Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.09.17, the SlowMist security team received the Jovay Network team's security audit application for Jovay Contracts Phase2, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Jovay is a high-performance, user-friendly Layer 2 scaling solution tailored for real-world assets, applications, and users, designed to facilitate asset bridging for interchain liquidity, conducting asset tokenization, and integrate value. This audit focuses on Jovay's related contracts, primarily encompassing the sequencer and rollup modules. The sequencer module is predominantly responsible for validator management. The rollup module is mainly utilized for cross-chain asset transfers and for relayers to submit L2 data and proofs to L1. Currently, the rollup module exclusively supports TEE verification.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Redundant permission check for address 0	Authority Control Vulnerability Audit	Suggestion	Acknowledged
N2	Optimizable sliceBytes	Gas Optimization Audit	Suggestion	Fixed
N3	Excess native tokens will be locked	Design Logic Audit	Low	Acknowledged
N4	Risk of excessive privilege	Authority Control Vulnerability Audit	Medium	Fixed
N5	Redundant receive function in the mailbox	Others	Low	Fixed
N6	l1MsgCount not cleared when revert batch	Design Logic Audit	Suggestion	Acknowledged
N7	Redundant rollupTimeLimit	Others	Suggestion	Acknowledged
N8	TEE cache validation will bypass certificate expiration checks	Design Logic Audit	Low	Acknowledged
N9	Incompatible with attestation fee verification feature	Design Logic Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/jovaynetwork/jovay-contracts/tree/audit/dev

commit: de7de4cc268d5cc805ea1759b4ef21b009b43448

Fixed Version:

https://github.com/jovaynetwork/jovay-contracts/tree/audit/dev

commit: e0c415dc2856a4d0ad956d31f9771ea07e03a7ac

Audit Scope:

```
.
├── rollup_contracts/contracts
│   ├── common
│   │   ├── BridgeBase.sol
│   │   ├── interfaces
│   │   │   ├── IBridgeBase.sol
│   │   │   ├── IERC20Token.sol
│   │   │   ├── IGasPriceOracle.sol
│   │   │   ├── IMailBoxBase.sol
│   │   │   └── ITokenBridge.sol
│   │   ├── MailBoxBase.sol
│   │   └── TokenBridge.sol
│   ├── L1
│   │   ├── bridge
│   │   │   ├── interfaces
│   │   │   │   ├── IL1BridgeProof.sol
│   │   │   │   └── IL1ETHBridge.sol
│   │   │   ├── L1BridgeProof.sol
│   │   │   └── L1ETHBridge.sol
│   │   ├── core
│   │   │   ├── L1Mailbox.sol
│   │   │   └── Rollup.sol
│   │   ├── interfaces
│   │   │   ├── IL1Mailbox.sol
│   │   │   ├── IL1MailQueue.sol
│   │   │   └── IRollup.sol
│   │   ├── libraries
│   │   │   ├── codec
│   │   │   │   └── BatchHeaderCodec.sol
│   │   │   └── verifier
│   │   │       ├── ITeeRollupVerifier.sol
│   │   │       ├── IZkRollupVerifier.sol
│   │   │       └── WithdrawTrieVerifier.sol
│   │   └── tee_verifier
│   │       ├── src
│   │       │   ├── DcapAttestationRouter.sol
│   │       │   ├── interfaces
│   │       │   │   └── ITeeRollupVerifier.sol
│   │       │   ├── MeasurementDao.sol
│   │       │   ├── TEECacheVerifier.sol
│   │       │   └── TEEVerifierProxy.sol
│   │       └── lib
│   │           ├── automata-dcap-attestation/evm/contracts/verifiers/V5QuoteVerifier.sol
│   │           └── automata-dcap-attestation/evm/contracts/PCCSRouter.sol
│   └── L2
│       ├── bridge
│       │ ├── interfaces
│       │ │ └── IL2ETHBridge.sol
│       │ └── L2ETHBridge.sol
│       ├── core
│       │ └── L2Mailbox.sol
│       ├── interfaces
│       │ ├── IL2Mailbox.sol
│       │ └── IL2MailQueue.sol
│       └── libraries
│           └── common
│               └── AppendOnlyMerkleTree.sol
└── sequencer_contracts
    └── sys_contract
        └── artifact_src
            └── solidity
                ├── permission_control.sol
                ├── rule_mng.sol
                ├── sys_chaincfg.sol
                └── sys_staking.sol
```

