OMNISCIA

November 1, 2024

SMART CONTRACT AUDIT REPORT

Sophon

Support Infrastructure


omniscia.io


info@omniscia.io


Online report: sophon-support-infrastructure

Omniscia.io is one of the fastest growing and most trusted blockchain security firms and has rapidly become a true market leader. To date, our team has collectively secured over 370+ clients, detecting 1,500+ high-severity issues in widely adopted smart contracts.

Founded in France at the start of 2020, and with a track record spanning back to 2017, our team has been at the forefront of auditing smart contracts, providing expert analysis and identifying potential vulnerabilities to ensure the highest level of security of popular smart contracts, as well as complex and sophisticated decentralized protocols.

Our clients, ecosystem partners, and backers include leading ecosystem players such as L'Oréal, Polygon, AvaLabs, Gnosis, Morpho, Vesta, Gravita, Olympus DAO, Fetch.ai, and LimitBreak, among others.

To keep up to date with all the latest news and announcements follow us on twitter @omniscia_sec.


omniscia.io


info@omniscia.io

Online report: sophon-support-infrastructure

Support Infrastructure Security Audit

Audit Report Revisions

Commit Hash	Date	Audit Report Hash
518b2dbbc7	October 15th 2024	812fd6d8e6
1edaebf4ac	October 30th 2024	532143f481
c2b7d50827	November 1st 2024	7aa0d98465

Audit Overview

We were tasked with performing an audit of the Sophon codebase and in particular their Support Infrastructure w/ an NFT implementation, an **EIP-20** token, and a `BridgeHub` integrator module.

Over the course of the audit, we identified issues in the NFT implementation that could result in a sabotage of a whitelisted user's allocation decrease as well as an unexpected mid-execution re-entrancy attack during the custom batch transfer procedure implemented in the NFT.

Additionally, we would like to outline that the contracts of the system appear to be upgradeable yet inherit from non-upgradeable variants and make use of index-based storage. We strongly advise against this approach, and we recommend approaches such as the **EIP-7201** standard namespaced layout to be followed.

We advise the Sophon team to closely evaluate all minor-and-above findings identified in the report and promptly remediate them as well as consider all optimizational exhibits identified in the report.

Post-Audit Conclusion

The Sophon team iterated through all findings within the report and provided us with a revised commit hash to evaluate all exhibits on.

We evaluated all alleviations performed by Sophon and have identified that a single exhibit has not been adequately dealt with even though it was marked as alleviated explicitly by the Sophon team.

We advise the Sophon team to revisit the following exhibit: `GNF-01S`

Post-Audit Conclusion (c2b7d50827)

The Sophon team provided us with a follow-up commit to evaluate the remediative actions for `GNF-01S` on.

We confirmed that `GNF-01S` was properly resolved and did not evaluate the remaining code changes that were associated with the updated commit hash.

We consider all outputs of the audit report properly consumed by the Sophon team with no further remediative actions remaining.

Audit Synopsis

Severity	Identified	Alleviated	Partially Alleviated	Acknowledged
Unknown	2	0	0	2
Informational	12	10	0	2
Minor	7	3	0	4
Medium	2	2	0	0
Major	0	0	0	0

During the audit, we filtered and validated a total of **6 findings utilizing static analysis** tools as well as identified a total of **17 findings during the manual review** of the codebase. We strongly recommend that any minor severity or higher findings are dealt with promptly prior to the project's launch as they can introduce potential misbehaviours of the system as well as exploits.

Scope

The audit engagement encompassed a specific list of contracts that were present in the commit hash of the repository that was in scope. The tables below detail certain meta-data about the target of the security assessment and a navigation chart is present at the end that links to the relevant findings per file.

Target

- Repository: **https://github.com/sophon-org/support-contracts**
- Commit: 518b2dbbc7cea573b228a8637a48aba0fa4b94a3
- Language: Solidity
- Network: Sophon
- Revisions: **518b2dbbc7**, **1edaebf4ac**, **c2b7d50827**

Contracts Assessed

File	Total Finding(s)
contracts/paymaster-on-l1/BridgeHubWrapper.sol (BHW)	4
contracts/paymaster-on-l1/BridgeHubWrapperProxy.sol (BHP)	0
contracts/tokens/GuardianNFT.sol (GNF)	6
contracts/tokens/GuardianNFTProxy.sol (GNT)	0
contracts/tokens/GuardianNFTState.sol (GNS)	1
contracts/tokens/delegation/GuardianDelegation.sol (GDN)	4
contracts/tokens/delegation/GuardianDelegationProxy.sol (GDP)	0
contracts/tokens/delegation/GuardianDelegationState.sol (GDS)	1
contracts/proxies/Proxy2Step.sol (PSP)	2
contracts/proxies/ProxyAccessControl.sol (PAC)	2

contracts/tokens/SophonToken.sol (STN)	0
contracts/proxies/Upgradeable2Step.sol (USP)	2
contracts/proxies/UpgradeableAccessControl.sol (UAC)	1

Compilation

The project utilizes `foundry` as its development pipeline tool, containing an array of tests and scripts coded in Solidity.

To compile the project, the `build` command needs to be issued via the `forge` CLI tool:

```bash
forge build
```

The `forge` tool automatically selects Solidity version `0.8.26` based on the version specified within the `foundry.toml` file.

The project contains discrepancies with regards to the Solidity version used, however, they are solely located in external dependencies and can thus be safely ignored.

The `pragma` statements have been locked to `0.8.26` (`=0.8.26`), the same version utilized for our static analysis as well as optimizational review of the codebase.

During compilation with the `foundry` pipeline, no errors were identified that relate to the syntax or bytecode size of the contracts.

Static Analysis

The execution of our static analysis toolkit identified **61 potential issues** within the codebase of which **52 were ruled out to be false positives** or negligible findings.

