A&D FORENSICS

REDSWAN®

# SMART CONTRACT AUDIT

## TOKEN STUDIO REPORT

### DESCRIPTION

This is a report for the Swan project, themed 'Security Token Studio', identifying issues and vulnerabilities in the smart contract source code.

# TABLE OF CONTENTS

# Introduction

RedSwan contracted A&D Forensics to conduct an audit on Red Swan Security Token Studio Project. We detailed our methodology in this report to evaluate potential security issues in the smart contract and stated our observations in this report. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

# Abstract

This report has been prepared for a smart contract project themed Token Studio with the aim to discover issues and vulnerabilities in the source code. A comprehensive examination has been performed, utilizing tooling analysis and manual review technique by smart contract security expert.

**The auditing process pays special attention to the following considerations:**

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the general specifications and intentions of the standard design patterns
- Cross referencing contract structure and implementation against similar contract produced by industry leaders.
- Through line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from high to low. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspectives:

- Enhancing general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public; Provide more transparency on privileged activities once the protocol is live.

# OVERVIEW

## Project Summary

| | |
|---|---|
| **Project Name** | Token Studio |
| **Platform** | Ethereum |
| **Code base** | https://github.com/hashgraph/asset-tokenization-studio |
| **Commit** | 03bf945620ca4a9e0cea2491424f3a39e0b213fe |

### Audit Summary

| | |
|---|---|
| **Delivery date** | 31/5/2024 |
| **Audit Methodology** | Manual Review and Static test |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 |
| High | 1 | 0 | 1 | 0 | 1 |
| Medium | 17 | 0 | 4 | 0 | 0 |
| Low | 9 | 0 | 2 | 0 | 0 |
| Informational | 21 | 0 | 0 | 0 | 0 |
| Discussion | 0 | 0 | 0 | 0 | 0 |

# FINDINGS

## Project Description

A solidity based smart contract solution themed security token studio that focuses on deploying new and independent security token in the form of bonds or equities.

## Manual Test Findings (FN)

### FN1

**Title:** ISIN check can easily be bypassed

**Severity:** Medium

**Description:** The smart contract features a checkISIN modifier aimed at validating International Securities Identification Numbers (ISINs), essential for securities trading. However, its implementation solely verifies ISIN length, disregarding format or uniqueness checks.

Code - contracts/factory/Factory.sol

```
modifier checkISIN(string calldata isin) { @> if (bytes(isin).lengt
h != _ISIN_LENGTH) { revert WrongISIN(isin); } _; }
```

**Details:** The checkISIN modifier inadequately verifies ISIN codes, only examining their length. This oversight allows incorrect or duplicate ISINs to pass validation, potentially leading to erroneous transactions.

**Impact:** Incorrect ISIN can be used to deploy both equity and bonds security.

**Tools Used:** Manual Review

**Recommendations:** Enhance the checkISIN modifier to perform comprehensive ISIN validation, including format and uniqueness checks. Chain-link functions can be used to verify the ISIN off chain.

## FN2

**Title:** Centralization Risk : A user with Issuer Role can issue unlimited number of token

**Severity:** High

**Description:** The smart contract system allows a user with the Issuer role to issue an unlimited number of tokens when isWhitelist is set to false. This presents a centralization risk, as it enables the issuer to exercise unchecked control over the token issuance process

**POC:** The code below shows that there is no limit to the amount of tokens the user with Issuer role can give out

Code - contracts/layer_1/ERC1400/ERC1594/ERC1594.sol

```
function issue( address _tokenHolder, uint256 _value, bytes calldat
a _data ) external virtual override onlyUnpaused onlyRole(_ISSUER_R
OLE) checkControlList(_tokenHolder) onlyWithoutMultiPartition onlyI
ssuable { _issue(_tokenHolder, _value, _data); }
```

Code - contracts/layer_1/ERC1400/ERC1594/ERC1594StorageWrapper.sol

```
function _issue( address _tokenHolder, uint256 _value, bytes callda
ta _data ) internal virtual { // Add a function to validate the `_d
ata` parameter _mint(_tokenHolder, _value); emit Issued(_msgSender
(), _tokenHolder, _value, _data); }
```

**Details:** The checkISIN modifier inadequately verifies ISIN codes, only examining their length. This oversight allows incorrect or duplicate ISINs to pass validation, potentially leading to erroneous transactions.

**Impact:** The unrestricted ability to issue tokens poses significant risks to the integrity and stability of the token economy. It undermines decentralization principles and may erode

user trust if exploited maliciously or negligently. Moreover, excessive token issuance can dilute the value of existing tokens and disrupt market dynamics.

**Tools Used:** Manual Review

**Recommendations**: Implement strict controls and limitations on token issuance for users with the Issuer role. Consider implementing a cap on the maximum number of tokens that can be issued within a specified timeframe, along with additional authorization checks to prevent abuse.

