ANASTASIA LABS

Security Audit Report

Date: May 1, 2026
Project: Via Gateway V1 EVM Contracts
Version: 1.1

Contents

Disclosure

Disclaimer and Scope

A code review represents a snapshot in time, and the findings and recommendations presented in this report reflect the information gathered during the assessment period. It is important to note that any modifications made outside of this timeframe will not be captured in this report.

While diligent efforts have been made to uncover potential vulnerabilities, it is essential to recognize that this assessment may not uncover all potential security issues in the protocol.

It is imperative to understand that the findings and recommendations provided in this audit report should not be construed as investment advice.

Furthermore, it is strongly recommended that projects consider undergoing multiple independent audits and/or participating in bug bounty programs to increase their protocol security.

The assessment covered the Via Gateway V1 EVM smart contracts at the commit listed in the Code base section. Third-party dependencies, deployment configuration, off-chain relayer infrastructure, operational key management, and future integrations outside the reviewed source tree were not directly audited except where their trust assumptions affect the smart-contract security model.

The review assumes an honest initial deployment and sensible production operation of a single live gateway deployment. Findings requiring malicious initial setup, intentionally self-sabotaging configuration, or full trusted signer-set collusion are not treated as core protocol vulnerabilities.

Assessment overview

Anastasia Labs reviewed the Via Gateway V1 EVM contracts, a Solidity bridge gateway and integration framework for sending, validating, and processing cross-chain messages. The review focused on the gateway trust boundary, signature and relayer authorization, replay protection, recipient callback behavior, example bridge integrations, fee collection, and relayer gas refund flows.

Phases of code auditing activities included the following:

- **Planning** - Establish review scope, trust assumptions, and production threat model.
- **Discovery** - Perform manual code review and inspect protocol invariants, external calls, and authorization paths.
- **Attack** - Validate candidate issues through targeted reasoning and proof-of-concept tests where appropriate.
- **Reporting** - Document confirmed findings, impact, and remediation guidance.

Each issue was classified by severity and is presented with impact and recommendation sections.

Assessment components

Manual revision

Our manual code auditing is focused on a wide range of attack vectors, including but not limited to:

- Cross-chain message replay protection
- Signature validation and signer-threshold enforcement
- Relayer authorization and project-level policy isolation
- Destination callback failure and retry semantics
- Source-side burn, lock, mint, and release accounting
- Fee-on-transfer and non-standard ERC20 behavior
- Project-funded fee and gas-refund reserve depletion
- Reentrancy and callback-time state mutation
- Owner and project-owner access control boundaries
- Unchecked ERC20 return values and partial setup failure modes

Executive summary

Protocol architecture

Via Gateway V1 is an EVM bridge messaging system centered around `ViaGatewayV1`. The gateway emits outbound messages, validates inbound messages using Chain, Via, and optional Project-layer signatures, authorizes relayers, dispatches recipient callbacks, and coordinates optional fee and gas refund collectors. `ViaIntegrationV1` is the base integration contract used by project adapters and example bridge clients.

Risk profile

The review identified one high-impact protocol issue: destination callback failure permanently consumes a message even after source-side assets have already been burned or locked. Medium-severity issues affect token-accounting assumptions, unchecked ERC20 approvals, and callback-time project-policy mutation for permissionless executor-style integrations. Minor and informational items cover relayer self-registration behavior, ambiguous PoS hook semantics, and sponsored fee/refund reserve depletion.

Finding summary

The report contains 1 Major, 3 Medium, 2 Minor, and 1 Informational findings. Following the client's remediation pass, VIA-EVM-001 through VIA-EVM-006 are marked as Resolved based on client-provided remediation commits, and VIA-EVM-007 is marked as Acknowledged.

