Audited by

CredShields

Smart Contract Audit

12th August, 2025 • CONFIDENTIAL

Description

This document details the process and result of the Smart Contract audit performed by CredShields Technologies PTE. LTD. on behalf of Tarmiiz between July 21st, 2025, and July 26th, 2025. A retest was performed on July 29th, 2025.

Author
Shashank (Co-founder, CredShields) shashank@CredShields.com

Reviewers
Aditya Dixit (Research Team Lead), Shreyas Koli(Auditor), Naman Jain (Auditor), Sanket Salavi (Auditor), Yash Shah (Auditor), Prasad Kuri (Auditor)

Prepared for
Tarmiiz

Table of Contents

1. Executive Summary ---------------------

Tarmiiz engaged CredShields to perform a smart contract audit from July 21st, 2025, to July 26th, 2025. During this timeframe, 20 vulnerabilities were identified. **A retest was performed on July 29th, 2025, and all the bugs have been addressed.**

During the audit, 5 vulnerabilities were found with a severity rating of either High or Critical. These vulnerabilities represent the greatest immediate risk to "Tarmiiz" and should be prioritized for remediation.

The table below shows the in-scope assets and a breakdown of findings by severity per asset. Section 2.3 contains more information on how severity is calculated.

Assets in Scope	Critical	High	Medium	Low	info	Gas	Σ
Vault Contracts	2	3	4	9	0	2	**20**
	2	**3**	**4**	**9**	**0**	**2**	**20**

Table: Vulnerabilities Per Asset in Scope

The CredShields team conducted the security audit to focus on identifying vulnerabilities in Vault Contract's scope during the testing window while abiding by the policies set forth by Tarmiiz's team.

State of Security

To maintain a robust security posture, it is essential to continuously review and improve upon current security processes. Utilizing CredShields' continuous audit feature allows both Tarmiiz's internal security and development teams to not only identify specific vulnerabilities but also gain a deeper understanding of the current security threat landscape.

To ensure that vulnerabilities are not introduced when new features are added or code is refactored, we recommend conducting regular security assessments. Additionally, by analyzing the root cause of resolved vulnerabilities, the internal teams at Tarmiiz can implement both manual and automated procedures to eliminate entire classes of vulnerabilities in the future. By taking a proactive approach, Tarmiiz can future-proof its security posture and protect its assets.

2. The Methodology --------------------

Tarmiiz engaged CredShields to perform a Tarmiiz Vault Smart Contract audit. The following sections cover how the engagement was put together and executed.

2.1 Preparation Phase

The CredShields team meticulously reviewed all provided documents and comments in the smart contract code to gain a thorough understanding of the contract's features and functionalities. They meticulously examined all functions and created a mind map to systematically identify potential security vulnerabilities, prioritizing those that were more critical and business-sensitive for the refactored code. To confirm their findings, the team deployed a self-hosted version of the smart contract and performed verifications and validations during the audit phase.

A testing window from July 21st, 2025, to July 26th, 2025, was agreed upon during the preparation phase.

2.1.1 Scope

During the preparation phase, the following scope for the engagement was agreed upon:

IN SCOPE ASSETS
https://github.com/Tarmiiz/Vault-Contracts/tree/4dac763f7291f64fbff28d729cceefc53e07c770

2.1.2 Documentation

Tarmiiz team provided all the required documentation and helped the Credshields team with questions throughout the audit process.

2.1.3 Audit Goals

CredShields employs a combination of in-house tools and thorough manual review processes to deliver comprehensive smart contract security audits. The majority of the audit involves manual inspection of the contract's source code, guided by OWASP's Smart Contract Security Weakness Enumeration (SCWE) framework and an extended, self-developed checklist built from industry best practices. The team focuses on deeply understanding the contract's core logic, designing targeted test cases, and assessing business logic for potential vulnerabilities across OWASP's identified weakness classes.