NOTE: The automata-dcap-attestation module only audits the differences from the Automata Network's implementation.

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

BridgeBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
setMailBox	External	Can Modify State	whenPaused onlyOwner
setToBridge	External	Can Modify State	whenPaused onlyOwner
pause	External	Can Modify State	onlyOwner

BridgeBase			
unpause	External	Can Modify State	onlyOwner
mailBoxCall	Internal	Can Modify State	-

MailBoxBase			
Function Name	Visibility	Mutability	Modifiers
__MailBox_init	Internal	Can Modify State	onlyInitializing
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
estimateMsgFee	Public	-	-
setBaseFee	External	Can Modify State	onlyOwner
_appendMsg	Internal	Can Modify State	-
_sendMsgCheck	Internal	Can Modify State	-
_receiveMsgCheck	Internal	Can Modify State	-
_msgExistCheck	Internal	-	-
_encodeCall	Internal	-	-
<Receive Ether>	External	Payable	-
addBridge	External	Can Modify State	onlyOwner
removeBridge	External	Can Modify State	onlyOwner
_getRollingHash	Internal	Can Modify State	-

TokenBridge			
Function Name	Visibility	Mutability	Modifiers
setTokenMapping	Public	Payable	onlyOwner whenNotPaused

TokenBridge

_increaseBalance	Internal	Can Modify State	-
_decreaseBalance	Internal	Can Modify State	-

L1BridgeProof

Function Name	Visibility	Mutability	Modifiers
relayMsgWithProof	External	Payable	whenNotPaused

L1ETHBridge

Function Name	Visibility	Mutability	Modifiers
deposit	External	Payable	nonReentrant whenNotPaused
finalizeWithdraw	External	Payable	nonReentrant onlyMailBox whenNotPaused

L1Mailbox

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
setRollup	External	Can Modify State	whenPaused onlyOwner
setWithdrawer	External	Can Modify State	onlyOwner
sendMsg	External	Payable	onlyBridge whenNotPaused nonReentrant
relayMsgWithProof	External	Payable	whenNotPaused nonReentrant
withdrawDepositFee	External	Can Modify State	onlyWithdrawer whenNotPaused
setL2GasLimit	External	Can Modify State	onlyOwner

L1Mailbox

setL2FinalizeDepositGasUsed	External	Can Modify State	onlyOwner
nextMsgIndex	Public	-	-
getMsg	External	-	-
setLastQueueIndex	External	Can Modify State	whenPaused onlyOwner
_appendMsg	Internal	Can Modify State	-
popMsgs	External	Can Modify State	onlyRollup whenNotPaused

Rollup

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
importGenesisBatch	External	Can Modify State	onlyOwner
commitBatch	External	Can Modify State	OnlyRelayer whenNotPaused
verifyBatch	External	Can Modify State	OnlyRelayer whenNotPaused
revertBatches	External	Can Modify State	onlyOwner
getL2MsgRoot	External	-	-
_getBlobDataHash	Internal	Can Modify State	-
_loadBatchHeader	Internal	-	-
_verifyTeeProof	Internal	Can Modify State	-
addRelayer	External	Can Modify State	onlyOwner
removeRelayer	External	Can Modify State	onlyOwner
setPause	External	Can Modify State	onlyOwner