Code base

Repository

git@github.com:VIA-Labs-Tech/vg1-evm-contracts.git

Commit

1fc3432be238f3126be190e7500c3683e4722ccd

Files audited

SHA256 Checksum	Files
6a503a4926ad0c6c940c3f5ef660d37f5daf655b9f7710b95bde57b6501deac4	src/ViaGatewayV1.sol
cd433c096814068b3608f2ffe5fd76008c088f795c44123576ea111b182d6231	src/ViaIntegrationV1.sol
277ce346618de10b25a989aab3158e283fbaf1369d7d2a87d0a52d59528cf730	src/GasRefundV1.sol
d9b9ae34f69aab71c777297b733d56a980d899134cbcb878ba63f28462d83161f	src/FeeCollectorV1.sol
92aec352c415d83e69c268b0c67a99f95e249d96df589d696585cb60b4adc329	src/VIAMintBurnToken.sol
4cd7689cdb7537c8dadc91edfefe4bc8951420e05490e929194427d23d12a2[illegible]1	src/VIAMintBurnTokenMinimal.sol
7467f7a6457ec3b085ae2d1c67cda8544e9817f2941608b1d4acd5eef769d677	src/VIALockerRelease.sol
dc1919f8452b6c55e85df55eacf7f34ffd767255a2a31a0044ae117737c3a1b[illegible]	src/IViaGatewayV1.sol
18821f751eda6fa144ad33be017daf968fe833913737871a8dd404c1f7646728	src/IViaIntegrationV1.sol
8843d33192268f1195ff5752d99702a499bab80574bb0e494327e90f59f8cc8c	src/IFeeCollector.sol
5dd5a8f987a177d076f7bb58ac129de6f2688f3ab4b53dc9d52cb05fa661a4e9	src/IGasCollector.sol
ed557e180e8fafcd85be9f3b0c592d2bc8a48f3e0666f5be33d7f4c5c12dac4b	src/IWeth.sol
b9f51cb97ed6681bfb0fa2f5800c015024429b22881c1e2512d239df46cc4afa	src/IPos.sol

Severity Classification

- **Critical**: This vulnerability has the potential to result in significant financial losses to the protocol. They often enable attackers to directly steal assets from contracts or users, or permanently lock funds within the contract.
- **Major**: Can lead to damage to the user or protocol, although the impact may be restricted to specific functionalities or temporal control. Attackers exploiting major vulnerabilities may cause harm or disrupt certain aspects of the protocol.
- **Medium**: May not directly result in financial losses, but they can temporarily impair the protocol's functionality. Examples include susceptibility to front-running attacks, which can undermine the integrity of transactions.
- **Minor**: Minor vulnerabilities do not typically result in financial losses or significant harm to users or the protocol. The attack vector may be inconsequential or the attacker's incentive to exploit it may be minimal.
- **Informational**: These findings do not pose immediate financial risks. These may include protocol optimizations, code style recommendations, alignment with naming conventions, overall contract design suggestions, and documentation discrepancies between the code and protocol specifications.

Finding severity ratings

The following table shows all findings by severity.

Level	Severity	Findings
5	Critical	0
4	Major	1
3	Medium	3
2	Minor	2
1	Informational	1

The following table shows outstanding findings by severity. Findings marked `Resolved` are excluded; acknowledged findings remain counted as outstanding.

Level	Severity	Outstanding Findings
5	Critical	0
4	Major	0
3	Medium	0
2	Minor	0
1	Informational	1

Findings

ID	Title	Severity	Status
VIA-EVM-001	Failed destination execution permanently consumes messages after source-side asset destruction	Major	Resolved
VIA-EVM-002	`VIALockerRelease` does not account for fee-on-transfer or balance-changing tokens	Medium	Resolved
VIA-EVM-003	Unchecked `approve` return values can leave collector approvals unset	Medium	Resolved
VIA-EVM-004	Recipient callback code can self-mutate future gateway project policy state	Medium	Resolved
VIA-EVM-005	Arbitrary EOAs can self-register as relayers and process messages for unrestricted recipients	Minor	Resolved
VIA-EVM-006	`posHandler.validateMessage(signatures)` return value is ignored	Minor	Resolved
VIA-EVM-007	Public bridge calls can drain project-funded fee and gas reserves	Informational	Acknowledged

VIA-EVM-001 Failed destination execution permanently consumes messages after source-side asset destruction

Severity	Status
Major	Resolved

Description

`ViaGatewayV1.process()` marks a message as processed before the destination callback succeeds. The destination call is then executed through `processCall()` and wrapped in `try/catch`. If the recipient callback reverts, the gateway emits an `Error` event rather than reverting or restoring retryability.

At the same time, the source-side bridge adapters already destructively changed user balances before the message is delivered on the destination chain. The mint/burn adapters burn tokens before sending the message, and the locker/release adapter locks tokens before sending the message.