**Update**: FN2 Resolved

## FN3

**Title:** No zero address check when issuing tokens if isWhitelist is set to false.

**Severity:** Low

**Description:** The smart contract system lacks a zero-address check when issuing tokens if the isWhitelist flag is set to false. This absence of validation exposes the contract to potential vulnerabilities, as it allows tokens to be minted and transferred without proper verification of recipient addresses.

**POC:** The code below is only effective when isWhitelist is set to true, when it is set to false, _checkControlList always returns true for every account including zero address.

Code - contracts/layer_1/controlList/ControlListStorageWrapper.sol

```
modifier checkControlList(address account) { if (!_checkControlList
(account)) { revert AccountIsBlocked(account); } _; }
```

Code - contracts/layer_1/controlList/ControlListStorageWrapper.sol

```
function _checkControlList(address account) internal view virtual r
eturns (bool) { return _getControlListType() == _isInControlList(ac
count); }
```

**Details:** The vulnerability stems from the failure to verify recipient addresses against the zero address when the isWhitelist flag is disabled. Without this check, tokens can be minted and transferred to invalid or non-existent addresses, leading to loss of assets or unintended consequences.

**Impact:** The absence of a zero address check poses risks to the security and integrity of the token issuance process.

**Tools Used:** Manual Review

**Recommendations**: Implement a zero address check to validate recipient addresses before minting or transferring tokens, especially when the isWhitelist flag is set to false. This check will help mitigate the risk of tokens being issued to invalid or non-existent addresses.

## FN4

**Title:** ERC1643::setDocument lacks check for if name already exists.

**Severity:** Medium

**Description:** The setDocument function in the ERC1643 lacks a check to verify if a document name already exists before adding a new document. This oversight introduces a potential vulnerability where creating new documents will update previously created documents.

**POC:** In the test below, i used the setDocument function twice with the same name and it updated the document instead of throwing an error that the document name already exists.

**Code**: contracts/contracts/layer_1/ERC1400/ERC1643/ERC1643.sol

```
it('Documents Creation Test', async () => { // ADD TO LIST --------
---------------------------------------------------------- // Grant
ing Role to account C accessControlFacet = accessControlFacet.conne
ct(signer_A) await accessControlFacet.grantRole(_DOCUMENTER_ROLE, a
ccount_C) // Using account C (with role) erc1643Facet = erc1643Face
t.connect(signer_C) // check that Document not in the list let docu
ments = await erc1643Facet.getAllDocuments() expect(documents.lengt
h).to.equal(0) // add document await expect( erc1643Facet.setDocume
nt( documentName_1, documentURI_1, documentHASH_1 ) ) .to.emit(erc1
643Facet, 'DocumentUpdated') .withArgs(documentName_1, documentURI_
1, documentHASH_1) // Create a new document with documentName_1 but
another document hash await erc1643Facet.setDocument( documentName_
1, documentURI_3, documentHASH_2 ) // check documents documents = a
wait erc1643Facet.getAllDocuments() expect(documents.length).to.equ
al(2) expect(documents[0]).to.equal(documentName_1) const document
= await erc1643Facet.getDocument(documentName_1) expect(document
[0]).to.equal(documentURI_1) // creation of new document updated th
e document hash expect(document[1]).to.equal(documentHASH_2) })
```

**Details:** The vulnerability arises due to the absence of a name existence check within the setDocument function. Without this validation, the function allows documents to be added without ensuring uniqueness of document names. As a result, previously created documents can be updated.

**Impact:** The lack of a name existence check undermines the reliability and usability of the document management system. it may lead to unintended consequences such as overwriting existing documents or difficulty in identifying and accessing specific documents.

**Tools Used:** Manual Review

**Recommendations**: Implement a name existence check within the setDocument function to ensure that duplicate document names are not added to the system. This check should verify if a document with the same name already exists before allowing the addition of a new document. By enforcing uniqueness of document names, the system can maintain clarity and prevent potential issues associated with duplicate documents.

## FN5

**Title:** NatSpec standard was not followed throughout the code

**Severity:** Low

**Description:** The codebase does not fully adhere to the Natspec standard, which provides guidelines for writing documentation comments within Solidity contracts. Natspec comments serve as inline documentation to improve code readability and understandability by providing descriptions of contract functions, parameters, and return values.

**Details:** The absence of Natspec comments in the codebase indicates a deviation from best practices in Solidity development. Natspec comments play a crucial role in documenting the purpose, behavior, and usage of contract functions, ensuring that developers and users can easily understand the contract's functionality without the need to inspect the source code extensively. Without proper documentation, the code becomes less accessible and more prone to misunderstandings, errors, and maintenance challenges.

