# WCT WalletConnect

HALBORN

# WCT - WalletConnect

Prepared by: HALBORN

Last Updated 11/08/2024

Date of Engagement by: August 28th, 2024 - September 2nd, 2024

## Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

| ALL FINDINGS | CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|---|
| 5 | 0 | 0 | 0 | 2 | 3 |

## TABLE OF CONTENTS

## 1. Introduction

`WalletConnect` engaged `Halborn` to conduct a security assessment on their smart contracts beginning on **08-28-2024** and ending on **09-02-2024**. The security assessment was scoped to the smart contracts provided in the **https://github.com/WalletConnectFoundation/contracts** GitHub repository. Commit hashes and further details can be found in the Scope section of this report. The `WalletConnect` codebase in scope consists of **a decentralized infrastructure for permissionless, interoperable messaging between dApps and wallets to be deployed On Ethereum and Optimism networks.**

## 2. Assessment Summary

`Halborn` was provided 4 days for the engagement and assigned 1 full-time security engineer to review the security of the smart contracts 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.
Additionally, another round of review was performed on the updated codebase, which took place from 10-11-2024 to 10-14-2024.

The purpose of the assessment is to:

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

In summary, `Halborn` identified some improvements to reduce the likelihood and impact of risks, which were mostly addressed by the `WalletConnect team`. The main identified issues were:

- **Lack of upper limit restriction for delay**
- **Missing _disableInitializers function call.**
- **Single step ownership transfers risks DoS for privileged functions.**

## 3. Test Approach And Methodology

`Halborn` performed a combination of manual review of the code and automated security testing 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 smart contracts and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture, purpose and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could led to arithmetic related vulnerabilities.
- Local testing with custom scripts (`Foundry`).
- Fork testing against main networks (`Foundry`).
- Static analysis of security for scoped contract, and imported functions (`Slither`).

### 3.1 Out-Of-Scope

- External libraries and financial-related attacks.
- New features/implementations after/within the **remediation commit IDs**.
- Changes that occur outside of the scope of PRs.

## 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)<br>Specific (AO:S) | 1<br>0.2 |
| Attack Cost (AC) | Low (AC:L)<br>Medium (AC:M)<br>High (AC:H) | 1<br>0.67<br>0.33 |
| Attack Complexity (AX) | Low (AX:L)<br>Medium (AX:M)<br>High (AX:H) | 1<br>0.67<br>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)<br>Low (I:L)<br>Medium (I:M)<br>High (I:H)<br>Critical (I:C) | 0<br>0.25<br>0.5<br>0.75<br>1 |
| Integrity (I) | None (I:N)<br>Low (I:L)<br>Medium (I:M)<br>High (I:H)<br>Critical (I:C) | 0<br>0.25<br>0.5<br>0.75<br>1 |
| Availability (A) | None (A:N)<br>Low (A:L)<br>Medium (A:M)<br>High (A:H)<br>Critical (A:C) | 0<br>0.25<br>0.5<br>0.75<br>1 |
| Deposit (D) | None (D:N)<br>Low (D:L)<br>Medium (D:M)<br>High (D:H)<br>Critical (D:C) | 0<br>0.25<br>0.5<br>0.75<br>1 |
| Yield (Y) | None (Y:N)<br>Low (Y:L)<br>Medium (Y:M)<br>High (Y:H)<br>Critical (Y:C) | 0<br>0.25<br>0.5<br>0.75<br>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)<br>Partial (R:P)<br>Full (R:F) | 1<br>0.5<br>0.25 |
| Scope ($s$) | Changed (S:C)<br>Unchanged (S:U) | 1.25<br>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 |
| Informational | 0 - 1.9 |

## 5. SCOPE

### FILES AND REPOSITORY

(a) Repository: contracts

(b) Assessed Commit ID: 58615f6

(c) Items in scope:

- src/CNKT.sol
- src/L2CNKT.sol
- src/Timelock.sol
- script/Base.s.sol
- script/deploy/EthereumDeploy.s.sol
- script/deploy/OptimismDeploy.s.sol

Out-of-Scope:

### FILES AND REPOSITORY

(a) Repository: contracts

(b) Assessed Commit ID: e9d2f4d

(c) Items in scope:

- src/WCT.sol
- src/L2WCT.sol
- src/Timelock.sol
- script/Base.s.sol
- script/deploy/EthereumDeploy.s.sol
- script/deploy/OptimismDeploy.s.sol
- script/helpers/Proxy.sol

Out-of-Scope:

### REMEDIATION COMMIT ID:

- 3bb4b4d
- 8ab9473

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 | 3 |

| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
| --- | --- | --- |
| LACK OF UPPER LIMIT RESTRICTION FOR DELAY | LOW | SOLVED - 09/04/2024 |
| MISSING _DISABLEINITIALIZERS() FUNCTION CALL | LOW | SOLVED - 09/04/2024 |
| SINGLE STEP OWNERSHIP TRANSFER RISKS DOS FOR PRIVILEGED FUNCTIONS | INFORMATIONAL | ACKNOWLEDGED |
| RISK OF EVM VERSION INCOMPATIBILITY ACROSS CHAINS | INFORMATIONAL | ACKNOWLEDGED |
| LACK OF FLEXIBILITY FOR ENABLING/DISABLING TRANSFER RESTRICTIONS | INFORMATIONAL | ACKNOWLEDGED |

# 7. FINDINGS & TECH DETAILS

## 7.1 LACK OF UPPER LIMIT RESTRICTION FOR DELAY

// LOW

### Description

In the `Timelock` contract, the `delay` variable is used to set the delay for timelock transactions, established in the constructor. While the `delay` requires a minimum of 3 days, there is no upper limit set for the delay.

```
28  constructor(
29      uint256 delay,
30      address[] memory proposers,
31      address[] memory executors,
32      address canceller
33  )
34      TimelockController(delay, proposers, executors, address(0))
35  {
36      if (delay < 3 days) revert InvalidDelay();
37      if (canceller == address(0)) revert InvalidCanceller();
38      if (proposers.length == 0) revert InvalidProposer();
39      if (executors.length == 0) revert InvalidExecutor();
40      _grantRole(CANCELLER_ROLE, canceller);
41  }
```

This could lead to potential unwanted consequences, as an excessively long delay could prevent the timelock from being executed in a timely manner.

### BVSS

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

### Recommendation

Set an upper limit for the delay to prevent potential excessive delays.

### Remediation

**SOLVED:** The **WalletConnect team** solved this finding in commit `3bb4b4daf63c49dfba95369aef061539a8e0c557` by following the mentioned recommendation.

### Remediation Hash

https://github.com/WalletConnectFoundation/contracts/commit/3bb4b4daf63c49dfba95369aef061539a8e0c557

### References

WalletConnectFoundation/contracts/src/Timelock.sol#L28

## 7.2 MISSING _DISABLEINITIALIZERS() FUNCTION CALL

// LOW

### Description

The `WCT` contract is upgradeable, inheriting from the `Initializable` module from OpenZeppelin. In order to prevent leaving the contracts uninitialized OpenZeppelin's documentation recommends adding the `_disableInitializers` function in the constructor to automatically lock the contracts when they are deployed. However, the contract is missing this function call.
This omission can lead to potential security vulnerabilities, as an uninitialized implementation contract can be taken over by an attacker, which may impact the proxy.

### BVSS

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

### Recommendation

Consider adding a constructor and calling the `_disableinitializers()` method within it to prevent the implementation from being initialized.