Rollup			
setMaxTxsInChunk	External	Can Modify State	onlyOwner
setMaxBlockInChunk	External	Can Modify State	onlyOwner
setMaxCallDataInChunk	External	Can Modify State	onlyOwner
setL1BlobNumberLimit	External	Can Modify State	onlyOwner
setRollupTimeLimit	External	Can Modify State	onlyOwner
setL2ChainId	External	Can Modify State	onlyOwner
setTeeVerifierAddress	External	Can Modify State	onlyOwner whenPaused
setZkVerifierAddress	External	Can Modify State	onlyOwner whenPaused

DcapAttestationRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setConfig	External	Can Modify State	onlyOwner
setAuthorized	External	Can Modify State	onlyOwner
enableCallerRestriction	External	Can Modify State	onlyOwner
disableCallerRestriction	External	Can Modify State	onlyOwner
verifyProof	External	Can Modify State	onlyAuthorized
_setConfig	Private	Can Modify State	-
enableVerifyMRTD	External	Can Modify State	onlyOwner
disableVerifyMRTD	External	Can Modify State	onlyOwner
_verifyMeasurement	Private	-	-
_verifyProof	Private	Can Modify State	-

MeasurementDao			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
add_mr_enclave	External	Can Modify State	onlyOwner
delete_mr_enclave	External	Can Modify State	onlyOwner
get_mr_enclave	External	-	-
clearup_mr_enclave	External	Can Modify State	onlyOwner
add_rtMr	External	Can Modify State	onlyOwner
delete_rtMr	External	Can Modify State	onlyOwner
get_rtMr	External	-	-
clearup_rtMr	External	Can Modify State	onlyOwner
add_mrtd	External	Can Modify State	onlyOwner
delete_mrtd	External	Can Modify State	onlyOwner
get_mrtd	External	-	-
clearup_mrtd	External	Can Modify State	onlyOwner
verifyMeasurementSGX	External	-	-
verifyMeasurementTDX	External	-	-
verifyMRTD	External	-	-

TEECacheVerifier			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	P256Verifier
setAuthorized	External	Can Modify State	onlyOwner
enableCallerRestriction	External	Can Modify State	onlyOwner

TEECacheVerifier			
isInitialized	External	-	-
initializeCache	External	Can Modify State	onlyAuthorized
deleteKey	External	Can Modify State	onlyOwner
clearupAllKey	External	Can Modify State	onlyOwner
getAllKey	External	-	-
parseAttestationKey	External	-	-
verifyAndAttestOnChain	External	-	onlyAuthorized
_parseKeyV3	Private	-	-
_parseKeyV4	Private	-	-
_parseKeyV5	Private	-	-
_verifyQuoteV3	Private	-	-
_verifyQuoteV4	Private	-	-
_verifyQuoteV5	Private	-	-

TEEVerifierProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setConfig	External	Can Modify State	onlyOwner
setAuthorized	External	Can Modify State	onlyOwner
enableCallerRestriction	External	Can Modify State	onlyOwner
disableCallerRestriction	External	Can Modify State	onlyOwner
verifyProof	External	Can Modify State	onlyAuthorized
_setConfig	Private	Can Modify State	-

L2ETHBridge			
Function Name	Visibility	Mutability	Modifiers
withdraw	External	Payable	nonReentrant whenNotPaused
finalizeDeposit	External	Payable	nonReentrant onlyMailBox whenNotPaused
claimDeposit	External	Can Modify State	nonReentrant whenNotPaused
claimDeposit	External	Can Modify State	nonReentrant whenNotPaused

L2Mailbox			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
setL1MailBox	External	Can Modify State	whenPaused onlyOwner
sendMsg	External	Payable	onlyBridge whenNotPaused nonReentrant
relayMsg	External	Can Modify State	whenNotPaused nonReentrant
claimAmount	External	Can Modify State	onlyBridge whenNotPaused nonReentrant
_appendMsg	Internal	Can Modify State	-
msgRoot	External	-	-
_receiveMsgFailed	Internal	Can Modify State	-
_receiveMsgSuccess	Internal	Can Modify State	-
_checkMsgClaimValid	Internal	-	-
_finalizeClaimMsg	Internal	Can Modify State	-