**Impact:** The lack of Natspec-standard documentation diminishes the readability and usability of the codebase, particularly for developers who need to interact with or integrate the contract into other systems. It increases the likelihood of misinterpretation or misuse of contract functions, leading to potential bugs, vulnerabilities, or unintended behavior. Moreover, the absence of clear documentation hinders the onboarding of new developers and users, slowing down the adoption and development process.

**Tools Used:** Manual Review

**Recommendations**: Integrate Natspec-standard documentation comments throughout the codebase to provide clear and comprehensive explanations of contract functions, parameters, and return values. Follow the Natspec guidelines to ensure consistency and readability in the documentation. By documenting the contract's behavior and usage effectively, developers and users can better understand and utilize the contract, improving

overall usability, reliability, and security. Additionally, consider incorporating automated tools or linters to enforce Natspec compliance and maintain consistent documentation standards across the codebase.

## FN6

**Title:** ERC1594::redeem and ERC1594::redeemFrom does not work as intended, it just burns tokens and data parameter has no use.

**Severity:** Medium

**Description:** The ERC1594::redeem and ERC1594::redeemFrom functions in the contract do not operate as intended. Both functions are designed to facilitate the redemption of tokens, but they only perform token burning without utilizing the accompanying data. As a result, the redemption process lacks functionality and fails to fulfill its intended purpose effectively.

**POC:** The functions below do not have a complete implementation of the intended functionality, and currently only burns tokens.

Code - contracts/layer_1/ERC1400/ERC1594/ERC1594.sol

```
function redeem( uint256 _value, bytes calldata _data ) external vi
rtual override onlyUnpaused checkControlList(_msgSender()) onlyWith
outMultiPartition { _redeem(_value, _data); }
```

Code - contracts/layer_1/ERC1400/ERC1594/ERC1594StorageWrapper.sol

```
function _redeem(uint256 _value, bytes calldata _data) internal vir
tual { // Add a function to validate the `_data` parameter _burn(_m
sgSender(), _value); emit Redeemed(address(0), _msgSender(), _valu
e, _data); }
```

**Details:** The discrepancy between the expected behavior and the actual implementation of ERC1594::redeem and ERC1594::redeemFrom functions poses a critical vulnerability. These functions are crucial for enabling token holders to redeem their tokens for certain benefits or privileges. However, their current implementation overlooks the utilization of associated data, rendering the redemption process incomplete and potentially unusable for token holders.

**Impact:** The ERC1594::redeem and ERC1594::redeemFrom functions currently function as token burning functions.

**Tools Used:** Manual Review

**Recommendations**: To rectify the issue, the contract's redeem and redeemFrom functions should be updated to incorporate the appropriate utilization of associated data. Ensure that redeeming tokens triggers the intended actions or benefits specified in the contract, such as accessing services, unlocking features, or receiving rewards.

## FN7

**Title:** ERC1594::transferWithData does not use the data parameter

**Severity:** Medium

**Description** The ERC1594::transferWithData function in the contract does not utilize the data parameter provided as part of the function signature. Although the function is designed to facilitate token transfers with accompanying data, the current implementation fails to incorporate this data into the transfer process. As a result, the data parameter remains unused, potentially leading to missed opportunities for enhancing token functionality or interoperability with other smart contracts

**POC:**

Code - contracts/layer_1/ERC1400/ERC1594/ERC1594.sol

```
function transferWithData( address _to, uint256 _value, bytes calld
ata _data // solhint-disable-line no-unused-vars ) external virtual
override onlyUnpaused checkControlList(_msgSender()) checkControlLi
st(_to) onlyWithoutMultiPartition { // Add a function to validate t
he `_data` parameter _transfer(_msgSender(), _to, _value); }
```

**Details:** The absence of data utilization in ERC1594::transferWithData represents a critical vulnerability as it undermines the intended functionality of the token transfer mechanism. Data parameters are essential for conveying additional information or instructions along with token transfers, enabling more versatile and context-aware interactions between smart contracts. However, the oversight in utilizing the data parameter limits the contract's interoperability and may restrict its integration with other decentralized applications (DApps) or protocols that rely on data exchange during token transfers.

**Impact:** The failure to utilize the data parameter in ERC1594::transferWithData compromises the contract's ability to support complex or conditional token transfers that require accompanying data payloads.

**Tools Used:** Manual Review

**Recommendations**: To address the issue, the contract's ERC1594::transferWithData function should be updated to properly handle the data parameter and incorporate it into the token transfer process. Ensure that the function can effectively process and interpret the provided data.

## FN8

**Title:** Cap::setMaxSupply can be set to unlimited

**Severity:** Medium

**Description**: The setMaxSupply function in the Cap contract allows for the specification of the maximum cap for the token supply. However, if the cap is set to zero, it results in an unlimited maximum cap.

**POC:** There is no zero check to prevent setting the max supply as zero and the modifier for checkMaxSupply returns true if max supply is zero.