The remaining **9 issues** were validated and grouped and formalized into the **6 exhibits** that follow:

ID	Severity	Addressed	Title
BHW-01S	Informational	Yes	Inexistent Event Emissions
BHW-02S	Informational	Yes	Inexistent Sanitization of Input Addresses
GDN-01S	Informational	Yes	Inexistent Sanitization of Input Address
GNF-01S	Informational	Yes	Inexistent Event Emissions
PSP-01S	Informational	Yes	Inexistent Sanitization of Input Address
PAC-01S	Informational	Yes	Inexistent Sanitization of Input Address

Manual Review

A **thorough line-by-line review** was conducted on the codebase to identify potential malfunctions and vulnerabilities in Sophon's token, NFT, and `BridgeHub` integrator.

As the project at hand implements multiple infrastructure components, intricate care was put into ensuring that the **flow of funds within the system conforms to the specifications and restrictions** laid forth within the protocol's specification.

We validated that **all state transitions of the system occur within sane criteria** and that all rudimentary formulas within the system execute as expected. We **pinpointed two distinct vulnerabilities** within the system which could have had **moderate ramifications** to its overall operation; we urge the Sophon team to promptly evaluate and remediate the relevant medium-severity exhibits of the audit report.

Additionally, the system was investigated for any other commonly present attack vectors such as re-entrancy attacks, mathematical truncations, logical flaws and **ERC / EIP** standard inconsistencies. The documentation of the project was satisfactory to a certain extent, however, we strongly recommend it to be expanded in how the `BridgeHub` relayed transactions are precisely expected to be executed.

A total of **17 findings** were identified over the course of the manual review of which **12 findings** concerned the behaviour and security of the system. The non-security related findings, such as optimizations, are included in the separate **Code Style** chapter.

The finding table below enumerates all these security / behavioural findings:

ID	Severity	Addressed	Title
BHW-01M	Informational	Yes	Inconsistent Handling of Native & Token Funds
BHW-02M	Minor	Acknowledged	Inexplicable Capability of Re-Initialization
GDN-01M	Minor	Yes	Improper Immutable Domain Separator
GNF-01M	Minor	Acknowledged	Inexplicable Capability of Re-Initialization
GNF-02M	Minor	Acknowledged	Upgradeable Incompatibility of Dependency

GNF-03M	Medium	Yes	Incorrect Batch Transfer Implementation
GNF-04M	Medium	Yes	Incorrect Decrease of Whitelist Count
PSP-01M	Minor	Yes	Improper Proxy Pattern
PAC-01M	Minor	Yes	Improper Proxy Pattern
USP-01M	Unknown	Acknowledged	Invalid Upgradeability Support
USP-02M	Minor	Acknowledged	Inexplicable Acceptance Mechanism
UAC-01M	Unknown	Acknowledged	Invalid Upgradeability Support

Code Style

During the manual portion of the audit, we identified **5 optimizations** that can be applied to the codebase that will decrease the operational cost associated with the execution of a particular function and generally ensure that the project complies with the latest best practices and standards in Solidity.

Additionally, this section of the audit contains any opinionated adjustments we believe the code should make to make it more legible as well as truer to its purpose.

These optimizations are enumerated below:

ID	Severity	Addressed	Title
GDN-01C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
GDN-02C	Informational	Acknowledged	Inefficient `mapping` Lookups
GDS-01C	Informational	Acknowledged	Inefficient Data Structures
GNF-01C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
GNS-01C	Informational	Yes	Redundant Inheritance Chain

BridgeHubWrapper Static Analysis Findings

BHW-01S: Inexistent Event Emissions

Type	Severity	Location
Language Specific	Informational	BridgeHubWrapper.sol:L74-L77

Description:

The linked functions adjust sensitive contract variables yet do not emit an event for it.

Example:

contracts/paymaster-on-l1/BridgeHubWrapper.sol

SOL

```sol
74  function initialize(uint256 maxFee_, address refundRecipient_) external onlyOwner
{
75      maxFee = maxFee_;
76      refundRecipient = refundRecipient_;
77  }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted for each function to ensure off-chain processes can properly react to this system adjustment.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The `SetMaxFee`, and `SetRefundRecipient` events were introduced to the codebase and are correspondingly emitted in the **`BridgeHubWrapper::initialize`**, and **`BridgeHubWrapper::initialize`** functions respectively, addressing this exhibit in full.

BHW-02S: Inexistent Sanitization of Input Addresses

Type	Severity	Location
Input Sanitization	Informational	BridgeHubWrapper.sol:L62-L66

Description:

The linked function(s) accept `address` arguments yet do not properly sanitize them.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Example:

contracts/paymaster-on-l1/BridgeHubWrapper.sol

SOL

```sol
62  constructor(uint256 chainId_, IBridgeHub bridgeHub_, IERC20 sophonToken_) {
63      chainId = chainId_;
64      bridgeHub = bridgeHub_;
65      sophonToken = sophonToken_;
66  }
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that each `address` specified is non-zero.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

All input arguments of the `BridgeHubWrapper::constructor` function are adequately sanitized as non-zero in the latest in-scope revision of the codebase, addressing this exhibit.

GuardianDelegation Static Analysis Findings

GDN-01S: Inexistent Sanitization of Input Address

Type	Severity	Location
Input Sanitization	Informational	GuardianDelegation.sol:L81-L92

Description:

The linked function accepts an `address` argument yet does not properly sanitize it.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Example:

contracts/tokens/delegation/GuardianDelegation.sol

SOL

```sol
constructor(IERC721 guardianNFT_) {
    guardianNFT = guardianNFT_;
    DOMAIN_SEPARATOR = keccak256(
        abi.encode(
            DOMAIN_TYPEHASH,
            keccak256(bytes("GuardianDelegation")),
            keccak256(bytes("1")),
            block.chainid,
            address(this)
        )
```