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
  _disableInitializers();
}
```

### Remediation

**SOLVED:** The **WalletConnect team** solved this finding in commit `8ab9473797368d348f0f77fb55828e330513a909` by following the mentioned recommendation.

### Remediation Hash

https://github.com/WalletConnectFoundation/contracts/commit/8ab9473797368d348f0f77fb55828e330513a909

### References

## 7.3 SINGLE STEP OWNERSHIP TRANSFER RISKS DOS FOR PRIVILEGED FUNCTIONS

// INFORMATIONAL

### Description

The `WCT` contract inherits from OpenZeppelin's OwnableUpgradeable contract and allows for single-step ownership transfer via the `transferOwnership()` function. In this aspect, it is crucial that the address to which ownership is transferred is verified to be active and willing to assume ownership responsibilities. Otherwise, the contract could be locked in a situation where it is no longer possible to make administrative changes to it.
Additionally, in the `OwnableUpgradeable` contract, the `renounceOwnership()` function allows renouncing to the owner permission. Renouncing ownership before transferring it would result in the contract having no owner, eliminating the ability to call privileged functions.

### BVSS

https://www.halborn.com/bvss?q=AO:S/AC:L/AX:L/R:N/S:U/C:N/A:H/I:H/D:N/Y:N (1.8)

### Recommendation

Consider using OpenZeppelin's `Ownable2StepUpgradeable` contract over the `OwnableUpgradeable` implementation. `Ownable2StepUpgradeable` provides a two-step ownership transfer process, which adds an extra layer of security to prevent accidental ownership transfers.
Additionally, it is recommended that the owner cannot call the `renounceOwnership()` function to avoid losing ownership of the contract.

### Remediation

**ACKNOWLEDGED:** The **WalletConnect team** made a business decision to acknowledge this finding and not alter the contracts.

### References

WalletConnectFoundation/contracts/src/WCT.sol#L17

## 7.4 RISK OF EVM VERSION INCOMPATIBILITY ACROSS CHAINS

// INFORMATIONAL

### Description

From Solidity versions `0.8.20` through `0.8.24`, the default target EVM version is set to `Shanghai`, which results in the generation of bytecode that includes `PUSH0` opcodes. Starting with version `0.8.25`, the default EVM version shifts to `Cancun`, introducing new opcodes for transient storage, `TSTORE` and `TLOAD`.
In this aspect, it is crucial to select the appropriate EVM version when it's intended to deploy the contracts on networks other than the Ethereum mainnet, which may not support these opcodes. Failure to do so could lead to unsuccessful contract deployments or transaction execution issues.

### BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:L/A:L/I:L/D:N/Y:N (0.8)

### Recommendation

Make sure to specify the target EVM version when using Solidity versions from `0.8.20` and above if deploying to chains that may not support newly introduced opcodes. Additionally, it is crucial to stay informed about the opcode support of different chains to ensure smooth deployment and compatibility.

### Remediation

**ACKNOWLEDGED:** The **WalletConnect team** made a business decision to acknowledge this finding and not alter the contracts.

### References

WalletConnectFoundation/contracts/src/WCT.sol
WalletConnectFoundation/contracts/src/L2WCT.sol
WalletConnectFoundation/contracts/src/Timelock.sol

## 7.5 LACK OF FLEXIBILITY FOR ENABLING/DISABLING TRANSFER RESTRICTIONS

// INFORMATIONAL

### Description

In the `L2WCT` contract, the `disableTransferRestrictions()` function is used to disable transfer restrictions. However, the contract does not provide flexibility for re-enabling these restrictions. The function can only be called once, and once it is called, it cannot be called again. This lack of flexibility may limit the ability to manage transfer restrictions effectively.

```
169 function disableTransferRestrictions() external onlyRole(DEFAULT_ADMIN_ROLE) {
170     if (transferRestrictionsDisabledAfter != type(uint256).max) {
171         revert TransferRestrictionsAlreadyDisabled();
172     }
173     transferRestrictionsDisabledAfter = 0;
174     emit TransferRestrictionsDisabled();
175 }
```

### BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:M/I:M/D:N/Y:N (1.3)

### Recommendation

Consider adding a function to enable transfer restrictions after they have been disabled. This will provide more flexibility in managing transfer restrictions.

### Remediation

**ACKNOWLEDGED:** The **WalletConnect team** made a business decision to acknowledge this finding and not alter the contracts.

### References

WalletConnectFoundation/contracts/src/L2WCT.sol#L169

# STATIC ANALYSIS REPORT

## Description

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.
The security team assessed all findings identified by the Slither software, however, findings with related to external dependencies are not included in the below results for the sake of report readability.

## Output

The findings obtained as a result of the Slither scan were reviewed, and were not included in the report because they were determined as false positives.

