, AxelarDVNAdapter		
Category	Business Logic	**Severity**	High
Likelihood	Low	**Impact**	High

Description

One of the main properties of the system is that the admin should not be able to forge messages for `eid`'s that are already configured on the DVN. This can be bypassed by abusing `setDstConfig`.

```
function setDstConfig(DstConfigParam[] calldata _params)
    external onlyRole(ADMIN_ROLE) {
    for (uint256 i = 0; i < _params.length; i++) {
        DstConfigParam calldata param = _params[i];

        // set once per chainName
        // only one adapter per dvn that services both endpoint v1 and v2
        // we standardize the eid stored here with mod 30000
        if (srcConfig[param.chainName].eid == 0) {
            srcConfig[param.chainName].eid = param.eid % 30000;
            srcConfig[param.chainName].peer = param.peer;
        }
        // set once per eid (v1/v2 eid)
        if (bytes(dstConfig[param.eid].chainName).length == 0) {
            dstConfig[param.eid].chainName = param.chainName;
            dstConfig[param.eid].peer = param.peer;
        }

        dstConfig[param.eid].multiplierBps = param.multiplierBps;
        dstConfig[param.eid].nativeGasFee = param.nativeGasFee;
    }
```

This function is mainly called by the admin to add support for new chains and it doesn't allow overwriting `dstConfig` if the `param.eid` already exists. This does not apply to `srcConfig`, where a new `param.chainName` can be configured with an already existing `eid`.

Two main checks exist when verifying the authenticity of the receiving messages:

- `_assetPeer`
- `_decodeAndVerify`

```
function _execute(
    string calldata _sourceChain,
    string calldata _sourceAddress,
    bytes calldata _payload
) internal override {
    SrcConfig memory config = srcConfig[_sourceChain];

    // assert peer is the same as the source chain
    _assertPeer(_sourceChain, _sourceAddress, config.peer);

    _decodeAndVerify(config.eid, _payload);
}

function _assertPeer(string memory _sourceChain, string memory _sourceAddress,
    string memory peer) private pure {
    if (keccak256(bytes(_sourceAddress)) != keccak256(bytes(peer))) {
        revert AxelarDVNAdapter_UntrustedPeer(_sourceChain, _sourceAddress);
    }
}

...

    function _decodeAndVerify(uint32 _srcEid, bytes calldata _payload)
    internal {
    require((DVNAdapterMessageCodec.srcEid(_payload) % 30000) == _srcEid,
    "DVNAdapterBase: invalid srcEid");

    (address receiveLib, bytes memory packetHeader, bytes32 payloadHash)
    = DVNAdapterMessageCodec.decode(_payload);

    IReceiveUln(receiveLib).verify(packetHeader, payloadHash,
    MAX_CONFIRMATIONS);
}
```

`_assertPeer` can be bypassed by just using a peer that is controllable on the destination chain (and admin can set in `setDstConfig`). As we are able to control the `eid` in `srcConfig`, the `_decodeAndVerify` check is also bypassed.

Impact

This allows the admin to fake messages for any configured eid the DVN is used for. As an example:

- Assume LayerZero supports chain A (1), B (2) and C (3) and Axelar supports B, C and D.
- LayerZero has now added support for chain D (4).

- The DVN Admin deploys a malicious contract (at address 0xdeadbeef) on chain D where it can control the full payload
- Admin called `setDstConfig` with `chainName: D, eid: 1`
- Now the admin can send arbitrary messages from chain D using Axelar and the DVN will assume it originated from eid 1. All applications that trust the DVN to deliver eid 1 messages can now be delivered forged messages

Recommendations

We recommend the team to prevent setting `srcConfig.eid` if the `eid` already exists.

Remediation

This issue has been acknowledged by LayerZero Labs, and a fix was implemented in commit e34c204a ↗.

3.2. Centralization risk with modifiable router

Target	CCIPDVNAdapter		
Category	Business Logic	**Severity**	High
Likelihood	Low	**Impact**	High

Description

The `setRouter` function can be used by the admin to modify the CCIP router. By setting the router to a malicious contract, it is possible to send arbitrary messages to the DVN to deliver.

Impact

This leads to complete message forgery as it is possible to deliver arbitrary messages.

Recommendations

We recommend making the `router` immutable

Remediation

This issue has been acknowledged by LayerZero Labs, and a fix was implemented in commit `835fb593` ↗.

4. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

4.1. Assessment Focus

The main focus of the Assessment was to ensure the privlidged admin role is not able to forge/send arbitrary messages to the applications supporting it. It is also a known risk that the admin is able to DoS the message delivery.

The general connectivity and functionality issues between Axelar/CCIP and LayerZero were not a part of the assessment focus, but the LayerZero team has done extensive integration test to ensure messages are being delivered and handled properly.

5. Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped DVN contracts, we discovered two findings, both of which were high impact. LayerZero Labs acknowledged all findings and implemented fixes.

5.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.