Code - contracts/layer_1/cap/Cap.sol

```
function setMaxSupply(
    uint256 _maxSupply
)
    external
    virtual
    override
    onlyUnpaused
    onlyRole(_CAP_ROLE)
    checkNewMaxSupply(_maxSupply)
    returns (bool success_)
{
    _setMaxSupply(_maxSupply);
    success_ = true;
}
```

Code - contracts/layer_1/cap/CapStorageWrapper.sol

```
function _setMaxSupply(uint256 _maxSupply) internal virtual {
    uint256 previousMaxSupply = _capStorage().maxSupply;
    _capStorage().maxSupply = _maxSupply;
    emit MaxSupplySet(_msgSender(), _maxSupply, previousMaxSupply);
}
```

```
// modifiers
modifier checkMaxSupply(uint256 _amount) {
    uint256 newTotalSupply = _totalSupply() + _amount;
    if (!_checkMaxSupply(newTotalSupply)) {
        revert MaxSupplyReached(_capStorage().maxSupply);
    }
    _;
}
```

Code - contracts/layer_1/cap/CapStorageWrapper.sol

```
 function _checkMaxSupplyCommon(uint256 _amount, uint256 _maxSupply) private pur
e returns (bool) {
        if (_maxSupply == 0) return true;
        if (_amount <= _maxSupply) return true;
        return false;
    }
```

**Details:** The vulnerability arises from the fact that setting the maximum cap to zero effectively renders the cap unlimited, allowing for the potential inflation of the token supply beyond any predefined limits.

**Impact:** This vulnerability poses a significant risk as it can lead to uncontrolled token supply inflation, potentially resulting in adverse consequences such as devaluation of the token and loss of investor confidence.

**Tools Used:** Manual Review

**Recommendations**: Max supply should not be zero to prevent inflation.

## FN9

**Title:** Duplicate corporate actions can be added;

**Severity:** Medium

**Description:** The CorporateActionsSecurity contract allows for the addition of corporate actions, but it lacks proper validation to prevent the addition of duplicate corporate actions. As a result, multiple identical corporate actions can be added to the contract.

**POC:** Code - contracts/layer_2/corporateActions/CorporateActionsSecurity.sol

Test - test/unitTests/layer_2/corporateActions/corporateActions.test.ts

```
it(' corporateActions Test', async () => {
    // Granting Role to account C
    accessControlFacet = accessControlFacet.connect(signer_A)
    await accessControlFacet.grantRole(_CORPORATE_ACTION_ROLE, account_C)
    // Using account C (with role)
    corporateActionsFacet = corporateActionsFacet.connect(signer_C)

    // add to list
    await corporateActionsFacet.addCorporateAction(actionType, actionData)
    await corporateActionsFacet.addCorporateAction(actionType, actionData)

    // check list members
    const listCount = await corporateActionsFacet.getCorporateActionCount()
    const listMembers = await corporateActionsFacet.getCorporateActionIds(
        0,
        listCount
    )
    const listCountByType =
        await corporateActionsFacet.getCorporateActionCountByType(
            actionType
        )
    const listMembersByType =
        await corporateActionsFacet.getCorporateActionIdsByType(
            actionType,
            0,
            listCount
        )
    const corporateAction = await corporateActionsFacet.getCorporateAction(
        corporateActionId_1
    )

    expect(listCount).to.equal(2)

 })
```

**Details:** The vulnerability arises from the absence of checks to verify whether a corporate action being added already exists in the contract's records. Without this validation, the contract accepts duplicate corporate actions, which can lead to inconsistencies and errors in processing corporate actions.

**Impact:** This vulnerability poses a risk of data inconsistency and operational inefficiency. Duplicate corporate actions may result in incorrect processing of actions, leading to inaccuracies in record-keeping and potentially causing confusion among stakeholders.

**Tools Used:** Manual Review

**Recommendations**: Enhance the checkISIN modifier to perform comprehensive ISIN validation, including format and uniqueness checks. Chain-link functions can be used to verify the ISIN off chain.

## Automated Test Findings

Below are findings using an automation tool, these findings should be addressed no lesser than the manual findings.