Affected smart contracts: `src/ViaGatewayV1.sol`, `src/ViaIntegrationV1.sol`, `src/VIAMintBurnToken.sol`, `src/VIAMintBurnTokenMinimal.sol`, `src/VIALockerRelease.sol`.

Relevant destination-side behavior:

```
processedMessages[txId] = true;

_validateMessage(...);

try this.processCall(...) returns (bool success, string memory reason) {
    if (success) {
        emit Success();
    } else {
        emit Error(txId, string.concat("ViaGatewayV1 rcpt ", reason));
    }
} catch {
    emit Error(txId, "ViaGatewayV1 uclb fatal");
}
```

Relevant source-side destructive actions:

```
// VIAMintBurnToken / VIAMintBurnTokenMinimal
_burn(msg.sender, amount);

// VIALockerRelease
token.safeTransferFrom(msg.sender, address(this), amount);
```

Impact: A valid message can become permanently consumed even though the destination action failed. If the source side already burned or locked user assets, the user can be left with permanently lost or stranded funds unless there is a separate off-chain or administrative compensation process.

Recommendation

Use an explicit delivery lifecycle that does not make failed recipient execution terminal. Possible approaches include marking messages completed only after the recipient callback succeeds, introducing a retryable pending/completed state machine, or adding a protocol-level compensation path for failed destination deliveries after source-side burns or locks.

Resolution

Resolved. VIA Labs remediated this issue in 1c03912460f61f09397592d506a9b39ae687b61a on April 8, 2026.

VIA-EVM-002 `VIALockerRelease` does not account for fee-on-transfer or balance-changing tokens

Severity	Status
Medium	Resolved

Description

`VIALockerRelease.bridge()` and `deposit()` trust the requested `amount` instead of measuring the actual amount received. This is unsafe for fee-on-transfer tokens, rebasing or balance-changing tokens, and other non-standard ERC20 behavior.

Affected smart contract: `src/VIALockerRelease.sol`.

```
function bridge(
    bytes32 tokenRecipient,
    uint64 destChainId,
    uint256 amount
) external payable returns (uint256) {
    if (amount == 0) revert ZeroAmount();

    token.safeTransferFrom(msg.sender, address(this), amount);

    bytes memory chainData = abi.encode(tokenRecipient, amount);
    uint256 txId = messageSend(destChainId, chainData, 1);

    return txId;
}
```

The destination side later releases the full signed `amount`:

```
(bytes32 tokenRecipient, uint256 amount) =
    abi.decode(onChainData, (bytes32, uint256));

uint256 bal = token.balanceOf(address(this));
if (bal < amount) revert InsufficientLiquidity(amount, bal);

token.safeTransfer(recipientAddress, amount);
```

Impact: If the source-side token transfers less than `amount`, the destination side can release more value than was actually locked. Over time, this can undercollateralize the bridge and drain prefunded destination liquidity.

Recommendation

Use balance-delta accounting for all token receipt paths and encode the actual amount received, not the user-requested amount.

```
uint256 beforeBalance = token.balanceOf(address(this));
token.safeTransferFrom(msg.sender, address(this), requestedAmount);
uint256 actualReceived = token.balanceOf(address(this)) - beforeBalance;
```

If non-standard tokens are intentionally unsupported, enforce that through allowlisting, wrappers, or explicit deployment constraints.

Resolution

Resolved. VIA Labs remediated this issue in 37b5114d1e4ec08bd52c8df831a63a0ad8fe879b.

VIA-EVM-003 Unchecked `approve` return values can leave collector approvals unset

Severity	Status
Medium	Resolved

Description

Some ERC20 tokens return `false` from `approve()` rather than reverting. The current approval paths ignore the return value. As a result, setup can appear to succeed even when the approval was not actually granted.

Affected smart contracts: `src/ViaIntegrationV1.sol`, `src/GasRefundV1.sol`.

```
IERC20(currentGasToken).approve(
    gasCollector,
    type(uint256).max
);
```

```
if (addr != address(0)) gasToken.approve(addr, type(uint256).max);
```

Impact: Collector approvals may remain unset or stale after configuration. The most likely result is broken fee/refund operation or confusing partial setup rather than direct asset theft.