CredShields aligns its auditing methodology with the OWASP Smart Contract Security projects, including the Smart Contract Security Verification Standard (SCSVS), the Smart Contract Weakness Enumeration (SCWE), and the Smart Contract Secure Testing Guide (SCSTG). These frameworks, actively contributed to and co-developed by the CredShields team, aim to bring consistency, clarity, and depth to smart contract security assessments. By adhering to these OWASP standards, we ensure that each audit is performed against a transparent, community-driven, and technically robust baseline. This approach enables us to deliver structured, high-quality audits that address both common and complex smart contract vulnerabilities systematically.

2.2 Retesting Phase

Tarmiiz is actively partnering with CredShields to validate the remediations implemented towards the discovered vulnerabilities.

2.3 Vulnerability classification and severity

CredShields follows OWASP's Risk Rating Methodology to determine the risk associated with discovered vulnerabilities. This approach considers two factors - Likelihood and Impact - which are evaluated with three possible values - **Low**, **Medium**, and **High**, based on factors such as Threat agents, Vulnerability factors, and Technical and Business Impacts. The overall severity of the risk is calculated by combining the likelihood and impact estimates.

Overall Risk Severity				
Impact	HIGH	● Medium	● High	● Critical
	MEDIUM	● Low	● Medium	● High
	LOW	● None	● Low	● Medium
		LOW	MEDIUM	HIGH
Likelihood				

Overall, the categories can be defined as described below -

1. Informational

We prioritize technical excellence and pay attention to detail in our coding practices. Our guidelines, standards, and best practices help ensure software stability and reliability. Informational vulnerabilities are opportunities for improvement and do not pose a direct risk to the contract. Code maintainers should use their own judgment on whether to address them.

2. Low

Low-risk vulnerabilities are those that either have a small impact or can't be exploited repeatedly or those the client considers insignificant based on their specific business circumstances.

3. Medium

Medium-severity vulnerabilities are those caused by weak or flawed logic in the code and can lead to exfiltration or modification of private user information. These vulnerabilities can harm the client's reputation under certain conditions and should be fixed within a specified timeframe.

4. **High**

   High-severity vulnerabilities pose a significant risk to the Smart Contract and the organization. They can result in the loss of funds for some users, may or may not require specific conditions, and are more complex to exploit. These vulnerabilities can harm the client's reputation and should be fixed immediately.

5. **Critical**

   Critical issues are directly exploitable bugs or security vulnerabilities that do not require specific conditions. They often result in the loss of funds and Ether from Smart Contracts or users and put sensitive user information at risk of compromise or modification. The client's reputation and financial stability will be severely impacted if these issues are not addressed immediately.

6. **Gas**

   To address the risk and volatility of smart contracts and the use of gas as a method of payment, CredShields has introduced a "Gas" severity category. This category deals with optimizing code and refactoring to conserve gas.

2.4 CredShields staff

The following individual at CredShields managed this engagement and produced this report:

- Shashank, Co-founder CredShields shashank@CredShields.com

Please feel free to contact this individual with any questions or concerns you have about the engagement or this document.

3. Findings Summary -------------------

This chapter contains the results of the security assessment. Findings are sorted by their severity and grouped by asset and OWASP SCWE classification. Each asset section includes a summary highlighting the key risks and observations. The table in the executive summary presents the total number of identified security vulnerabilities per asset, categorized by risk severity based on the OWASP Smart Contract Security Weakness Enumeration framework.

3.1 Findings Overview

3.1.1 Vulnerability Summary

During the security assessment, 20 security vulnerabilities were identified in the asset.

VULNERABILITY TITLE	SEVERITY	SCWE | Vulnerability Type
Owner Can Fully Control User Accounts via Overpowered API Role	Critical	Centralization Risk
Any User Can Change Password of Any Account	Critical	Sensitive Data Exposure (SCWE-021)
Missing Ownership Validation Before Attachment	High	Business Logic (SC03-LogicErrors)
Large Dividend Distributions Can Exceed Gas Limits and Revert	High	Denial of Service (SCWE-058)
Inconsistent Owner Validation Blocks Document Removal	High	Business Logic (SC03-LogicErrors)
Missing Validation for Dual Inputs (values & tokens) in orderMake	Medium	Business Logic (SC03-LogicErrors)
Unnecessary Array Initialization Can Trigger Compilation Errors	Medium	Compilation Error

