// Private Solana Program Security Assessment 05.08.2024 - 05.24.2024

Solana Endpoint

LayerZero

HALBORN

Solana Endpoint - LayerZero

Prepared by: HALBORN

Last Updated 05/29/2024

Date of Engagement by: May 8th, 2024 - May 24th, 2024

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
4	0	0	0	2	2

TABLE OF CONTENTS

1. Introduction

The `LayerZero Endpoint`, implemented as an immutable open-source smart contract and deployed in one or more instances per chain, provides a stable application-facing interface, the abstraction of a lossless network channel with exactly once guaranteed delivery, and manages OApp Security Stacks.

The LayerZero team engaged Halborn to conduct a security assessment of their Solana programs beginning on 5/8/2024 and ending on 5/24/2024. The security assessment was scoped to the program code provided in the GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

2. Assessment Summary

Halborn was provided 2 weeks for the engagement and assigned one full-time security engineer to review the security of the programs in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the programs.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn did not identify any significant security risks. However, some improvements were highlighted that were accepted and acknowledged by the `LayerZero team`.

3. Test Approach And Methodology

Halborn performed a combination of manual review and security testing based on scripts to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and programs logic/connectivity/functions along with state changes
- Manual testing by custom scripts.

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: monorepo

(b) Assessed Commit ID: aa8cd2e

(c) Items in scope:

- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/state/mod.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/state/compose_message.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/state/message_lib.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/state/endpoint.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/state/messaging_channel.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/errors.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/mod.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/init_verify.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/mod.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/transfer_admin.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/set_default_receive_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/set_default_send_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/init_default_send_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/register_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/set_lz_token.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/set_default_receive_library_timeout.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/init_default_receive_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/init_endpoint.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/admin/withdraw_rent.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/lz_compose_alert.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/init_nonce.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/mod.rs

- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/clear.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/skip.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/send.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/burn.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/init_receive_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/set_delegate.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/clear_compose.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/send_compose.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/set_send_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/set_receive_library_timeout.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/init_send_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/set_receive_library.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/lz_receive_alert.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/init_config.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/set_config.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/nilify.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/register_oapp.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/oapp/quote.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/instructions/verify.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/events.rs
- monorepo/packages/layerzero-v2/solana/programs/endpoint/src/lib.rs

Out-of-Scope: Third party dependencies., Economic attacks., Integration with other LayerZero programs in the repository.

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
LIMITED INTEROPERABILITY	Low	RISK ACCEPTED
SINGLE STEP OWNERSHIP TRANSFER PROCESS	Low	RISK ACCEPTED
REDUNDANT ACCOUNT STATE DATA	Informational	ACKNOWLEDGED
POTENTIALLY WEAK CONSTRAINT	Informational	ACKNOWLEDGED

7. FINDINGS & TECH DETAILS

7.1 LIMITED INTEROPERABILITY

// LOW

Description

The Solana runtime implements some basic security protection measures to prevent malicious actors to abuse reentrancy bugs. To do so, the runtime limits the call stack depth to 5.
The program in scope facilitates the functionality of one side of a cross-chain bridge. Bridge architecture assumes some form of onchain->offchain data flow, typically implemented with events or log messages. Because log messages in Solana are not a part of consensus and by default are capped at 10kB per transaction, some risk exists that some on-chain event won't be communicated to offchain listeners. To mitigate that risk, the LayerZero team opted in for CPI-based log messages as defined by the Anchor framework. While in some way more secure than purely log-message-based events, this approach has two main limitations:

- CPI size is restricted to about 1.2kB which significantly limits the size of outbound messages and, in consequence, the interoperability with other chains
- Because the call stack depth tracking starts with the EOA originating the transaction, there are only 3 more levels available, which limits the interoperability with other programs.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (2.5)

Recommendation

Instead of relying on unsafe log messages or too restricting and not scalable Anchor `emit!` CPI log, consider storing log message contents in dedicated accounts at deterministic addresses.
For implementation-level ideas, refer to the Circle's Solana CCTP dev documentation.