```
TimelockController._execute(address,uint256,bytes) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#412-415) sends eth to arbitrary user
        Dangerous calls:
        - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
TimelockController.getOperationState(bytes32) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#207-218) uses a dangerous strict equality:
        - timestamp == 0 (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#209)
TimelockController.getOperationState(bytes32) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#207-218) uses a dangerous strict equality:
        - timestamp == _DONE_TIMESTAMP (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
TimelockController._execute(address,uint256,bytes) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#412-415) ignores return value by Address.verifyCallResult(success,returndata) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
TimelockController._execute(address,uint256,bytes) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#412-415) has external calls inside a loop: (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in TimelockController.execute(address,uint256,bytes,bytes32,bytes32) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#358-371):
        External calls:
        - _execute(target,value,payload) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#368)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#413)
        Event emitted after the call(s):
        - CallExecuted(id,0,target,value,payload) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#369)
Reentrancy in TimelockController.executeBatch(address[],uint256[],bytes[],bytes32,bytes32) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#385-407):
        External calls:
        - _execute(target,value,payload) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#403)
                - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#413)
        Event emitted after the call(s):
        - CallExecuted(id,i,target,value,payload) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
TimelockController.getOperationState(bytes32) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#207-218) uses timestamp for comparisons
        Dangerous comparisons:
        - timestamp == 0 (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#209)
        - timestamp == _DONE_TIMESTAMP (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#211)
        - timestamp > block.timestamp (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#213)
Votes.delegateBySig(address,uint256,uint8,bytes32,bytes32) (lib/openzeppelin-contracts/contracts/governance/utils/Votes.sol#142-161) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > expiry (lib/openzeppelin-contracts/contracts/governance/utils/Votes.sol#158)
ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (lib/openzeppelin-contracts/contracts/token/ERC20/extensions/ERC20Permit.sol#44-67) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > deadline (lib/openzeppelin-contracts/contracts/token/ERC20/extensions/ERC20Permit.sol#53)
Time._getFullAt(Time.Delay,uint48) (lib/openzeppelin-contracts/contracts/utils/types/Time.sol#74-77) uses timestamp for comparisons
        Dangerous comparisons:
        - effect ≤ timepoint (lib/openzeppelin-contracts/contracts/utils/types/Time.sol#76)
VotesUpgradeable.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (lib/openzeppelin-contracts-upgradeable/contracts/governance/utils/VotesUpgradeable.sol#165-184) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > expiry (lib/openzeppelin-contracts-upgradeable/contracts/governance/utils/VotesUpgradeable.sol#173)
ERC20PermitUpgradeable.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (lib/openzeppelin-contracts-upgradeable/contracts/token/ERC20/extensions/ERC20PermitUpgradeable.sol#49-72) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > deadline (lib/openzeppelin-contracts-upgradeable/contracts/token/ERC20/extensions/ERC20PermitUpgradeable.sol#58)
L2CNKT._update(address,address,uint256) (src/L2CNKT.sol#200-209) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp ≤ transferRestrictionsDisabledAfter (src/L2CNKT.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
INFO:Detectors:
Low level call in TimelockController._execute(address,uint256,bytes) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#412-415):
        - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/governance/TimelockController.sol#413)
Low level call in SafeERC20._callOptionalReturnBool(IERC20,bytes) (lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#110-117):
        - (success,returndata) = address(token).call(data) (lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#115)
Low level call in Address.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts/utils/Address.sol#41-50):
        - (success,None) = recipient.call{value: amount}() (lib/openzeppelin-contracts/contracts/utils/Address.sol#46)
Low level call in Address.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts/utils/Address.sol#83-89):
        - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#87)
Low level call in Address.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts/utils/Address.sol#95-98):
        - (success,returndata) = target.staticcall(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#96)
Low level call in Address.functionDelegateCall(address,bytes) (lib/openzeppelin-contracts/contracts/utils/Address.sol#104-107):
        - (success,returndata) = target.delegatecall(data) (lib/openzeppelin-contracts/contracts/utils/Address.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Slither:. analyzed (63 contracts with 94 detectors), 20 result(s) found
```

# UNIT TESTS AND FUZZING

The original repository used the **Foundry** environment to develop and test the smart contracts. All tests were executed successfully.

```
Ran 4 tests for test/integration/concrete/timelock/execute/execute.t.sol:Execute_Timelock_Integration_Concrete_Test
[PASS] test_Execute() (gas: 102895)
[PASS] test_RevertGiven_OperationIsNotReady() (gas: 65329)
[PASS] test_RevertWhen_CallerIsNotExecutor() (gas: 26517)
[PASS] test_RevertWhen_NoOperationIsScheduled() (gas: 30757)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 15.03ms (485.98µs CPU time)

Ran 2 tests for test/unit/concrete/brr/mint/mint.t.sol:Mint_BRR_Unit_Concrete_Test
[PASS] test_Mint() (gas: 121639)
[PASS] test_RevertWhen_CallerNotOwner() (gas: 18792)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 15.09ms (250.30µs CPU time)

Ran 1 test for test/unit/concrete/brr/init.t.sol:Init_BRR_Unit_Concrete_Test
[PASS] test_Init() (gas: 3284573)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 15.90ms (1.29ms CPU time)

Ran 5 tests for test/unit/concrete/l2brr/constructor.t.sol:Constructor_L2BRR_Unit_Concrete_Test
[PASS] test_constructor() (gas: 45727)
[PASS] test_revertWhen_BridgeIsZero() (gas: 91596)
[PASS] test_revertWhen_InitialAdminIsZero() (gas: 91609)
[PASS] test_revertWhen_InitialManagerIsZero() (gas: 91665)
[PASS] test_revertWhen_RemoteTokenIsZero() (gas: 91588)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 16.06ms (2.83ms CPU time)

Ran 2 tests for test/unit/concrete/permissioned-node-registry/is-node-whitelisted/isNodeWhitelisted.t.sol:IsNodeWhitelisted_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_GivenANodeIsNotWhitelisted() (gas: 12807)
[PASS] test_GivenANodesIsWhitelisted() (gas: 84956)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 16.03ms (169.17µs CPU time)