## General Overview

```
Total number of contracts in source files: 107
Number of contracts in dependencies: 19
Source lines of code (SLOC) in source files: 8041
Source lines of code (SLOC) in dependencies: 887
Number of  assembly lines: 0
Number of optimization issues: 0
Number of informational issues: 169
Number of low issues: 7
Number of medium issues: 11
Number of high issues: 0


Use: Openzeppelin-Ownable
ERCs: ERC165

+----------------------------+-------------+--------+------------+--------------+--------------------+
| Name                       | # functions | ERCS   | ERC20 info | Complex code |           Features |
+----------------------------+-------------+--------+------------+--------------+--------------------+
| Diamond                    |          38 |        |            |           No |        Receive ETH |
|                            |             |        |            |              |           Assembly |
| DiamondFacet               |          65 | ERC165 |            |           No |           Assembly |
| Factory                    |          11 |        |            |           No | Tokens interaction |
| IERC173                    |           2 |        |            |           No |                    |
| ERC1594                    |         105 |        |            |           No |           Assembly |
| ERC1643                    |          45 |        |            |           No |           Assembly |
| ERC1644                    |          94 |        |            |           No |           Assembly |
| ERC20                      |          98 |        |            |           No |           Assembly |
| Cap                        |          67 |        |            |           No |           Assembly |
| LibCommon                  |           5 |        |            |           No |                    |
| ControlList                |          50 |        |            |           No |           Assembly |
| IERC1410                   |          22 |        |            |           No |                    |
| IAssociation               |           3 |        |            |           No |                    |
| IAssociationStorageWrapper |           0 |        |            |           No |                    |
| Lock                       |          86 |        |            |           No |           Assembly |
| Snapshots                  |         113 |        |            |           No |           Assembly |
| ERC1410ScheduledSnapshot   |         185 |        |            |           No |           Assembly |
| CorporateActionsSecurity   |         144 |        |            |           No |           Assembly |
| ScheduledSnapshots         |         139 |        |            |           No |           Assembly |
| BondUSA                    |         162 |        |            |           No |           Assembly |
| EquityUSA                  |         169 |        |            |           No |           Assembly |
| TransferAndLock            |         150 |        |            |           No |           Assembly |
| Proxies                    |           0 |        |            |           No |                    |
| BusinessLogicResolver      |          92 |        |            |           No |           Assembly |
+----------------------------+-------------+--------+------------+--------------+--------------------+
INFO:Slither:. analyzed (126 contracts)
```

## Findings Summary

```
Summary
 - [locked-ether](#locked-ether) (1 results) (Medium)
 - [unused-return](#unused-return) (10 results) (Medium)
 - [reentrancy-events](#reentrancy-events) (2 results) (Low)
 - [timestamp](#timestamp) (5 results) (Low)
 - [assembly](#assembly) (21 results) (Informational)
 - [similar-names](#similar-names) (148 results) (Informational)
```

### AFN:1

### Finding Category: *Locked Ether*

**Description:** Contract locking ether found but does not have a function to withdraw the ether.

**Impact:** Medium

**Confidence:** High

**Code:**

```
Contract [Diamond](contracts/diamond/Diamond.sol#L17-L56) has payable functions:
 - [Diamond.constructor(IBusinessLogicResolver,bytes32[],IDiamond.Rbac[])](contracts/diamond/Diamond.sol#L18-L24)
 - [Diamond.fallback()](contracts/diamond/Diamond.sol#L29-L53)
 - [Diamond.receive()](contracts/diamond/Diamond.sol#L55)
```

**Location:** contracts/diamond/Diamond.sol#L17-L56

**Recommendation:** Ensure proper handling of locked ether to mitigate liquidity issues and potential vulnerabilities. Consider implementing mechanisms for secure fund management and liquidity provision.

## AFN:2

## Finding Category: *Unused Return*

**Description:**This finding indicates that there are return statements present in the smart contract code but are not being utilized. It's categorized as a medium severity issue, implying potential inefficiencies or unintended behavior resulting from unused return statements.

**Impact:** Medium

**Confidence:** Medium

**Code:**

```
 - [ ] ID-1
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[IControlList(securityAddress_).initialize_ControlList(_securityData.isWhiteList)](contracts/factory/Factory.sol#L171-L173)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-2
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[IERC20(securityAddress_).initialize_ERC20(erc20Metadata)](contracts/factory/Factory.sol#L191)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-3
[Factory.deployBond(IFactory.BondData,FactoryRegulationData)](contracts/factory/Factory.sol#L124-L156) ignores return value by
[IBondUSA(bondAddress_)._initialize_bondUSA(_bondData.bondDetails,_bondData.couponDetails,buildRegulationData(_factoryRegulationData.regulat
ionType,_factoryRegulationData.regulationSubType),_factoryRegulationData.additionalSecurityData)](contracts/factory/Factory.sol#L140-L148)

contracts/factory/Factory.sol#L124-L156
```