PermissionControl			
Function Name	Visibility	Mutability	Modifiers

PermissionControl			
<Constructor>	Public	Can Modify State	-
checkSuperPermission	Internal	-	-
checkAdminPermission	Internal	-	-
checkGrantPermission	Internal	-	-
tranferSuperAdmin	External	Can Modify State	-
getSuperAdmin	Public	-	-
getGranteeAdmin	Public	-	-
grantAdmin	External	Can Modify State	-
revokeAdmin	External	Can Modify State	-

InferRuleManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
checkExist	Private	-	-
checkExist	Private	-	-
checkSuperPermission	Private	-	-
checkGrantPermission	Private	-	-
checkAdminPermission	Private	-	-
checkPermission	Private	-	-
checkPermission	Private	-	-
addRule	Public	Can Modify State	-
updateRule	Public	Can Modify State	-
delRule	Public	Can Modify State	-

InferRuleManager

getAllRules	Public	-	-
getNextId	Public	-	-
getContractRules	Public	-	-
updateProvingResult	Public	Can Modify State	-
advanceEpoch	External	Can Modify State	-
tranferSuperAdmin	External	Can Modify State	-
getSuperAdmin	Public	-	-
getGranteeAdmin	Public	-	-
grantAdmin	External	Can Modify State	-
revokeAdmin	External	Can Modify State	-

ChainCfg

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
changeSys	Public	Can Modify State	onlyOwner
get_config	Public	-	-
get_configs	Public	-	-
set_config	External	Can Modify State	onlyOwner

DPoSValidatorManager

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
hexStringToBytes	Public	-	-

DPoSValidatorManager			
sliceBytes	Internal	-	-
_fromHexChar	Internal	-	-
isArrayContains	Internal	-	-
isValidatorActive	Public	-	-
isValidatorPendingAdd	Public	-	-
isValidatorPendingExit	Public	-	-
_transferTo	Internal	Can Modify State	-
updateValidator	External	Can Modify State	-
advanceEpoch	Public	Can Modify State	onlyOwner
advanceEpoch	Public	Can Modify State	onlyOwner
setChainEpochBlock	Internal	Can Modify State	-
getActiveValidators	External	-	-
getPendingAddValidators	External	-	-
getPendingExitValidators	External	-	-
getWithdrawEffectiveWindow	Internal	-	-
getChainCfg	Internal	-	-
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Suggestion] Redundant permission check for address 0

Category: Authority Control Vulnerability Audit

Content

In the `PermissionControl` and `InferRuleManager` contracts, the `checkSuperPermission` function is designed to verify if a provided address possesses super permissions. Notably, if the input address is the zero address, it is implicitly granted super permissions. However, it has been observed that there is no existing implementation within the contracts where the zero address is actually passed to the `checkSuperPermission` function for permission verification. Consequently, this check for the zero address becomes superfluous.

Code location:

sequencer_contracts/sys_contract/artifact_src/solidity/permission_control.sol#L15

sequencer_contracts/sys_contract/artifact_src/solidity/rule_mng.sol#L71

```
    function checkSuperPermission(address _addr) private view returns(bool) {
        if (_addr == administrator_ || _addr == address(0)) {
            return true;
        }

        return false;
    }
```

Solution

It is recommended to remove this redundant zero address check.

Status

Acknowledged

[N2] [Suggestion] Optimizable sliceBytes

Category: Gas Optimization Audit

Content

In the DPoSValidatorManager contract, the `sliceBytes` function iterates through all bytes of the data using a `for` loop to trim specified bytes. Its primary purpose is to remove the `0x` prefix from public keys provided by users. Although this approach is unlikely to cause a Denial of Service (DoS) issue for public keys with a fixed number of bytes, a simpler `sliceBytes` implementation could still significantly reduce gas consumption.