Ran 4 tests for test/integration/concrete/timelock/cancel/cancel.t.sol:Cancel_Timelock_Integration_Concrete_Test
[PASS] test_GivenOperationIsPending() (gas: 47308)
[PASS] test_RevertGiven_OperationDoesNotExist() (gas: 16538)
[PASS] test_RevertGiven_OperationIsAlreadyExecuted() (gas: 100866)
[PASS] test_RevertWhen_CallerIsNotProposerOrCanceller() (gas: 17130)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 16.50ms (502.25µs CPU time)

Ran 4 tests for test/integration/concrete/staking/claim-rewards/claimRewards.t.sol:ClaimRewards_Staking_Integration_Concrete_Test
[PASS] test_GivenRewardsVaultHasEnoughBalance() (gas: 414170)
[PASS] test_RevertGivenCallerHasNoPendingRewards() (gas: 20437)
[PASS] test_RevertGiven_RewardsVaultDoesntHaveEnoughBalance() (gas: 414364)
[PASS] test_RevertGiven_stakingHasNoAllowanceFromRewardsVault() (gas: 388431)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 17.67ms (2.02ms CPU time)

Ran 2 tests for test/unit/concrete/permissioned-node-registry/get-whitelisted-node-at-index/getWhitelistedNodeAtIndex.t.sol:GetWhitelistedNodeAtIndex_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_GivenThereIsANodeAtASpecificIndex() (gas: 181770)
[PASS] test_RevertGiven_ThereIsNoNodeAtASpecificIndex() (gas: 10537)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.84ms (160.28µs CPU time)

Ran 5 tests for test/unit/concrete/timelock/constructor.t.sol:Constructor_Timelock_Unit_Concrete_Test
[PASS] test_RevertWhen_CancellerIsAddressZero() (gas: 161756)
[PASS] test_RevertWhen_DelayIsLessThan3Days() (gas: 161667)
[PASS] test_RevertWhen_ExecutorsArrayIsEmpty() (gas: 136675)
[PASS] test_RevertWhen_ProposersArrayIsEmpty() (gas: 111977)
[PASS] test_WhenAllParametersAreValid() (gas: 1808104)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 4.05ms (829.87µs CPU time)
```

```
Ran 2 tests for test/unit/concrete/l2brr/set-allowed-from/setAllowedFrom.t.sol:SetAllowedFrom_L2BRR_Unit_Concrete_Test
[PASS] test_RevertWhen_CallerNotManager() (gas: 16906)
[PASS] test_SetAllowedFrom() (gas: 42307)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 5.38ms (166.01µs CPU time)

Ran 2 tests for test/unit/concrete/l2brr/set-allowed-to/setAllowedTo.t.sol:SetAllowedTo_L2BRR_Unit_Concrete_Test
[PASS] test_RevertWhen_CallerNotManager() (gas: 16883)
[PASS] test_SetAllowedTo() (gas: 42303)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.46ms (115.61µs CPU time)

Ran 6 tests for test/integration/concrete/staking/unstake/unstake.t.sol:Unstake_Staking_Integration_Concrete_Test
[PASS] test_RevertGiven_StakingIsPaused() (gas: 66386)
[PASS] test_RevertWhen_UnstakeAmountIsGTCallerStake() (gas: 296704)
[PASS] test_RevertWhen_UnstakeAmountIsZero() (gas: 41873)
[PASS] test_RevertWhen_UnstakeAmountWillPutCallerStakeBelowMinimum() (gas: 298212)
[PASS] test_WhenUnstakeAmountEqCallerStakes() (gas: 295660)
[PASS] test_WhenUnstakeAmountWillNotPutCallerStakeBelowMinimum() (gas: 335137)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 6.78ms (1.67ms CPU time)

Ran 3 tests for test/integration/concrete/l2-brr/mint/mint.t.sol:Mint_L2BRR_Integration_Concrete_Test
[PASS] test_MintWhenSupplyNotExceedMax() (gas: 144507)
[PASS] test_RevertWhen_CallerNotBridge() (gas: 11023)
[PASS] test_RevertWhen_SupplyExceedsMax() (gas: 77122)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 6.99ms (374.14µs CPU time)

Ran 2 tests for test/unit/concrete/permissioned-node-registry/get-whitelisted-nodes/getWhitelistedNodes.t.sol:GetWhitelistedNodes_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_GivenNoNodesHaveBeenWhitelisted() (gas: 11301)
[PASS] test_GivenNodesHaveBeenWhitelisted() (gas: 184985)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.68ms (191.70µs CPU time)