Example (Cont.):

```sol
        );
    }
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that the `address` specified is non-zero.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The input `guardianNFT_` address argument of the **`GuardianDelegation::constructor`** function is adequately sanitized as non-zero in the latest in-scope revision of the codebase, addressing this exhibit.

GuardianNFT Static Analysis Findings

GNF-01S: Inexistent Event Emissions

Type	Severity	Location
Language Specific	Informational	GuardianNFT.sol:L172-L174, L253-L255

Description:

The linked functions adjust sensitive contract variables yet do not emit an event for it.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
172 function setBaseURI(string memory baseURI_) external onlyRole(DEFAULT_ADMIN_ROLE)
{
173     baseURI = baseURI_;
174 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted for each function to ensure off-chain processes can properly react to this system adjustment.

Alleviation (1edaebf4ac):

An event emission has been introduced for the former of the two referenced instances, addressing this exhibit partially.

Alleviation (c2b7d50827):

A `DelegationManagerSet` event was introduced and is now emitted in the **`GuardianNFT::setDelegationManager`** function, rendering this exhibit to be fully addressed.

Proxy2Step Static Analysis Findings

PSP-01S: Inexistent Sanitization of Input Address

Type	Severity	Location
Input Sanitization	Informational	Proxy2Step.sol:L17-L19

Description:

The linked function accepts an `address` argument yet does not properly sanitize it.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Example:

contracts/proxies/Proxy2Step.sol

SOL

```sol
17  constructor(address impl_) {
18      implementation = impl_;
19  }
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that the `address` specified is non-zero.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The input `impl_` address argument of the **`Proxy2Step::constructor`** function is adequately sanitized as non-zero in the latest in-scope revision of the codebase, addressing this exhibit.

ProxyAccessControl Static Analysis Findings

PAC-01S: Inexistent Sanitization of Input Address

Type	Severity	Location
Input Sanitization	Informational	ProxyAccessControl.sol:L19-L21

Description:

The linked function accepts an `address` argument yet does not properly sanitize it.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Example:

contracts/proxies/ProxyAccessControl.sol

SOL

```sol
19  constructor(address impl_, bytes memory initData_) {
20      replaceImplementation(impl_, initData_);
21  }
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that the `address` specified is non-zero.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

Input sanitization was introduced at the `UpgradeableAccessControl::replaceImplementation` function, alleviating this exhibit indirectly.

BridgeHubWrapper Manual Review Findings

BHW-01M: Inconsistent Handling of Native & Token Funds

Type	Severity	Location
Logical Fault	Informational	BridgeHubWrapper.sol:L86, L111-L114, L117, L119, L136, L152-L153, L156,

Description:

The `BridgeHubWrapper::requestL2TransactionTwoBridges` and `BridgeHubWrapper::requestL2TransactionDirect` functions do not appear to consistently handle transaction relays to the `BridgeHub`.

Specifically, if the `chainId` of the system points to an `ETH_TOKEN_ADDRESS` at the `BridgeHub` level interactions via the `BridgeHubWrapper::requestL2TransactionTwoBridges` will fail whereas interactions with a `msg.value` that does not match the `_request.mintValue` through the `BridgeHubWrapper::requestL2TransactionDirect` function will fail.

On the other hand, the `sharedBridge` (a distinct concept from the second bridge) during a `BridgeHubWrapper::requestL2TransactionTwoBridges` function call will not have been approved with the `mintValue` to process the transaction whereas all `BridgeHubWrapper::requestL2TransactionDirect` function calls will fail with a non-zero `msg.value`.

Example:

contracts/paymaster-on-l1/BridgeHubWrapper.sol

SOL

```sol
79  /**
80   * @notice Toggles the enabled status of a shared bridge.
81   * @dev Approves or revokes approval for the shared bridge to spend Sophon
tokens.
82   * @param sharedBridge_ The address of the shared bridge to toggle.
83   * @param isEnabled_ True to enable, false to disable the shared bridge.
84   */
85  function toggleSharedBridge(address sharedBridge_, bool isEnabled_) public onlyOwner
{
86      allowedBridges[sharedBridge_] = isEnabled_;
87      if (isEnabled_) {
88          sophonToken.forceApprove(sharedBridge_, type(uint256).max);
```

Example (Cont.):

```sol
89      } else {
90          sophonToken.forceApprove(sharedBridge_, 0);
91      }
92      emit ToggleSharedBridge(sharedBridge_, isEnabled_);
93  }
94
95  /**
96   * @notice Requests an L2 transaction using two bridges.
97   * @dev Validates the request parameters, handles token transfers, and interacts with the BridgeHub contract.
98   *      - Requires that the request's chainId matches this contract's chainId.
99   *      - Requires that the second bridge address is allowed.
100  *      - Ensures the correct amount of ETH is sent if needed.
101  *      - Validates that the mintValue is greater than or equal to l2Value, and the fee is within maxFee.
102  *      - Transfers the l2Value amount of Sophon tokens from the sender.
103  *      - Decodes the second bridge calldata and handles token transfers or ETH accordingly.
104  *      - Sets the refund recipient to the contract's refundRecipient.
105  *      - Calls the BridgeHub's requestL2TransactionTwoBridges function with the prepared request.
106  * @param _request The L2 transaction request parameters.
107  * @return canonicalTxHash The canonical transaction hash of the L2 transaction.
108  */
109 function requestL2TransactionTwoBridges(IBridgeHub.L2TransactionRequestTwoBridgesOuter memory _request) external payable returns (bytes32 canonicalTxHash) {
110     require(_request.chainId == chainId, "invalid chainId");
111     require(allowedBridges[_request.secondBridgeAddress], "invalid secondBridgeAddress");
```

Recommendation:

We advise the documentation of the project to be expanded so as to confirm the precise execution path expected, and we advise both functions to be corrected in how they validate their interactions to ensure they match the expected invocation scenario of the `BridgeHub`.

To note, the **`BridgeHubWrapper::toggleSharedBridge`** function is also incorrect as it specifies that the `allowedBridges` entry is a "shared bridge" whereas it is a "second bridge" per the relevant cross-chain concepts based on zkSync Era.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit, and clarified that Sophon L2 chain bridge operations will be executed where SOPH is the native asset.

As such, the Sophon team proceeded with removing the `payable` modifier from the **`BridgeHubWrapper::requestL2TransactionDirect`** function so as to avoid any ambiguities.

BHW-02M: Inexplicable Capability of Re-Initialization

Type	Severity	Location
Logical Fault	Minor	BridgeHubWrapper.sol:L74-L77

Description:

The `BridgeHubWrapper::initialize` function permits the contract to be re-initialized.

Impact:

The `BridgeHubWrapper` can be re-initialized an arbitrary number of times.

Example:

contracts/paymaster-on-l1/BridgeHubWrapper.sol

SOL