**Location:** contracts/factory/Factory

```
 - [ ] ID-4
[LockStorageWrapper._releaseByPartition(bytes32,uint256,address)](contracts/layer_1/lock/LockStorageWrapper.sol#L86-L125) ignores return
value by [lockStorage.lockIds[_tokenHolder][_partition].remove(lock.id)](contracts/layer_1/lock/LockStorageWrapper.sol#L104)

contracts/layer_1/lock/LockStorageWrapper.sol#L86-L125


 - [ ] ID-5
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[IERC1410Basic(securityAddress_).initialize_ERC1410_Basic(_securityData.isMultiPartition)](contracts/factory/Factory.sol#L176-L178)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-6
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[IERC1644(securityAddress_).initialize_ERC1644(_securityData.isControllable)](contracts/factory/Factory.sol#L181-L183)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-7
[Factory.deployEquity(IFactory.EquityData,FactoryRegulationData)](contracts/factory/Factory.sol#L87-L121) ignores return value by
[IEquityUSA(equityAddress_)._initialize_equityUSA(_equityData.equityDetails,buildRegulationData(_factoryRegulationData.regulationType,_fac
ryRegulationData.regulationSubType),_factoryRegulationData.additionalSecurityData)](contracts/factory/Factory.sol#L106-L113)

contracts/factory/Factory.sol#L87-L121
```

```
 - [ ] ID-8
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[ICap(securityAddress_).initialize_Cap(_securityData.maxSupply,new ICap.PartitionCap[](0))](contracts/factory/Factory.sol#L197-L200)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-9
[Factory._deploySecurity(IFactory.SecurityData,IFactory.SecurityType)](contracts/factory/Factory.sol#L158-L201) ignores return value by
[IERC1594(securityAddress_).initialize_ERC1594()](contracts/factory/Factory.sol#L194)

contracts/factory/Factory.sol#L158-L201


 - [ ] ID-10
[LockStorageWrapper._lockByPartition(bytes32,uint256,address,uint256)](contracts/layer_1/lock/LockStorageWrapper.sol#L63-L84) ignores return
value by [lockStorage.lockIds[_tokenHolder][_partition].add(lockId_)](contracts/layer_1/lock/LockStorageWrapper.sol#L78)

contracts/layer_1/lock/LockStorageWrapper.sol#L63-L84
```

**Location:** contracts/layer_1/lock/

**Recommendation:** Identify and remove all unused return statements from the smart contract code. Unused return statements can lead to inefficiencies and unintended behavior. Review the contract's logic to ensure proper utilization of return statements where necessary

## AFN:3

## Finding Category: *Reentrancy Events*

**Description**:Reentrancy is a vulnerability where an external contract can invoke a function in the current contract before the previous invocation completes.

**Impact:** Low

**Confidence:** Medium

**Code:**

```
## reentrancy-events
Impact: Low
Confidence: Medium
 - [ ] ID-11
Reentrancy in [Factory.deployEquity(IFactory.EquityData,FactoryRegulationData)](contracts/factory/Factory.sol#L87-L121):
    External calls:
    - [equityAddress_ = _deploySecurity(_equityData.security,SecurityType.Equity)](contracts/factory/Factory.sol#L101-L104)
        - [IControlList(securityAddress_).initialize_ControlList(_securityData.isWhiteList)](contracts/factory/Factory.sol#L171-L173)
        - [IERC1410Basic(securityAddress_).initialize_ERC1410_Basic(_securityData.isMultiPartition)](contracts/factory/Factory.sol#L176-
L178)
        - [IERC1644(securityAddress_).initialize_ERC1644(_securityData.isControllable)](contracts/factory/Factory.sol#L181-L183)
        - [IERC20(securityAddress_).initialize_ERC20(erc20Metadata)](contracts/factory/Factory.sol#L191)
        - [IERC1594(securityAddress_).initialize_ERC1594()](contracts/factory/Factory.sol#L194)
        - [ICap(securityAddress_).initialize_Cap(_securityData.maxSupply,new ICap.PartitionCap[](0))](contracts/factory/Factory.sol#L197-
L200)
    -
[IEquityUSA(equityAddress_)._initialize_equityUSA(_equityData.equityDetails,buildRegulationData(_factoryRegulationData.regulationType,_facto
ryRegulationData.regulationSubType),_factoryRegulationData.additionalSecurityData)](contracts/factory/Factory.sol#L106-L113)
    Event emitted after the call(s):
    - [EquityDeployed(_msgSender(),equityAddress_,_equityData,_factoryRegulationData)](contracts/factory/Factory.sol#L115-L120)

contracts/factory/Factory.sol#L87-L121
```

```
 - [ ] ID-12
Reentrancy in [Factory.deployBond(IFactory.BondData,FactoryRegulationData)](contracts/factory/Factory.sol#L124-L156):
    External calls:
    - [bondAddress_ = _deploySecurity(_bondData.security,SecurityType.Bond)](contracts/factory/Factory.sol#L138)
        - [IControlList(securityAddress_).initialize_ControlList(_securityData.isWhiteList)](contracts/factory/Factory.sol#L171-L173)
        - [IERC1410Basic(securityAddress_).initialize_ERC1410_Basic(_securityData.isMultiPartition)](contracts/factory/Factory.sol#L176-
L178)
        - [IERC1644(securityAddress_).initialize_ERC1644(_securityData.isControllable)](contracts/factory/Factory.sol#L181-L183)
        - [IERC20(securityAddress_).initialize_ERC20(erc20Metadata)](contracts/factory/Factory.sol#L191)
        - [IERC1594(securityAddress_).initialize_ERC1594()](contracts/factory/Factory.sol#L194)
        - [ICap(securityAddress_).initialize_Cap(_securityData.maxSupply,new ICap.PartitionCap[](0))](contracts/factory/Factory.sol#L197-
L200)
    -
[IBondUSA(bondAddress_)._initialize_bondUSA(_bondData.bondDetails,_bondData.couponDetails,buildRegulationData(_factoryRegulationData.regulat
ionType,_factoryRegulationData.regulationSubType),_factoryRegulationData.additionalSecurityData)](contracts/factory/Factory.sol#L140-L148)
    Event emitted after the call(s):
    - [BondDeployed(_msgSender(),bondAddress_,_bondData,_factoryRegulationData)](contracts/factory/Factory.sol#L150-L155)

contracts/factory/Factory.sol#L124-L156
```