Recommendation

Use checked approval handling, such as OpenZeppelin `SafeERC20.forceApprove`, or explicitly require that `approve()` returns `true` when interacting with ERC20s that return a boolean.

Resolution

Resolved. VIA Labs remediated this issue in b89536b0e1f4ce61b2968e0285e9cc518019f44e.

VIA-EVM-004 Recipient callback code can self-mutate future gateway project policy state

Severity	Status
Medium	Resolved

Description

During the real `ViaGatewayV1.process()` path, the shared OpenZeppelin `nonReentrant` lock blocks recipient callbacks from calling `setProjectSigner()` or `setProjectRelayer()`. The owner-only gateway APIs for setting specific project signers or relayers also remain protected by `onlyOwner`.

However, the project-policy setters are unguarded and self-scoped to `msg.sender`:

```
function setRequiredProjectSignerCounts(uint256 amount) external {
    projectSignersRequired[msg.sender] = amount;

    emit SetRequiredProjectSignerCounts(amount);
}

function setIsProjectRelayerRestricted(bool restricted) external {
    isProjectRelayerRestricted[msg.sender] = restricted;

    emit SetIsProjectRelayerRestricted(restricted);
}
```

This means recipient callback code can rewrite that recipient's own future gateway policy during message delivery.

The shipped `VIAMintBurnToken` and `VIALockerRelease` integrations are not vulnerable to this specific pattern because their callbacks do not dispatch user-controlled calls back into the gateway. The issue is still relevant to the permissionless integration model. An external client can integrate with the gateway and implement a cross-chain executor, router, cross-chain account, or automation contract whose callback executes destination actions encoded by a source-chain user.

A semi-realistic vulnerable integration looks like:

```
contract PermissionlessExecutorIntegration is ViaIntegrationV1 {
    constructor(address projectOwner_) ViaIntegrationV1(projectOwner_) {}

    function messageProcess(
        uint256,
        uint64,
        bytes32,
        bytes32,
        bytes memory onChainData,
        bytes memory,
        uint256
    ) internal override {
        (address target, bytes memory callData) =
            abi.decode(onChainData, (address, bytes));
```

```
        (bool success, ) = target.call(callData);
        require(success, "executor call failed");
    }
}
```

For such an integration, a user can encode a valid cross-chain message whose destination action is:

`gateway.setRequiredProjectSignerCounts(1);`

or:

`gateway.setIsProjectRelayerRestricted(true);`

Because the integration contract performs the call during its callback, `msg.sender` at the gateway is the integration contract. The gateway therefore rewrites policy for that integration itself.

Impact: For vulnerable executor-style integrations, a normal signed message can unexpectedly change the integration's future gateway policy. Depending on the chosen value, this can cause future messages to require project signatures that operators did not configure, or can restrict future relayer eligibility and create avoidable liveness failures.

Recommendation

If callback-time policy mutation is not intended, add equivalent reentrancy protection to the project-policy setters or gate those writes behind explicit owner-controlled integration flows. Executor-style integrations should also denylist calls to the gateway policy setters unless policy mutation is a deliberate supported action.

Resolution

Resolved. VIA Labs remediated this issue in c4922d2a72631e55d6b7daeca24bcfcfc734756f.

VIA-EVM-005 Arbitrary EOAs can self-register as relayers and process messages for unrestricted recipients

Severity	Status
Minor	Resolved

Description

`ViaGatewayV1.setProjectRelayer()` lets any contract call register `tx.origin` as a Project-layer relayer for `msg.sender`. The only authorization check is that the immediate caller is not the transaction origin.

Affected smart contracts: `src/ViaGatewayV1.sol`, `src/ViaIntegrationV1.sol`, `src/GasRefundV1.sol`.

```
function setProjectRelayer(bool enabled) external nonReentrant {
    if (msg.sender == tx.origin) revert CallerMismatch();

    relayers[tx.origin] = Node({
        recipient: msg.sender,
        isEnabled: enabled,
        layer: NodeLayer.Project
    });

    emit SetRelayer(tx.origin, msg.sender, NodeLayer.Project);
}
```

The integration wrapper exposes a stricter project-owner-controlled whitelist flow, but the gateway does not enforce that calls must come through this wrapper.

```
function setProjectRelayer(bool enabled) external onlyAllowedRelayer {
    IViaGatewayV1(gateway).setProjectRelayer(enabled);
}
```