```sol
74  function initialize(uint256 maxFee_, address refundRecipient_) external onlyOwner
{
75      maxFee = maxFee_;
76      refundRecipient = refundRecipient_;
77  }
```

Recommendation:

We advise re-initialization to be prevented by utilizing a `bool` flag to indicate that the contract has already been initialized.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit and clarified that they intend to be able to re-initialize the contract multiple times, rendering this exhibit to be acknowledged.

GuardianDelegation Manual Review Findings

GDN-01M: Improper Immutable Domain Separator

Type	Severity	Location
Logical Fault	Minor	GuardianDelegation.sol:L83-L91, L458

Description:

Per the **EIP-712** standard, the domain separator should not be `immutable` as a blockchain fork will cause it to be incorrect on the forked chain due to the difference in the `block.chainid` value.

Impact:

Although a blockchain fork event has a low likelihood, we still advise our recommendation to be adhered to so as to avoid potential complications in the future.

Example:

contracts/tokens/delegation/GuardianDelegation.sol

SOL

```sol
73  /// @notice EIP-712 Domain Separator
74  bytes32 public immutable DOMAIN_SEPARATOR;
75
76
77  /**
78   * @notice Initializes the contract with the GuardianNFT contract address
79   * @param guardianNFT_ The address of the GuardianNFT contract
80   */
81  constructor(IERC721 guardianNFT_) {
82      guardianNFT = guardianNFT_;
```

Example (Cont.):

```sol
        DOMAIN_SEPARATOR = keccak256(
            abi.encode(
                DOMAIN_TYPEHASH,
                keccak256(bytes("GuardianDelegation")),
                keccak256(bytes("1")),
                block.chainid,
                address(this)
            )
        );
    }
```

Recommendation:

We advise an approach similar to OpenZeppelin's `EIP712.sol` implementation to be followed, re-calculating the `DOMAIN_SEPARATOR` solely when the `block.chainid` changes.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The `DOMAIN_SEPARATOR` has been relocated as a state declaration of the `GuardianDelegationState` contract and is configured by the newly introduced `GuardianDelegation::setDomainSeparator` function, permitting it to be re-calculated if needed thus alleviating this exhibit.

GuardianNFT Manual Review Findings

GNF-01M: Inexplicable Capability of Re-Initialization

Type	Severity	Location
Logical Fault	Minor	GuardianNFT.sol:L96-L98

Description:

The `GuardianNFT::initialize` function permits the contract to be re-initialized.

Impact:

The `GuardianNFT` can be re-initialized an arbitrary number of times.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
92  /**
93   * @notice Initializes the contract and grants whitelist manager role to a
manager
94   * @param manager The address to be assigned as the whitelist manager
95   */
96  function initialize(address manager) external onlyRole(DEFAULT_ADMIN_ROLE) {
97      _grantRole(WHITELIST_MANAGER_ROLE, manager);
98  }
```

Recommendation:

We advise re-initialization to be prevented by utilizing a `bool` flag to indicate that the contract has already been initialized.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit and clarified that they intend to be able to re-initialize the contract multiple times, rendering this exhibit to be acknowledged.

GNF-02M: Upgradeable Incompatibility of Dependency

Type	Severity	Location
Language Specific	Minor	GuardianNFT.sol:L81

Description:

The `GuardianNFT` implementation is meant to be an upgradeable NFT implementation, however, the `ERC721AQueryable` dependency it inherits from does not support upgradeability and multiple crucial items within its implementation will not be initialized properly.

Impact:

The `_name` and `_symbol` of the `GuardianNFT` token will be incorrect and will yield empty values. Although the public-facing getter functions have been properly overridden, it is still not advisable to utilize a non-upgradeable dependency in an upgradeable context.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
78  /**
79   * @notice Contract constructor, initializes the ERC721A token
80   */
81  constructor() ERC721A(name(), symbol()) {}
82
83  /**
84   * @notice Checks if the contract supports a specific interface
85   * @param interfaceId The interface identifier
86   * @return True if the contract supports the interface, false otherwise
87   */
```

Example (Cont.):

```sol
function supportsInterface(bytes4 interfaceId) public view override(UpgradeableAccessControl, ERC721A, IERC721A) returns (bool) {
    return super.supportsInterface(interfaceId);
}

/**
 * @notice Initializes the contract and grants whitelist manager role to a manager
 * @param manager The address to be assigned as the whitelist manager
 */
function initialize(address manager) external onlyRole(DEFAULT_ADMIN_ROLE) {
    _grantRole(WHITELIST_MANAGER_ROLE, manager);
}
```

Recommendation:

We advise the upgradeable variant of the `ERC721A` dependency to be utilized, ensuring that its data entries are correctly initialized during the `GuardianNFT::initialize` function.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit and clarified that they are aware of the upgradeability related pitfalls that their current pattern inherits and wish to acknowledge them.

GNF-03M: Incorrect Batch Transfer Implementation

Type	Severity	Location
Logical Fault	Medium	GuardianNFT.sol:L122-L126

Description:

The current `GuardianNFT::batchSafeTransferFrom` function will permit re-entrancies to occur in between transfers.

Impact:

A re-entrancy in between transfers should be prohibited as it can lead to unexpected interactions with the function's caller.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
116 /**
117  * @notice Safely transfers multiple tokens in a single transaction
118  * @param from The address to transfer from
119  * @param to The address to transfer to
120  * @param tokenIds The list of token IDs to transfer
121  */
122 function batchSafeTransferFrom(address from, address to, uint256[] memory
tokenIds) external {
123     for (uint256 i; i < tokenIds.length; i++) {
124         safeTransferFrom(from, to, tokenIds[i], '');
125     }
```

Example (Cont.):

```sol
126 }
```

Recommendation:

We advise the `ERC721A::_safeBatchTransferFrom` function to be utilized instead, ensuring that batch transfers signal the recipient at the end of their execution.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

Proper batch-based transfer function implementations have been introduced to the codebase, with the `GuardianNFT::safeBatchTransferFrom` function informing the recipient of the transfer once and after all NFTs have been transferred.