**Location:** contracts/factory/Factory.

**Recommendation:** Mitigate the risk of reentrancy vulnerabilities by implementing proper checks and safeguards in the contract code. Utilize techniques such as the "Checks-Effects-Interactions" pattern to prevent reentrancy attacks.

## AFN:4

## Finding Category: *Timestamp*

**Description**: This finding indicates that the smart contract relies on timestamps for certain functionalities. Timestamp dependency can introduce vulnerabilities related to manipulation or inconsistencies in time-based logic.

**Confidence:** Medium

**Code:**

```
## timestamp
Impact: Low
Confidence: Medium
 - [ ] ID-13
[BondStorageWrapper._getCouponFor(uint256,address)](contracts/layer_2/bond/BondStorageWrapper.sol#L129-L146) uses timestamp for compari
    Dangerous comparisons:
    - [registeredCoupon.coupon.recordDate < _blockTimestamp()](contracts/layer_2/bond/BondStorageWrapper.sol#L139)

contracts/layer_2/bond/BondStorageWrapper.sol#L129-L146


 - [ ] ID-14
[ScheduledSnapshotsStorageWrapper._triggerScheduledSnapshots(uint256)]
(contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L81-L125) uses timestamp for comparisons
    Dangerous comparisons:
    - [currentScheduledSnapshot.scheduledTimestamp < _blockTimestamp()]
(contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L107)

contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L81-L125
```

**Locations:** contracts/layer_2/bond/BondStorageWrapper, contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper

```
 - [ ] ID-15
[EquityStorageWrapper._getVotingFor(uint256,address)](contracts/layer_2/equity/EquityStorageWrapper.sol#L163-L179) uses timestamp for
comparisons
    Dangerous comparisons:
    - [registeredVoting.voting.recordDate < _blockTimestamp()](contracts/layer_2/equity/EquityStorageWrapper.sol#L172)

contracts/layer_2/equity/EquityStorageWrapper.sol#L163-L179


 - [ ] ID-16
[EquityStorageWrapper._getDividendsFor(uint256,address)](contracts/layer_2/equity/EquityStorageWrapper.sol#L92-L111) uses timestamp for
comparisons
    Dangerous comparisons:
    - [registeredDividend.dividend.recordDate < _blockTimestamp()](contracts/layer_2/equity/EquityStorageWrapper.sol#L104)

contracts/layer_2/equity/EquityStorageWrapper.sol#L92-L111


 - [ ] ID-17
[LockStorageWrapper._isLockedExpirationTimestamp(bytes32,address,uint256)](contracts/layer_1/lock/LockStorageWrapper.sol#L231-L241) uses
timestamp for comparisons
    Dangerous comparisons:
    - [lock.expirationTimestamp > block.timestamp](contracts/layer_1/lock/LockStorageWrapper.sol#L238)

contracts/layer_1/lock/LockStorageWrapper.sol#L231-L241
```

**Locations:** contracts/layer_2/equity/EquityStorageWrapper,

**Recommendation:** Minimize reliance on timestamps for critical functionalities within the smart contract. Consider alternative approaches or additional checks to prevent manipulation or inconsistencies in time-based logic.

## AFN:5

## Finding Category: *Assembly*

**Description**:Reentrancy is a vulnerability where an external contract can invoke a function in the current contract before the previous invocation completes.