In `RelayerMode.Standard`, unrestricted recipients accept any enabled relayer. The gateway only checks that a Project-layer relayer's recorded `recipient` matches the current recipient when the recipient has explicitly enabled project-relayer restriction.

```
return
    isProjectRelayerRestricted[recipient]
        ? relayer_.recipient == recipient
        : true;
```

A minimal helper capable of registering the attacker looks like:

```
contract ProjectRelayerRegistrationHelper {
    function register(ViaGatewayV1 gateway, bool enabled) external {
        gateway.setProjectRelayer(enabled);
    }
}
```

Impact: An arbitrary EOA can gain message-processing authority for unrelated unrestricted recipients without approval from those projects. This can let the attacker control delivery timing, front-run legitimate relayers, and claim project-funded gas refunds. Because processed messages are single-use, attacker-controlled delivery timing can also amplify liveness failures for fragile recipient callbacks.

Recommendation

In `RelayerMode.Standard`, unrestricted recipients should only accept enabled Via/Chain relayers. Project-layer relayers should be accepted only when their recorded `recipient` matches the current recipient. If the wrapper whitelist flow is intended to be authoritative, enforce that invariant in the gateway instead of relying on integrations to use the wrapper correctly.

Resolution

Resolved. VIA Labs remediated this issue in c4922d2a72631e55d6b7daeca24bcfcfc734756f.

VIA-EVM-006 `posHandler.validateMessage(signatures)` return value is ignored

Severity	Status
Minor	Resolved

Description

The gateway calls the PoS validation hook but does not consume its return value. If the interface return value is intended to express validation success or failure, that decision is currently discarded.

Affected smart contract: `src/ViaGatewayV1.sol`.

```
if (address(posHandler) != address(0))
    posHandler.validateMessage(signatures);
```

Impact: The current semantics of the PoS hook are ambiguous. This is more likely an incomplete or stale interface than an immediate exploit, assuming the hook reverts on validation failure. If downstream implementations return `false` instead of reverting, the gateway would continue processing.

Recommendation

Either consume the return value meaningfully or change the interface to a no-return hook that must revert on failure.

Resolution

Resolved. VIA Labs remediated this issue in 6184c421d764ef0d5efe45cf241b07d371f9f838.

VIA-EVM-007 Public bridge calls can drain project-funded fee and gas reserves

Severity	Status
Informational	Acknowledged

Description

When a project configures its gateway, the integration approves the configured fee and gas collectors from the project contract itself.

Affected smart contracts: `src/ViaIntegrationV1.sol`, `src/FeeCollectorV1.sol`, `src/GasRefundV1.sol`.

```
function setMessageGateway(address gateway_) external onlyProjectOwner {
    gateway = gateway_;

    gasCollector = IViaGatewayV1(gateway).gasCollector();
    feeCollector = IViaGatewayV1(gateway).feeCollector();

    if (gasCollector != address(0)) {
        address currentGasToken = IGasCollector(gasCollector).getGasToken();

        if (currentGasToken != address(0))
            IERC20(currentGasToken).approve(
                gasCollector,
                type(uint256).max
            );
    }
}
```

`messageSend()` calls the gateway from the integration contract context, so the fee collector pulls fees from the project contract. Destination gas refunds also pull from the recipient project contract.

```
// FeeCollectorV1
function pay(address sender) external payable {
    if (msg.sender != gateway) revert CallerMismatch();
    ...
}

// GasRefundV1
function refund(
    address recipient,
    address relayer,
    uint256 gasRefundAmount
) external nonReentrant {
    if (msg.sender != gateway) revert CallerMismatch();
    ...
}
```

Impact: If the bridge entry point is public, any caller can repeatedly initiate bridge messages that consume project-funded fee and gas reserves. This can grief the project, deplete operational balances, and interrupt service even without compromising any privileged role.

Recommendation

If sponsorship is not intended, charge users directly rather than charging the integration contract. If sponsorship is intended, add explicit controls such as allowlists, quotas, per-user budgets, rate limits, or separate capped sponsorship vaults.

Resolution

Acknowledged. VIA Labs acknowledged this informational finding. No remediation commit was provided for this item.