GNF-04M: Incorrect Decrease of Whitelist Count

Type	Severity	Location
Logical Fault	Medium	GuardianNFT.sol:L226

Description:

The referenced `if-revert` check introduces a race-condition whereby a whitelisted user can detect their whitelist decreasing, minting the amount of tokens necessary to cause the decrease operation to fail.

Impact:

It is possible to hijack all whitelist decrease operations by consuming the necessary whitelist amount to trigger the `if-revert` check.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
222 function decreaseWhitelist(address[] memory users, uint256[] memory
subtractedCounts) external onlyRole(WHITELIST_MANAGER_ROLE) {
223     if (users.length != subtractedCounts.length) revert CountMismatch();
224     for (uint256 i; i < users.length; i++) {
225         uint256 currentCount = whitelist[users[i]];
226         if (currentCount < subtractedCounts[i]) revert DecreaseTooHigh(users[i],
currentCount);
227         unchecked {
228             whitelist[users[i]] = currentCount - subtractedCounts[i];
229         }
230     }
231 }
```

Recommendation:

We advise the code to assign `currentCount` to the `subtractedCounts[i]` if the subtracted value exceeds the user's current value, ensuring that a decrease cannot be sabotaged.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

An `if-else` block has been introduced, setting the `whitelist` of the user to `0` if the subtraction count exceeds the user's current count thereby alleviating this exhibit in full.

Proxy2Step Manual Review Findings

PSP-01M: Improper Proxy Pattern

Type	Severity	Location
Logical Fault	Minor	Proxy2Step.sol:L26-L35, L41

Description:

The `Proxy2Step::receive` function will prevent the `Proxy2Step::fallback` function from being triggered, thereby causing any native transfers to the `Proxy2Step` contract to not be forwarded to the `receive` hook of a potential `implementation`.

Impact:

Although the `BridgeHubWrapper` implementation that makes use of the `Proxy2Step` implementation does not possess a `receive` hook, any future implementation that would define one would cause this vulnerability to manifest.

Example:

contracts/proxies/Proxy2Step.sol

SOL

```sol
21  /**
22   * @notice Fallback function that delegates all calls to the current implementation.
23   * @dev Forwards all calldata to the implementation address and returns the result.
24   * @dev Uses `delegatecall` to execute functions in the context of the implementation.
25   */
26  fallback() external virtual payable {
27      assembly {
28          calldatacopy(0, 0, calldatasize())
29          let result := delegatecall(gas(), sload(implementation.slot), 0, calldatasize(), 0, 0)
30          returndatacopy(0, 0, returndatasize())
```

Example (Cont.):

```sol
        switch result
        case 0 { revert(0, returndatasize()) }
        default { return(0, returndatasize()) }
    }
}

/**
 * @notice Receives Ether sent to the contract.
 * @dev This function is used to handle direct ETH transfers without data.
 */
receive() external virtual payable {}
```

Recommendation:

We advise the code to remove the `Proxy2Step::receive` function, ensuring that native funds with no data are handled by the `Proxy2Step::fallback` function and are properly forwarded to the intended `implementation`.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team opted to instead manually forward the call to the `implementation` within the `Proxy2Step::receive` function override, addressing this exhibit as a result.

ProxyAccessControl Manual Review Findings

PAC-01M: Improper Proxy Pattern

Type	Severity	Location
Logical Fault	Minor	ProxyAccessControl.sol:L27-L36, L42

Description:

The `ProxyAccessControl::receive` function will prevent the `ProxyAccessControl::fallback` function from being triggered, thereby causing any native transfers to the `ProxyAccessControl` contract to not be forwarded to the `receive` hook of a potential `implementation`.

Impact:

Although the `GuardianNFT` and `GuardianDelegation` implementations that make use of the `ProxyAccessControl` implementation do not possess `receive` hooks, any future implementation that would define one would cause this vulnerability to manifest.

Example:

contracts/proxies/ProxyAccessControl.sol

SOL

```sol
23  /**
24   * @notice Fallback function that delegates all calls to the current implementation.
25   * @dev Uses `delegatecall` to execute functions in the context of the implementation.
26   */
27  fallback() external virtual payable {
28      assembly {
29          calldatacopy(0, 0, calldatasize())
30          let result := delegatecall(gas(), sload(implementation.slot), 0, calldatasize(), 0, 0)
31          returndatacopy(0, 0, returndatasize())
32          switch result
```

Example (Cont.):

```sol
        case 0 { revert(0, returndatasize()) }
        default { return(0, returndatasize()) }
    }
}

/**
 * @notice Receives Ether sent to the contract.
 * @dev Used to handle direct ETH transfers without data.
 */
receive() external virtual payable {}
```

Recommendation:

We advise the code to remove the `ProxyAccessControl::receive` function, ensuring that native funds with no data are handled by the `ProxyAccessControl::fallback` function and are properly forwarded to the intended `implementation`.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team opted to instead manually forward the call to the `implementation` within the `ProxyAccessControl::receive` function override, addressing this exhibit as a result.

Upgradeable2Step Manual Review Findings

USP-01M: Invalid Upgradeability Support

Type	Severity	Location
Language Specific	Unknown	Upgradeable2Step.sol:L11, L33, L36

Description:

The `Upgradeable2Step` contract is meant to introduce upgradeability support to the `Ownable2Step` contract, however, it does not do so properly as index-based storage slots are being utilized.

Impact:

The impact cannot be precisely quantified as a storage slot overlap would have to occur for the vulnerability to manifest.

In case of an overlap, the severity would be significantly high as data corruption would occur.

Example:

contracts/proxies/Upgradeable2Step.sol

SOL

```sol
11  contract Upgradeable2Step is Ownable2Step {
12
13      /**
14       * @dev Emitted when a new implementation is proposed.
15       * @param previousImplementation The address of the previous implementation.
16       * @param newImplementation The address of the new implementation proposed.
17       */
18      event ReplaceImplementationStarted(address indexed previousImplementation,
address indexed newImplementation);
19
20      /**
```

Example (Cont.):