Code location: sequencer_contracts/sys_contract/artifact_src/solidity/sys_staking.sol#L2343-L2349

```
    function sliceBytes(bytes memory data, uint start, uint end) internal pure returns
  (bytes memory) {
        bytes memory result = new bytes(end - start);
        for (uint i = start; i < end; i++) {
            result[i - start] = data[i];
        }
        return result;
    }
```

Solution

Gas consumption can be effectively reduced by either directly skipping the `0x` prefix using offset operations or by performing slicing operations with inline assembly.

Status

Fixed

[N3] [Low] Excess native tokens will be locked

Category: Design Logic Audit

Content

In the `DPoSValidatorManager` contract, a `receive` function is implemented, allowing users to directly transfer native tokens into the contract. It should be noted that this also implies users might inadvertently transfer native tokens. However, such mistakenly transferred tokens cannot be withdrawn and will either remain locked within the contract or be distributed as rewards to validators.

Code location: sequencer_contracts/sys_contract/artifact_src/solidity/sys_staking.sol#L2751

```
    receive() external payable {
        emit BalanceReceived(
            currentEpoch,
            block.number,
            msg.sender,
            msg.value,
            address(this).balance
        );
    }
```

Solution

If this is not the intended design, it is recommended to adopt alternative methods for receiving native token deposits instead of directly utilizing the `receive` function.

Status

Acknowledged

[N4] [Medium] Risk of excessive privilege

Category: Authority Control Vulnerability Audit

Content

In the `L2Mailbox` contract, the `l1MailBox` role can complete cross-chain operations from L1 to L2 for users through the `relayMsg` function. However, the `owner` role can arbitrarily modify the `l1MailBox` address via the `setL1MailBox` function, which poses a risk of excessive privilege.

Code location: rollup_contracts/contracts/L2/core/L2Mailbox.sol#L36-L39

```
    function setL1MailBox(address l1MailBox_) external onlyOwner {
         require(l1MailBox_ != address(0), "Invalid address");
         l1MailBox = l1MailBox_;
     }
```

Solution

In the short term, to satisfy business requirements, managing the privileged role through a multi-signature scheme can effectively mitigate single-point risk. In the long term, entrusting these privileged roles to DAO governance can effectively resolve the risk of excessive privilege. During the transition period, managing through a multi-signature scheme combined with delayed transaction execution via a timelock can significantly alleviate the risk of excessive privilege.

Status

Fixed; The protocol removed this function to fix the l1MailBox address.

[N5] [Low] Redundant receive function in the mailbox

Category: Others

Content

In the `MailBoxBase` contract, a `receive` function is defined for accepting native tokens, which means users can directly send native tokens to the L1 mailbox contract. However, it is crucial to note that directly sending native tokens to the mailbox contract does not facilitate normal cross-chain transfers; native token cross-chain transfers must be conducted through a bridge contract. Consequently, native tokens sent directly to the mailbox contract will be locked and cannot be withdrawn.

Code location: rollup_contracts/contracts/common/MailBoxBase.sol#L92

```
    receive() external payable {}
```

Solution

It is recommended to remove the unnecessary `receive` function from the `MailBoxBase` contract.

Status

Fixed

[N6] [Suggestion] l1MsgCount not cleared when revert batch

Category: Design Logic Audit

Content

In the Rollup contract, the owner can roll back unverified batches using the `revertBatches` function. However, the `l1MsgCount` corresponding to these batches is not deleted during the rollback. This results in the `l1MsgCount` data persisting even after the batch rollback.

Code location: rollup_contracts/contracts/L1/core/Rollup.sol#L235

```
    function revertBatches(uint256 _newLastBatchIndex) external override onlyOwner {
         ...
     }
```

Solution

If this is not the intended design, it is recommended to delete the `l1MsgCount` data concurrently with the `revertBatches` operation.

