SALUS SECURITY

DEC 2025

CODE SECURITY ASSESSMENT

GATE

Overview

Project Summary

- Name: GateChain - Evm cross chain
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/gatechain/perps
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	GateChain - Evm cross chain
Version	v2
Type	Solidity
Dates	Dec 08 2025
Logs	Dec 01 2025; Dec 08 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	3
Total informational issues	5
Total	9

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Medium	Centralization	Acknowledged
2	Invalid relay option can silently drop cross-chain withdrawals	Low	Business Logic	Acknowledged
3	Implementation contract lacks _disableInitializers()	Low	Business Logic	Resolved
4	Third-party dependencies	Low	Dependency	Acknowledged
5	Missing events for functions that change critical state	Informational	Logging	Acknowledged
6	Cross-chain managers lack input validation and rely on trusted external components	Informational	Data Validation	Acknowledged
7	Mapping declaration lacks named parameters for improved readability	Informational	Code Quality	Acknowledged
8	Redundant Code	Informational	Redundancy	Acknowledged
9	Use of floating pragma	Informational	Configuration	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk

Severity: Medium	Category: Centralization

Target:
- evm-cross-chain/VaultCrossChainManagerUpgradeable.sol
- evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol
- evm-cross-chain/CrossChainRelayV2.sol

Description

In the `CrossChainRelayV2` contract, there exists one privileged role, `Owner`. The owner has the authority to some key functions such as `withdrawNativeToken()` and `withdrawToken()`.If these roles' private keys are compromised, an attacker could trigger this function to drain all funds.

evm-cross-chain/CrossChainRelayV2.sol:L329-L342

```
function withdrawNativeToken(address payable to, uint256 amount) external onlyOwner {
    require(address(this).balance >= amount, "Insufficient balance");
    (bool success, ) = to.call{ value: amount }("");
    require(success, "Transfer failed");
}
function withdrawToken(address token, address to, uint256 amount) external onlyOwner {
    IERC20(token).safeTransfer(to, amount);
}
```

In the `VaultCrossChainManagerUpgradeable` contract and the `LedgerCrossChainManagerUpgradeable` contract , there exists one privileged role, `Owner`. The owner has the authority to upgrade the implementation contract. If these roles' private keys are compromised, an attacker could set a malicious contract by calling `upgradeTo()`.

evm-cross-chain/VaultCrossChainManagerUpgradeable.sol:90-L93

```
function upgradeTo(address newImplementation) public onlyOwner onlyProxy {
    // Upgrade to new implementation and call empty initialization data
    upgradeToAndCall(newImplementation, new bytes(0));
}
```

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team. The team states that this is a Business requirement.

2. Invalid relay option can silently drop cross-chain withdrawals

Severity: Low	Category: Configuration
Target: - evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol	

Description

The `_sendMessage()` selects the relay implementation based on message.option, but it has no other branch. If message.option is any value other than `LayerZeroV1` or `LayerZeroV2`, the function performs no external call and does not revert.

evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol:L397-L406

```
function _sendMessage(OrderlyCrossChainMessage.MessageV1 memory message, bytes memory
payload) internal {
    // Select appropriate sending method based on relay option
    if (message.option == uint8(OrderlyCrossChainMessage.CrossChainOption.LayerZeroV1)) {
        // Use LayerZeroV1 to send message
        crossChainRelay.sendMessage(message, payload);
    } else if (message.option ==
uint8(OrderlyCrossChainMessage.CrossChainOption.LayerZeroV2)) {
        // Use LayerZeroV2 to send message
        crossChainRelayV2.sendMessage(message, payload);
    }
}
```

The `message.option` is sourced from the owner-configurable mapping `ccRelayOption[data.chainId]` in `withdraw()`, `withdraw2Contract()`, and `sendTestWithdraw()`. Since `setCCRelayOption()` does not validate the input range, a misconfiguration (e.g., setting the option to 2) would cause `_sendMessage()` to return successfully without actually sending any cross-chain message.

As a result, cross-chain withdrawals (and test withdrawals) can be silently dropped. The ledger-side flow completes locally while the vault-side action never occurs, leading to stuck withdrawals and cross-chain state divergence that is difficult to detect on-chain.

Recommendation

Add a reverting else branch in `_sendMessage()` and validate `_ccRelayOption <= 1` in `setCCRelayOption()`.

Status

This issue has been acknowledged by the team.

3. Implementation contract lacks _disableInitializers()

Severity: Low	Category: Configuration

Target:
- evm-cross-chain/VaultCrossChainManagerUpgradeable.sol
- evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol
- evm-cross-chain/CrossChainRelayV2.sol

Description

The `OApp`, `VaultCrossChainManagerUpgradeable` and `LedgerCrossChainManagerUpgradeable` contract does not call OpenZeppelin's `_disableInitializers()` in its constructor.

As a result, anyone can send a direct transaction to the implementation address (not the proxy) and invoke `initialize()`, gaining the `owner` role within the implementation's own storage context.

Recommendation

Add a constructor that locks the implementation:

```
constructor() {
    _disableInitializers();
}
```

Status

The team has resolved this issue in commit 1431ea7.