Ran 3 tests for test/unit/concrete/l2brr/disable-transfer-restrictions/disableTransferRestrictions.t.sol:DisableTransferRestrictions_L2BRR_Unit_Concrete_Test
[PASS] test_RevertGiven_RestrictionsDisabled() (gas: 15340)
[PASS] test_RevertWhen_CallerNotAdmin() (gas: 14641)
[PASS] test_SetTransferRestrictionsDisabled() (gas: 16159)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.85ms (179.41µs CPU time)

Ran 3 tests for test/unit/concrete/staking/update-min-stake-amount/updateMinStakeAmount.t.sol:UpdateMinStakeAmount_Staking_Unit_Concrete_Test
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 18526)
[PASS] test_RevertWhen_NewMinStakeAmountIsSameAsOld() (gas: 21257)
[PASS] test_WhenNewMinStakeAmountIsDifferentFromOld() (gas: 28771)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.34ms (193.68µs CPU time)

Ran 6 tests for test/unit/concrete/l2brr/grant-role/grantRole.t.sol:GrantRole_L2BRR_Unit_Concrete_Test
[PASS] test_RevertWhen_AttackerNotDefaultAdmin() (gas: 22574)
[PASS] test_RevertWhen_CallerNotDefaultAdmin() (gas: 20609)
[PASS] test_WhenGrantingDefaultAdminRole() (gas: 50191)
[PASS] test_WhenGrantingDefaultAdminRoleToExistingHolder() (gas: 19463)
[PASS] test_WhenGrantingManagerRole() (gas: 50126)
[PASS] test_WhenGrantingManagerRoleToExistingHolder() (gas: 23506)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 4.57ms (360.80µs CPU time)

Ran 2 tests for test/Counter.t.sol:CounterTest
[PASS] testFuzz_SetNumber(uint256) (runs: 1000, µ: 30961, ~: 31240)
[PASS] test_Increment() (gas: 31249)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 36.61ms (30.33ms CPU time)
```

```
Ran 4 tests for test/integration/concrete/staking/update-rewards/updateRewards.t.sol:UpdateRewards_Staking_Integration_Concrete_Test
[PASS] test_GivenNodeHasNotStakedMinAmount() (gas: 95930)
[PASS] test_GivenStakingIsNotPaused() (gas: 384462)
[PASS] test_RevertGiven_StakingIsPaused() (gas: 381538)
[PASS] test_RevertWhen_CallerIsNotRewardManager() (gas: 34769)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 10.43ms (780.99µs CPU time)

Ran 6 tests for test/integration/concrete/l2-brr/burn/burn.t.sol:Burn_L2BRR_Integration_Concrete_Test
[PASS] test_Burn_WhenFromWhitelisted() (gas: 100559)
[PASS] test_Burn_WhenToWhitelisted() (gas: 73596)
[PASS] test_Burn_WhenTransferabilityOn() (gas: 46291)
[PASS] test_RevertWhen_CallerNotBridge() (gas: 11101)
[PASS] test_RevertWhen_FromNotWhitelisted() (gas: 42343)
[PASS] test_RevertWhen_FromNotWhitelistedAndToNotWhitelisted() (gas: 19828)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 3.35ms (389.59µs CPU time)

Ran 4 tests for test/unit/concrete/permissioned-node-registry/whitelist-node/whitelistNode.t.sol:WhitelistNode_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_GivenWhitelistCountLTMaxNodes() (gas: 96800)
[PASS] test_RevertGiven_NodeIsAlreadyWhitelisted() (gas: 86255)
[PASS] test_RevertGiven_WhitelistCountEqMaxNodes() (gas: 2460675)
[PASS] test_RevertWhen_CallerNotOwner() (gas: 14834)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 4.02ms (1.05ms CPU time)

Ran 1 test for test/unit/fuzz/l2-brr/setAllowedTo.t.sol:SetAllowedTo_L2BRR_Unit_Fuzz_Test
[PASS] testFuzz_setAllowedTo(address,address,bool) (runs: 1000, µ: 15061, ~: 15058)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 35.65ms (32.80ms CPU time)