```sol
21        * @dev Emitted when a new implementation is accepted and becomes active.
22        * @param previousImplementation The address of the previous implementation.
23        * @param newImplementation The address of the new active implementation.
24        */
25       event ReplaceImplementation(address indexed previousImplementation, address indexed newImplementation);
26
27       /**
28        * @dev Thrown when an unauthorized account attempts to execute a restricted function.
29        */
30       error Unauthorized();
31
32       /// @notice The address of the implementation that is pending acceptance.
33       address public pendingImplementation;
34
35       /// @notice The address of the current implementation contract.
36       address public implementation;
```

Recommendation:

We advise the upgradeable variant of the `Ownable2Step` implementation to be utilized and **EIP-7201** storage slots to be introduced to the `Upgradeable2Step` implementation and any downstream dependencies.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team has stated that they have utilized this implementation safely multiple times and that they wish to acknowledge the storage-related potential corruption.

USP-02M: Inexplicable Acceptance Mechanism

Type	Severity	Location
Input Sanitization	Minor	Upgradeable2Step.sol:L73-L78

Description:

The `Upgradeable2Step::becomeImplementation` function permits any caller to supply a `proxy` they own and accept it as an implementation which we consider incorrect.

Impact:

It is presently possible to supply a caller-controlled `Upgradeable2Step` implementation to the `Upgradeable2Step::becomeImplementation` function which we consider invalid.

Example:

contracts/proxies/Upgradeable2Step.sol

SOL

```sol
68  /**
69   * @notice Allows a new implementation to become the active implementation in a proxy contract.
70   * @dev Can only be called by the owner of the specified proxy contract. Calls `acceptImplementation` on the proxy contract.
71   * @param proxy The proxy contract where the new implementation should be accepted.
72   */
73  function becomeImplementation(Upgradeable2Step proxy) public {
74      if (msg.sender != proxy.owner()) {
75          revert Unauthorized();
76      }
77      proxy.acceptImplementation();
```

Example (Cont.):

```sol
}
```

Recommendation:

We advise some form of stricter access control to be introduced, disallowing arbitrary callers to the function and preventing the `Upgradeable2Step` contract from invoking the **`Upgradeable2Step::acceptImplementation`** function of arbitrary contracts.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit and stated that they do not envision any significant impact arising from the current behaviour, opting to acknowledge this exhibit.

UpgradeableAccessControl Manual Review Findings

UAC-01M: Invalid Upgradeability Support

Type	Severity	Location
Language Specific	Unknown	UpgradeableAccessControl.sol:L11, L13, L19, L30

Description:

The `UpgradeableAccessControl` contract is meant to introduce upgradeability support to the `AccessControlDefaultAdminRules` contract, however, it does not do so properly as index-based storage slots are being utilized.

Impact:

The impact cannot be precisely quantified as a storage slot overlap would have to occur for the vulnerability to manifest.

In case of an overlap, the severity would be significantly high as data corruption would occur.

Example:

contracts/proxies/UpgradeableAccessControl.sol

SOL

```sol
contract UpgradeableAccessControl is AccessControlDefaultAdminRules {
    /// @notice The address of the current implementation contract.
    address public implementation;

    /**
     * @notice Constructs the UpgradeableAccessControl contract.
     * @dev Initializes the AccessControlDefaultAdminRules with a delay of 3 days and sets the deployer as the initial default admin.
     */
    constructor() AccessControlDefaultAdminRules(3 days, msg.sender) {}
```

Recommendation:

We advise the `AccessControlDefaultAdminRules` implementation to be properly made upgradeable by introducing storage slots based on the **EIP-7201** standard, and similar slots to be introduced to the `UpgradeableAccessControl` implementation.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team has stated that they have utilized this implementation safely multiple times and that they wish to acknowledge the storage-related potential corruption.

GuardianDelegation Code Style Findings

GDN-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	GuardianDelegation.sol:L232, L241

Description:

The linked mathematical operations are guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

contracts/tokens/delegation/GuardianDelegation.sol

SOL

```sol
229 if (_senderTotal >= nftBalance)
230     return (0, 0);
231
232 totalUndelegated = nftBalance - _senderTotal;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statements to be wrapped in `unchecked` code blocks thereby optimizing their execution cost.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

An `unchecked` code block has been introduced for both referenced arithmetic operations, addressing this exhibit as advised.

GDN-02C: Inefficient `mapping` Lookups

Type	Severity	Location
Gas Optimization	Informational	GuardianDelegation.sol:L228, L235, L263, L264

Description:

The linked statements perform key-based lookup operations on `mapping` declarations from storage multiple times for the same key redundantly.

Example:

contracts/tokens/delegation/GuardianDelegation.sol

SOL

```sol
228 uint256 _senderTotal = _countSent[sender][delegationType];
229 if (_senderTotal >= nftBalance)
230     return (0, 0);
231
232 totalUndelegated = nftBalance - _senderTotal;
233 newDelegations = totalUndelegated;
234
235 uint256 _receiverTotal = _countReceived[receiver][delegationType];
236 uint256 _maxReceivedAllowed = maxReceivedAllowed[delegationType];
237 if (_maxReceivedAllowed != 0) {
```

Example (Cont.):

```sol
    if (_receiverTotal >= _maxReceivedAllowed) {
        return (0, totalUndelegated);
    }
    _maxReceivedAllowed -= _receiverTotal;

    if (newDelegations > _maxReceivedAllowed) {
        newDelegations = _maxReceivedAllowed;
    }
}

// Impose user-defined limit for this receiver
if (amount != 0 && newDelegations > amount) {
    newDelegations = amount;
}

Delegation memory thisDelegation = _delegationsSent[sender][receiver]
[delegationType];
if (thisDelegation.lastUpdate != 0) {
    thisDelegation.previousScore += (block.timestamp - thisDelegation.lastUpdate)
* thisDelegation.amount;
} else {
    _setsSent[sender][delegationType].add(receiver);
    _setsReceived[receiver][delegationType].add(sender);
}

thisDelegation.amount += newDelegations;

_countSent[sender][delegationType] = _senderTotal + newDelegations;
_countReceived[receiver][delegationType] = _receiverTotal + newDelegations;
```

Recommendation:

As the lookups internally perform an expensive `keccak256` operation, we advise the lookups to be cached wherever possible to a single local declaration that either holds the value of the `mapping` in case of primitive types or holds a `storage` pointer to the `struct` contained.