Status

Acknowledged

[N7] [Suggestion] Redundant rollupTimeLimit

Category: Others

Content

The `rollupTimeLimit` global variable is defined in the Rollup contract, but it is not utilized in the business logic. This makes it redundant.

Code location: rollup_contracts/contracts/L1/core/Rollup.sol#L33

```
    uint64 public rollupTimeLimit;
```

Solution

It is recommended to remove unnecessary parameters to simplify the contract.

Status

Acknowledged; After communicating with the project team, the project team stated that the relayer role will use this variable.

[N8] [Low] TEE cache validation will bypass certificate expiration checks

Category: Design Logic Audit

Content

In the `DcapAttestationRouter` contract, the `_verifyProof` function is used to validate the authenticity of proofs. To enhance verification efficiency, the system incorporates a cached validation option. When this option is enabled, public keys that have undergone full verification can directly proceed to signature validation, bypassing the cumbersome certificate chain verification. While this improves verification efficiency, it also introduces potential risks: if a cached certificate has expired or been revoked, the protocol's cached validation might still deem it legitimate.

Code location: rollup_contracts/contracts/L1/tee_verifier/src/DcapAttestationRouter.sol#L133-L136

```
    function _verifyProof(bytes calldata aggrProof) private returns (uint32
  _error_code, bytes32 commitment) {
        ...
        if (CacheOption && CacheAttestation.isInitialized(ecdsaAttestationKey)) {
            (_error_code, commitment) = CacheAttestation.verifyAndAttestOnChain(
                aggrProof, ecdsa256BitSignature, ecdsaAttestationKey, quoteVersion
            );
        } else {
            ...
        }
    }
```

Solution

This requires the protocol to have a robust off-chain certificate management system to ensure the timely removal of invalid certificates stored in the `TEECacheVerifier`.

Status

Acknowledged; After communicating with the project team, the project team stated that they will detect the validity of certificates in real time off-chain and clean up invalid certificates in real time through the deleteKey and clearupAllKey functions.

[N9] [Suggestion] Incompatible with attestation fee verification feature

Category: Design Logic Audit

Content

When performing TEE verification, the protocol utilizes Automate's `DcapAttestationFee` implementation for proof validity checks. It is important to note that the `DcapAttestationFee` implementation may involve fees, which the owner can set via `feeBP` to charge for verification. However, the `DcapAttestationRouter` currently lacks a fee payment mechanism. Although the project team has not yet decided to enable fee collection, this compatibility issue should still be acknowledged.

Code location: rollup_contracts/contracts/L1/tee_verifier/src/DcapAttestationRouter.sol#L138-L139

```
    function _verifyProof(bytes calldata aggrProof) private returns (uint32
_error_code, bytes32 commitment) {
        ...
        if (CacheOption && CacheAttestation.isInitialized(ecdsaAttestationKey)) {
            (_error_code, commitment) = CacheAttestation.verifyAndAttestOnChain(
                aggrProof, ecdsa256BitSignature, ecdsaAttestationKey, quoteVersion
            );
        } else {
            AutomataDcapAttestationFee attestation =
AutomataDcapAttestationFee(dcapAttestation);
            (success, output) = attestation.verifyAndAttestOnChain(aggrProof);

            ...
            }
        }
```

Solution

Should the fee collection function be activated in the future, the `DcapAttestationRouter` would need to implement the necessary fee payment functionality.

Status

Acknowledged; After communicating with the project team, the project team stated that the existing DcapAttestationFee implementation will not enable the feeBP function.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002509180001	SlowMist Security Team	2025.09.17 - 2025.09.18	Passed

Summary conclusion: The SlowMist security team uses a manual and the SlowMist team's analysis tool to audit the project. During the audit work, we found 1 medium risk, 3 low risks, and 5 suggestions. All the findings were fixed or acknowledged. The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.