Remediation Plan

RISK ACCEPTED: The **LayerZero team** accepted the risk of this finding.

7.2 SINGLE STEP OWNERSHIP TRANSFER PROCESS

// LOW

Description

The `transfer_admin` function does not validate that the `params.admin` is valid.
This parameter corresponds to the new admin address, which holds critical privileges for managing the `endpoint` program. If the admin is set to an incorrect address, admin privileges would be disabled with no way to revert this, as admin privileges require a signature for execution.
`transfer_admin.rs`
[data-language="Rust"]
[data-lines="13"]

```
pub struct TransferAdmin<'info> {
    pub admin: Signer<'info>,
    #[account(mut, has_one = admin, seeds = [ENDPOINT_SEED], bump =
endpoint.bump)]
    pub endpoint: Account<'info, EndpointSettings>,
}

impl TransferAdmin<'_> {
    pub fn apply(ctx: &mut Context<TransferAdmin>, params:
&TransferAdminParams) -> Result<()> {
        ctx.accounts.endpoint.admin = params.admin;
        emit_cpi!(AdminTransferredEvent { new_admin: params.admin });
        Ok(())
    }
}
```

BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:C/D:N/Y:N (2.0)

Recommendation

The recommended way to implement a `transfer_authority` functionality is to use a two-step ownership transfer pattern.
First, implement a function that designates a "new owner candidate." The ownership is not transferred until the new owner "accepts" it, by sending a signed transaction.
This method prevents the assignment of a 0x0 owner and ensures that the new owner explicitly accepts the ownership transfer.

Remediation Plan

RISK ACCEPTED: The **LayerZero team** accepted the risk of this finding.

7.3 REDUNDANT ACCOUNT STATE DATA

// INFORMATIONAL

Description

The program makes have use of Program Derived Addresses to as a way of authenticating and authorizing data and entities.
Program Derived Addresses are generated from a collection of seeds and a `u8` "bump". While there are potentially up to 123 bumps valid per a given collection of seeds, the Anchor framework uses the first available as default. Because the program does not allow the users to specify their own bumps, default bump values are used all over the place, making storing them in blockchain state redundant.
Persisting rather than computing those bumps increases transaction costs for both the protocol governance and its users.

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Remove `bump`s from all account definitions across the codebase and modify the logic so it computes the default bump rather than storing it.

Remediation Plan

ACKNOWLEDGED: The **LayerZero team** acknowledged this finding.

7.4 POTENTIALLY WEAK CONSTRAINT

// INFORMATIONAL

Description

The `MessageLibType` enum has three variants-send, receive, and send and receive. Libraries are checked **not** to have some specific undesired capabilities instead of being checked to have the desired capabilities. This may potentially lead to registering libraries with incorrect types still if new types are added and the constraint is not updated.

`init_default_receive_library.rs`

[data-language="Rust"]

[data-lines="22"]

```
19 | #[account(
20 |         seeds = [MESSAGE_LIB_SEED, &params.new_lib.to_bytes()],
21 |         bump = message_lib_info.bump,
22 |         constraint = message_lib_info.message_lib_type !=
23 | MessageLibType::Send @LayerZeroError::OnlyReceiveLib
24 |     )]
      pub message_lib_info: Account<'info, MessageLibInfo>,
```

[data-language="Rust"]

```
pub enum MessageLibType {
    Send,
    Receive,
    SendAndReceive,
}
```

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

To increase security, consider switching from negative validation to positive validation. On the implementation level, this can be facilitated efficiently with e.g. a bitmask.

Remediation Plan

ACKNOWLEDGED: The **LayerZero team** acknowledged this finding.

STATIC ANALYSIS REPORT

Description:
Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was `cargo audit`, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in `https://crates.io` are stored in a repository named The RustSec Advisory Database. `cargo audit` is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

Cargo Audit Results

ID	CRATE	DESCCRIPTION
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on `ed255109-dalek`

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.