As the compiler's optimizations may take care of these caching operations automatically at-times, we advise the optimization to be selectively applied, tested, and then fully adopted to ensure that the proposed caching model indeed leads to a reduction in gas costs.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit but opted to acknowledge it in the current iteration of the codebase.

GuardianDelegationState Code Style Findings

GDS-01C: Inefficient Data Structures

Type	Severity	Location
Gas Optimization	Informational	GuardianDelegationState.sol:L35, L38, L41, L44, L47, L50

Description:

The overall `GuardianDelegation` implementation is significantly inefficient as it will utilize multiple different complex data structures which can be greatly simplified as well as optimized.

Example:

contracts/tokens/delegation/GuardianDelegationState.sol

SOL

```sol
34  /// @notice Mapping of sender -> receiver -> delegationType -> Delegation data (amount, lastUpdate, previousScore)
35  mapping (address => mapping (address => mapping (DelegationType => Delegation))) public _delegationsSent;
36
37  /// @notice Internal set to track senders and the corresponding delegation types
38  mapping (address => mapping (DelegationType => EnumerableSet.AddressSet)) internal _setsSent;
39
40  /// @notice Internal set to track receivers and the corresponding delegation types
41  mapping (address => mapping (DelegationType => EnumerableSet.AddressSet)) internal _setsReceived;
42
43  /// @notice Internal mapping to track the number of tokens sent as delegation by a sender for each delegation type
```

Example (Cont.):

```sol
44  mapping (address => mapping (DelegationType => uint256)) internal _countSent;
45
46  /// @notice Internal mapping to track the number of tokens received as delegation by a receiver for each delegation type
47  mapping (address => mapping (DelegationType => uint256)) internal _countReceived;
48
49  /// @notice Mapping that defines the maximum delegation a receiver can accept for a given delegation type. If set to 0, there is no maximum.
50  mapping (DelegationType => uint256) public maxReceivedAllowed;
```

Recommendation:

We advise the `DelegationType` to become the primary key in all `mapping` entries, ensuring that their lookups can be cached in memory.

Additionally, we advise a common `mapping` to be utilized for the same lookup paths (i.e. `_setsSent`, `_setsReceived`, `_countSent`, `_countReceived`) by defining a `struct` that contains all relevant entries.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The Sophon team evaluated this exhibit but opted to acknowledge it in the current iteration of the codebase.

GuardianNFT Code Style Findings

GNF-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	GuardianNFT.sol:L139, L163, L279

Description:

The linked mathematical operations are guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

contracts/tokens/GuardianNFT.sol

SOL

```sol
138 if (allowedMints < quantity) revert QuantityTooHigh(allowedMints);
139 whitelist[msg.sender] = allowedMints - quantity;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statements to be wrapped in `unchecked` code blocks thereby optimizing their execution cost.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

An `unchecked` code block has been introduced for all referenced arithmetic operations, addressing this exhibit as advised.

GuardianNFTState Code Style Findings

GNS-01C: Redundant Inheritance Chain

Type	Severity	Location
Code Style	Informational	GuardianNFTState.sol:L12

Description:

The `GuardianNFTState` contract will simultaneously inherit from the `ERC721A` and `ERC721AQueryable` implementations which is redundant as the `ERC721AQueryable` contract already inherits from the `ERC721A` contract.

Example:

contracts/tokens/GuardianNFTState.sol

SOL