4. Third-party dependencies

Severity: Low	Category: Dependency
Target: - evm-cross-chain/CrossChainRelayV2.sol	

Description

The `CrossChainRelayV2` contracts rely on the `OAppUpgradeable` contract to enable the basic functionality. The current audit treats third-party entities as black boxes and assumes they are working correctly. However, in reality, third parties could be compromised, resulting in the disruption of token functionalities.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to regularly monitor the statuses of third parties to reduce the impacts when they are not functioning properly.

Status

This issue has been acknowledged by the team. The team states that this is a trusted dependency.

2.3 Informational Findings

5. Missing events for functions that change critical state

Severity: Informational	Category: Logging

Target:
- evm-cross-chain/VaultCrossChainManagerUpgradeable.sol
- evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol
- evm-cross-chain/CrossChainRelayV2.sol

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the `VaultCrossChainManagerUpgradeable`contract, events are lacking in the privileged functions (e.g. `setChainId()`, `setCrossChainRelay()`,`setCrossChainRelayV2()` and `setLedgerCrossChainManager()`).

In the `LedgerCrossChainManagerUpgradeable` contract, events are lacking in the privileged functions (e.g. `setChainId()`, `setCrossChainRelay()`,`setCrossChainRelayV2()` , `setOperatorManager`()` `setVaultCrossChainManager()` and `setTokenDecimal()`).

Recommendation

It is recommended to emit events for critical state changes.

Status

This issue has been acknowledged by the team.

6. Cross-chain managers lack input validation and rely on trusted external components

Severity: Informational	Category: Data Validation
Target: - evm-cross-chain/VaultCrossChainManagerUpgradeable.sol - evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol	

Description

The `LedgerCrossChainManagerUpgradeable` and `VaultCrossChainManagerUpgradeable` perform minimal validation on inbound and outbound cross-chain data, implicitly trusting the correctness of the relay layer and the `Vault/Ledger` contracts.

The system's correctness heavily depends on the integrity and correctness of external contracts and operators.

Recommendation

Ensure that all external contracts interacting with the cross-chain managers (relay, Vault, Ledger) are trusted, or adding defensive validation inside both contracts.

Status

This issue has been acknowledged by the team.

7. Mapping declaration lacks named parameters for improved readability

Severity: Informational	Category: Code Quality

Target:
- evm-cross-chain/VaultCrossChainManagerUpgradeable.sol
- evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol
- evm-cross-chain/CrossChainRelayV2.sol

Description

Since Solidity version 0.8.18, the language allows developers to add named key and value identifiers in mapping declarations using the syntax:

```
mapping(KeyType keyName => ValueType valueName)
```

This enhances code readability and clarifies how a mapping is intended to be used. In those contracts, the mappings are declared without named parameters, making the purpose of the key and value less obvious upon inspection.

evm-cross-chain/CrossChainRelayV2.sol::L15-L17,L84-L87

```
mapping(uint32 => mapping(bytes32 => mapping(uint64 => bool))) public nonce;
mapping(uint8 => LzOption) public lzOptions;
mapping(uint256 => uint32) public chainId2Eid;
mapping(uint32 => uint256) public eid2ChainId;
```

evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol:L42-L54

```
mapping(uint256 => address) public vaultCrossChainManagers;
mapping(bytes32 => mapping(uint256 => uint128)) public tokenDecimalMapping;
mapping(uint256 => uint8) public ccRelayOption;
mapping(address => bool) public enabledRelays;
```

evm-cross-chain/VaultCrossChainManagerUpgradeable.sol:L36-L43

```
mapping(uint256 => address) public ledgerCrossChainManagers;
mapping(address => bool) public enabledRelays;
```

Recommendation

Consider updating the mapping declaration to explicitly name the key and value identifiers, enhancing readability and making the mapping's purpose clearer.

Status

This issue has been acknowledged by the team.

8. Redundant Code

Severity: Informational	Category: Redundancy
Target: - evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol	

Description

Unused code should be removed before deploying the contract to mainnet. We have identified the following functions are not being utilized:

evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol:L419-L461

```
function sendTestWithdraw(uint256 dstChainId) external onlyOwner {}
```

Recommendation

Consider removing the redundant code.

Status

This issue has been acknowledged by the team.

9. Use of floating pragma

Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.18;
```

For example, the `VaultCrossChainManagerUpgradeable` uses a floating compiler version ^0.8.18.

Using a floating pragma ^0.8.`8 statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 5a478c2:

File	SHA-1 hash
evm-cross-chain/CrossChainRelayV2.sol	b15f537fe9ed9189712fa767437cec905ff9caeb
evm-cross-chain/LedgerCrossChainManagerUpgradeable.sol	e63b4eea85f22620fa8fb929ee839a48e6c68a11
evm-cross-chain/ERC1967Proxy.sol	99a7c9db7d9e2d9a1ef635827a2904034866a680
evm-cross-chain/OrderlyCrossChainMessage.sol	2077cdc819f88c3e8f92fa53636ed049f81ae38c
evm-cross-chain/VaultCrossChainManagerUpgradeable.sol	6d30de42b5c90f2d9e039e7ff3c69ba8425fcd51