Ran 7 tests for test/integration/concrete/reward-manager/post-performance-records/postPerformanceRecords.t.sol:PostPerformanceRecords_RewardManager_Integration_Concrete_Test
[PASS] testGasProfiling_PostPerformanceRecords() (gas: 8008666)
[PASS] test_PostPerformanceRecords_givenMaxNodesAreActive() (gas: 7483384)
[PASS] test_PostPerformanceRecords_givenOnlyOneNodeIsActive() (gas: 379819)
[PASS] test_RevertWhen_CallerNotOwner() (gas: 21267)
[PASS] test_RevertWhen_InputArraysDontMatchInLength() (gas: 23331)
[PASS] test_RevertWhen_PerformanceSumIsZero() (gas: 24131)
[PASS] test_RevertWhen_ReportingEpochLTELastUpdatedEpoch() (gas: 21769)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 58.26ms (39.62ms CPU time)

Ran 4 tests for test/unit/concrete/permissioned-node-registry/set-max-nodes/setMaxNodes.t.sol:SetMaxNodes_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_RevertWhen_CallerNotOwner() (gas: 17650)
[PASS] test_RevertWhen_NewMaxNodesEqCurrentMaxNodes() (gas: 16576)
[PASS] test_RevertWhen_NewMaxNodesIsLTWhitelistCount() (gas: 2408634)
[PASS] test_WhenNewMaxNodesIsGTWhitelistCount() (gas: 25096)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 2.99ms (748.85µs CPU time)

Ran 1 test for test/unit/fuzz/l2-brr/setAllowedFrom.t.sol:SetAllowedFrom_L2BRR_Unit_Fuzz_Test
[PASS] testFuzz_setAllowedFrom(address,address,bool) (runs: 1000, µ: 15081, ~: 15081)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 49.13ms (44.89ms CPU time)

Ran 3 tests for test/unit/concrete/permissioned-node-registry/remove-node-from-whitelist/removeNodeFromWhitelist.t.sol:RemoveNodeFromWhitelist_PermissionedNodeRegistry_Unit_Concrete_Test
[PASS] test_GivenTheNodeIsInTheWhitelist() (gas: 71374)
[PASS] test_RevertGiven_TheNodeIsNotInTheWhitelist() (gas: 16140)
[PASS] test_RevertWhen_TheCallerIsNotTheOwner() (gas: 16756)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 2.43ms (143.06µs CPU time)
```

```
Ran 6 tests for test/unit/concrete/l2brr/revoke-role/revokeRole.t.sol:RevokeRole_L2BRR_Unit_Concrete_Test
[PASS] test_RevertWhen_AttackerNotDefaultAdmin() (gas: 22573)
[PASS] test_RevertWhen_CallerNotDefaultAdmin() (gas: 20586)
[PASS] test_WhenRevokingDefaultAdminRole() (gas: 24084)
[PASS] test_WhenRevokingDefaultAdminRoleFromNonHolder() (gas: 23511)
[PASS] test_WhenRevokingManagerRole() (gas: 28236)
[PASS] test_WhenRevokingManagerRoleFromNonHolder() (gas: 23511)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 2.50ms (231.32µs CPU time)

Ran 2 tests for test/unit/fuzz/wallet-connect-token/mint.t.sol:Mint_BRR_Unit_Fuzz_Test
[PASS] testFuzz_Mint(address,uint256) (runs: 1000, µ: 121285, ~: 121182)
[PASS] testFuzz_RevertWhen_CallerNotOwner(address) (runs: 1000, µ: 14978, ~: 14978)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 128.36ms (169.58ms CPU time)

Ran 3 tests for test/unit/fuzz/permissioned-node-registry/setMaxNodes.t.sol:SetMaxNodes_PermissionedNodeRegistry_Unit_Fuzz_Test
[PASS] testFuzz_RevertWhen_CallerNotOwner(address,uint8) (runs: 1000, µ: 19234, ~: 19234)
[PASS] testFuzz_RevertWhen_NewMaxNodesIsLTWhitelistCount(uint8,uint8) (runs: 1000, µ: 1046627, ~: 933373)
[PASS] testFuzz_WhenNewMaxNodesIsGTWhitelistCount(uint8) (runs: 1000, µ: 24857, ~: 25549)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 466.41ms (461.31ms CPU time)

Ran 3 tests for test/unit/concrete/staking/set-staking-allowlist/setStakingAllowlist.t.sol:SetStakingAllowlist_Staking_Unit_Concrete_Test
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 18569)
[PASS] test_RevertWhen_NewValueIsSameAsOld() (gas: 21410)
[PASS] test_WhenNewValueIsDifferentFromOld() (gas: 23594)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 2.61ms (144.73µs CPU time)