**Impact:** Informational

**Confidence:** High

## Code:

```
## assembly
Impact: Informational
Confidence: High
 - [ ] ID-18
[ERC1643._getERC1643Storage()](contracts/layer_1/ERC1400/ERC1643/ERC1643.sol#L166-L177) uses assembly
    - [INLINE ASM](contracts/layer_1/ERC1400/ERC1643/ERC1643.sol#L174-L176)

contracts/layer_1/ERC1400/ERC1643/ERC1643.sol#L166-L177


 - [ ] ID-19
[ScheduledSnapshotsStorageWrapper._scheduledSnapshotsStorage()]
(contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L214-L225) uses assembly
    - [INLINE ASM](contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L222-L224)

contracts/layer_2/scheduledSnapshots/ScheduledSnapshotsStorageWrapper.sol#L214-L225


 - [ ] ID-20
[Diamond.fallback()](contracts/diamond/Diamond.sol#L29-L53) uses assembly
    - [INLINE ASM](contracts/diamond/Diamond.sol#L37-L52)

contracts/diamond/Diamond.sol#L29-L53
```

```
 - [ ] ID-21
[EquityStorageWrapper._equityStorage()](contracts/layer_2/equity/EquityStorageWrapper.sol#L191-L202) uses assembly
    - [INLINE ASM](contracts/layer_2/equity/EquityStorageWrapper.sol#L199-L201)

contracts/layer_2/equity/EquityStorageWrapper.sol#L191-L202


 - [ ] ID-22
[SecurityStorageWrapper._securityStorage()](contracts/layer_3/security/SecurityStorageWrapper.sol#L31-L41) uses assembly
    - [INLINE ASM](contracts/layer_3/security/SecurityStorageWrapper.sol#L38-L40)

contracts/layer_3/security/SecurityStorageWrapper.sol#L31-L41


 - [ ] ID-23
[SnapshotsStorageWrapper._snapshotStorage()](contracts/layer_1/snapshots/SnapshotsStorageWrapper.sol#L257-L268) uses assembly
    - [INLINE ASM](contracts/layer_1/snapshots/SnapshotsStorageWrapper.sol#L265-L267)

contracts/layer_1/snapshots/SnapshotsStorageWrapper.sol#L257-L268


 - [ ] ID-24
[LockStorageWrapper._lockStorage()](contracts/layer_1/lock/LockStorageWrapper.sol#L243-L254) uses assembly
    - [INLINE ASM](contracts/layer_1/lock/LockStorageWrapper.sol#L251-L253)

contracts/layer_1/lock/LockStorageWrapper.sol#L243-L254
```

```
 - [ ] ID-36
[CorporateActionsStorageWrapper._corporateActionsStorage()](contracts/layer_1/corporateActions/CorporateActionsStorageWrapper.sol#L178-L189)
uses assembly
    - [INLINE ASM](contracts/layer_1/corporateActions/CorporateActionsStorageWrapper.sol#L186-L188)

contracts/layer_1/corporateActions/CorporateActionsStorageWrapper.sol#L178-L189


 - [ ] ID-37
[ERC1644StorageWrapper._getErc1644Storage()](contracts/layer_1/ERC1400/ERC1644/ERC1644StorageWrapper.sol#L80-L90) uses assembly
    - [INLINE ASM](contracts/layer_1/ERC1400/ERC1644/ERC1644StorageWrapper.sol#L87-L89)

contracts/layer_1/ERC1400/ERC1644/ERC1644StorageWrapper.sol#L80-L90


 - [ ] ID-38
[ERC1594StorageWrapper._getErc1594Storage()](contracts/layer_1/ERC1400/ERC1594/ERC1594StorageWrapper.sol#L188-L199) uses assembly
    - [INLINE ASM](contracts/layer_1/ERC1400/ERC1594/ERC1594StorageWrapper.sol#L196-L198)

contracts/layer_1/ERC1400/ERC1594/ERC1594StorageWrapper.sol#L188-L199
```

**Locations:** contracts/layer_1/ERC1400/ERC1643/ERC1643,

contracts/layer_2/scheduledSnapshots/

contracts/diamond/Diamond,

contracts/layer_1/pause/

contracts/resolver/

**Recommendation:** Review the usage of assembly code within the smart contract and ensure it is necessary for optimization or specific operations. While assembly can be useful for efficiency, ensure it does not introduce security vulnerabilities

# Terms and Meaning

**POC:** Proof of Concept

**FN:** Findings

**AFN:** Automatic Findings

**Impact:** The implication of the finding being exploited

# DISCLAIMER

*This report is not an endorsement or indictment of any particular project or team, and the report does not guarantee the security of any particular project. Therefore, should not be interpreted as having any bearing on, the potential economics of a token, token sale or any other product, service or other asset.*

*In addition, this report does not provide any warranty or representation to any Third-Party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. Hence, no third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.*

*Specifically, for the avoidance of doubt, this report does not constitute investment advice. Hence, it is not intended to be relied upon as investment advice, an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project as the scope of our review is limited to the token module of the solana-cli residing at the specific address. We owe no duty to any Third-Party by virtue of publishing these Reports. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.*

## CONTACT US

A&D Forensics
+234 909 550 3040
www.adforensics.com.ng
contactus@adforensics.com.ng
No 3. Rabat Street, Wuse Zone 6, FCT Abuja, Nigeria