```sol
12  abstract contract GuardianNFTState is ERC721A, ERC721AQueryable {
```

Recommendation:

We advise the `ERC721A` inheritance to be omitted, optimizing the code's legibility.

Alleviation (1edaebf4ac1b56c5ef32a38ada079bbbaf0b95b4):

The redundant `ERC721A` inheritance has been omitted as advised, optimizing the code's legibility.

Finding Types

A description of each finding type included in the report can be found below and is linked by each respective finding. A full list of finding types Omniscia has defined will be viewable at the central audit methodology we will publish soon.

Input Sanitization

As there are no inherent guarantees to the inputs a function accepts, a set of guards should always be in place to sanitize the values passed in to a particular function.

Indeterminate Code

These types of issues arise when a linked code segment may not behave as expected, either due to mistyped code, convoluted if blocks, overlapping functions / variable names and other ambiguous statements.

Language Specific

Language specific issues arise from certain peculiarities that the Circom language boasts that discerns it from other conventional programming languages.

Curve Specific

Circom defaults to using the BN128 scalar field (a 254-bit prime field), but it also supports BSL12-381 (which has a 255-bit scalar field) and Goldilocks (with a 64-bit scalar field). However, since there are no constants denoting either the prime or the prime size in bits available in the Circom language, some Circomlib templates like `Sign` (which returns the sign of the input signal), and `AliasCheck` (used by the strict versions of `Num2Bits` and `Bits2Num`), hardcode either the BN128 prime size or some other constant related to BN128. Using these circuits with a custom prime may thus lead to unexpected results and should be avoided.

Code Style

In these types of findings, we identify whether a project conforms to a particular naming convention and whether that convention is consistent within the codebase and legible. In case of inconsistencies, we point them out under this category. Additionally, variable shadowing falls under this category as well which is identified when a local-level variable contains the same name as a toplevel variable in the circuit.

Mathematical Operations

This category is used when a mathematical issue is identified. This implies an issue with the implementation of a calculation compared to the specifications.

Logical Fault

This category is a bit broad and is meant to cover implementations that contain flaws in the way they are implemented, either due to unimplemented functionality, unaccounted-for edge cases or similar extraordinary scenarios.

Privacy Concern

This category is used when information that is meant to be kept private is made public in some way.

Proof Concern

Under-constrained signals are one of the most common issues in zero-knowledge circuits. Issues with proof generation fall under this category.

Severity Definition

In the ever-evolving world of blockchain technology, vulnerabilities continue to take on new forms and arise as more innovative projects manifest, new blockchain-level features are introduced, and novel layer-2 solutions are launched. When performing security reviews, we are tasked with classifying the various types of vulnerabilities we identify into subcategories to better aid our readers in understanding their impact.

Within this page, we will clarify what each severity level stands for and our approach in categorizing the findings we pinpoint in our audits. To note, all severity assessments are performed **as if the contract's logic cannot be upgraded** regardless of the underlying implementation.

Severity Levels

There are five distinct severity levels within our reports; `unknown`, `informational`, `minor`, `medium`, and `major`. A TL;DR overview table can be found below as well as a dedicated chapter to each severity level:

	Impact (None)	Impact (Low)	Impact (Moderate)	Impact (High)
Likelihood (None)	Informational	Informational	Informational	Informational
Likelihood (Low)	Informational	Minor	Minor	Medium
Likelihood (Moderate)	Informational	Minor	Medium	Major
Likelihood (High)	Informational	Medium	Major	Major

Unknown Severity

The `unknown` severity level is reserved for misbehaviors we observe in the codebase that cannot be quantified using the above metrics. Examples of such vulnerabilities include potentially desirable system behavior that is undocumented, reliance on external dependencies that are out-of-scope but could result in some form of vulnerability arising, use of external out-of-scope contracts that appears incorrect but cannot be pinpointed, and other such vulnerabilities.

In general, `unknown` severity level vulnerabilities require follow-up information by the project being audited and are either adjusted in severity (if valid), or marked as nullified (if invalid).

Additionally, the `unknown` severity level is sometimes assigned to centralization issues that cannot be assessed in likelihood due to their exploitation being tied to the honesty of the project's team.

Informational Severity

The `informational` severity level is dedicated to findings that do not affect the code functionally and tend to be stylistic or optimizational in nature. Certain edge cases are also set under `informational` vulnerabilities, such as overflow operations that will not manifest in the lifetime of the contract but should be guarded against as a best practice, to give an example.

Minor Severity

The `minor` severity level is meant for vulnerabilities that require functional changes in the code but tend to either have little impact or be unlikely to be recreated in a production environment. These findings can be acknowledged except for findings with a moderate impact but low likelihood which must be alleviated.

Medium Severity

The `medium` severity level is assigned to vulnerabilities that must be alleviated and have an observable impact on the overall project. These findings can only be acknowdged if the project deems them desirable behavior and we disagree with their point-of-view, instead urging them to reconsider their stance while marking the exhibit as acknowledged given that the project has ultimate say as to what vulnerabilities they end up patching in their system.

Major Severity

The `major` severity level is the maximum that can be specified for a finding and indicates a significant flaw in the code that must be alleviated.

Likelihood & Impact Assessment

As the preface chapter specifies, the blockchain space is constantly reinventing itself meaning that new vulnerabilities take place and our understanding of what security means differs year-to-year.

In order to reliably assess the likelihood and impact of a particular vulnerability, we instead apply an abstract measurement of a vulnerability's impact, duration the impact is applied for, and probability that the vulnerability would be exploited in a production environment.

Our proposed definitions are inspired by multiple sources in the security community and are as follows:

Disclaimer

The following disclaimer applies to all versions of the audit report produced (preliminary / public / private) and is in effect for all past, current, and future audit reports that are produced and hosted under Omniscia:

IMPORTANT TERMS & CONDITIONS REGARDING OUR SECURITY AUDITS/REVIEWS/REPORTS AND ALL PUBLIC/PRIVATE CONTENT/DELIVERABLES

Omniscia ("Omniscia") has conducted an independent security review to verify the integrity of and highlight any vulnerabilities, bugs or errors, intentional or unintentional, that may be present in the codebase that were provided for the scope of this Engagement.

Blockchain technology and the cryptographic assets it supports are nascent technologies. This makes them extremely volatile assets. Any assessment report obtained on such volatile and nascent assets may include unpredictable results which may lead to positive or negative outcomes.

In some cases, services provided may be reliant on a variety of third parties. This security review does not constitute endorsement, agreement or acceptance for the Project and technology that was reviewed. Users relying on this security review should not consider this as having any merit for financial advice or technological due diligence in any shape, form or nature.

The veracity and accuracy of the findings presented in this report relate solely to the proficiency, competence, aptitude and discretion of our auditors. Omniscia and its employees make no guarantees, nor assurance that the contracts are free of exploits, bugs, vulnerabilities, deprecation of technologies or any system / economical / mathematical malfunction.

This audit report shall not be printed, saved, disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Omniscia.

All the information/opinions/suggestions provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

Information in this report is provided 'as is'. Omniscia is under no covenant to the completeness, accuracy or solidity of the contracts reviewed. Omniscia's goal is to help reduce the attack vectors/surface and the high level of variance associated with utilizing new and consistently changing technologies.

Omniscia in no way claims any guarantee, warranty or assurance of security or functionality of the technology that was in scope for this security review.

In no event will Omniscia, its partners, employees, agents or any parties related to the design/creation of this security review be ever liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this security review.

Cryptocurrencies and all other technologies directly or indirectly related to cryptocurrencies are not standardized, highly prone to malfunction and extremely speculative by nature. No due diligence and/or safeguards may be insufficient and users should exercise maximum caution when participating and/or investing in this nascent industry.

The preparation of this security review has made all reasonable attempts to provide clear and actionable recommendations to the Project team (the “client”) with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts in scope for this engagement.

It is the sole responsibility of the Project team to provide adequate levels of test and perform the necessary checks to ensure that the contracts are functioning as intended, and more specifically to ensure that the functions contained within the contracts in scope have the desired intended effects, functionalities and outcomes, as documented by the Project team.

All services, the security reports, discussions, work product, attack vectors description or any other materials, products or results of this security review engagement is provided "as is" and "as available" and with all faults, uncertainty and defects without warranty or guarantee of any kind.

Omniscia will assume no liability or responsibility for delays, errors, mistakes, or any inaccuracies of content, suggestions, materials or for any loss, delay, damage of any kind which arose as a result of this engagement/security review.

Omniscia will assume no liability or responsibility for any personal injury, property damage, of any kind whatsoever that resulted in this engagement and the customer having access to or use of the products, engineers, services, security report, or any other other materials.

For avoidance of doubt, this report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or any other type of advice.