Ran 7 tests for test/integration/concrete/staking/stake/stake.t.sol:Stake_Staking_Integration_Concrete_Test
[PASS] test_GivenCallerHasEnoughBalance() (gas: 302898)
[PASS] test_GivenCallerHasEnoughBalanceAndAllowlistIsOff() (gas: 222578)
[PASS] test_RevertGiven_CallerHasNotApprovedStakingAmount() (gas: 177051)
[PASS] test_RevertGiven_CallerHasNotEnoughBalance() (gas: 203013)
[PASS] test_RevertGiven_StakingIsPaused() (gas: 66335)
[PASS] test_RevertWhen_AmountToStakeLTMinStake() (gas: 134216)
[PASS] test_RevertWhen_CallerIsNotAPermissionedNode() (gas: 49236)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 3.34ms (881.41µs CPU time)

Ran 6 tests for test/unit/concrete/l2brr/transfer/transfer.t.sol:Transfer_L2BRR_Unit_Concrete_Test
[PASS] test_RevertWhen_RestrictionsDisabledButInsufficientBalance() (gas: 23097)
[PASS] test_RevertWhen_SenderInAllowedFromListButInsufficientBalance() (gas: 48351)
[PASS] test_RevertWhen_SenderNotAllowedAndRecipientNotAllowed() (gas: 19794)
[PASS] test_TransferWhen_RecipientInAllowedToList() (gas: 89585)
[PASS] test_TransferWhen_RestrictionsDisabled() (gas: 62196)
[PASS] test_TransferWhen_SenderInAllowedFromList() (gas: 87439)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 2.89ms (377.09µs CPU time)

Ran 3 tests for test/unit/fuzz/l2-brr/transfer.t.sol:Transfer_L2BRR_Unit_Fuzz_Test
[PASS] testFuzz_AllowedFromCanSendAnywhere(address) (runs: 1000, µ: 178500, ~: 178500)
[PASS] testFuzz_DisableTransferRestrictions(address,address) (runs: 1000, µ: 165954, ~: 165954)
[PASS] testFuzz_NotAllowedFromCannotSendIfNoAllowedTos(address,address) (runs: 1000, µ: 125615, ~: 125615)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 265.53ms (263.12ms CPU time)

Ran 1 test for test/unit/fuzz/reward-manager/postPerformanceRecords.t.sol:PostPerformanceRecords_RewardManager_Unit_Fuzz_Test
[PASS] testFuzz_postPerformanceRecords(uint256[]) (runs: 1000, µ: 3909496, ~: 3982163)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 18.38s (18.38s CPU time)

Ran 2 tests for test/unit/fuzz/permissioned-node-registry/whitelistNode.t.sol:WhitelistNode_PermissionedNodeRegistry_Unit_Fuzz_Test
[PASS] testFuzz_RevertGiven_NodeIsAlreadyWhitelisted(address) (runs: 1000, µ: 88728, ~: 88848)
[PASS] testFuzz_RevertWhen_CallerNotOwner(address) (runs: 1000, µ: 15963, ~: 15963)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 67.64s (95.41ms CPU time)

Ran 7 tests for test/invariant/L2BRR.t.sol:L2BRR_Invariant_Test
[PASS] invariant_allowedFromAndToConsistent() (runs: 256, calls: 128000, reverts: 20885)
[PASS] invariant_balanceSum() (runs: 256, calls: 128000, reverts: 20600)
[PASS] invariant_callSummary() (runs: 256, calls: 128000, reverts: 20602)
[PASS] invariant_mintsMinusBurnsSumUpToTotalSupply() (runs: 256, calls: 128000, reverts: 20806)
[PASS] invariant_remoteTokenAndBridgeNeverChange() (runs: 256, calls: 128000, reverts: 20703)
[PASS] invariant_totalSupplyNeverExceedsMaxSupply() (runs: 256, calls: 128000, reverts: 20734)
[PASS] invariant_transferRestrictionsEnforced() (runs: 256, calls: 128000, reverts: 20625)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 148.25s (515.33s CPU time)

Ran 4 tests for test/invariant/BRR.t.sol:BRR_Invariant_Test
[PASS] invariant_balancesSumUpToTotalSupply() (runs: 256, calls: 128000, reverts: 1645)
[PASS] invariant_callSummary() (runs: 256, calls: 128000, reverts: 1859)
[PASS] invariant_mintsMinusBurnsSumUpToTotalSupply() (runs: 256, calls: 128000, reverts: 1887)
[PASS] invariant_totalSupplyNeverExceedsMaxSupply() (runs: 256, calls: 128000, reverts: 1827)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 187.87s (376.42s CPU time)

Ran 37 test suites in 187.88s (423.38s CPU time): 132 tests passed, 0 failed, 0 skipped (132 total tests)
```

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.