Missing Ownership Validation in documentDetach Function	Medium	Business Logic (SC03-LogicErrors)
Privileged Actor Can Mint or Burn Tokens for Inactive Assets	Medium	Missing Input Validation (SC04-Lack Of Input Validation)
Redundant Execution Check in orderMake Function	Low	Code With No Effects (SCWE-062)
Stale Document Data Not Deleted from Storage	Low	Business Logic (SC03-LogicErrors)
Unreachable Order Cancellation Logic	Low	Code With No Effects (SCWE-062)
Event Emission After Invalid State Change	Low	Business Logic (SC03-LogicErrors)
Role Entry Not Removed from Roles List	Low	Business Logic (SC03-LogicErrors)
Function Contains Unused Transfer Logic with No External Effects	Low	Code With No Effects (SCWE-062)
Missing Events in Important Functions	Low	Missing Best Practices (SCWE-063)
Outdated Pragma	Low	Outdated Compiler Version (SCWE-061)
Missing Zero Address Validations	Low	Missing Input Validation (SC04-Lack Of Input Validation)
Missing Break Statement in detach Loop	Gas	Gas Optimization
Inefficient Array Element Removal in Role Index	Gas	Gas Optimization

Table: Findings in Smart Contracts

4. Remediation Status -----------------

Tarmiiz is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. **A retest was performed on July 29th, 2025, and all the issues have been addressed.**

Also, the table shows the remediation status of each finding.

VULNERABILITY TITLE	SEVERITY	REMEDIATION STATUS
Owner Can Fully Control User Accounts via Overpowered API Role	Critical	Won't Fix [July 29, 2025]
Any User Can Change Password of Any Account	Critical	Fixed [July 29, 2025]
Missing Ownership Validation Before Attachment	High	Fixed [July 29, 2025]
Large Dividend Distributions Can Exceed Gas Limits and Revert	High	Fixed [July 29, 2025]
Inconsistent Owner Validation Blocks Document Removal	High	Fixed [July 29, 2025]
Missing Validation for Dual Inputs (values & tokens) in orderMake	Medium	Fixed [July 29, 2025]
Unnecessary Array Initialization Can Trigger Compilation Errors	Medium	Fixed [July 29, 2025]
Missing Ownership Validation in documentDetach Function	Medium	Fixed [July 29, 2025]
Privileged Actor Can Mint or Burn Tokens for Inactive Assets	Medium	Won't Fix [July 29, 2025]
Redundant Execution Check in orderMake Function	Low	Fixed [July 29, 2025]
Stale Document Data Not Deleted from Storage	Low	Won't Fix [July 29, 2025]

Unreachable Order Cancellation Logic	Low	Fixed [July 29, 2025]
Event Emission After Invalid State Change	Low	Fixed [July 29, 2025]
Role Entry Not Removed from Roles List	Low	Fixed [July 29, 2025]
Function Contains Unused Transfer Logic with No External Effects	Low	Won't Fix [July 29, 2025]
Missing Events in Important Functions	Low	Fixed [July 29, 2025]
Outdated Pragma	Low	Fixed [July 29, 2025]
Missing Zero Address Validations	Low	Fixed [July 29, 2025]
Missing Break Statement in detach Loop	Gas	Fixed [July 29, 2025]
Inefficient Array Element Removal in Role Index	Gas	Fixed [July 29, 2025]

Table: Summary of findings and status of remediation

5. Bug Reports ------------------------

Bug ID #C001 [Won't Fix]

Owner Can Fully Control User Accounts via Overpowered API Role

Vulnerability Type
Centralization Risk

Severity
Critical

Description
In TarmiizVaultE, all critical user-facing functions are gated by a single _role = 'api', which the owner exclusively manages. This API role can perform sensitive actions including account creation, credit updates, document changes, and buy/sell orders—all without direct user interaction or consent. The lack of user-specific authorization checks delegates unchecked control to the owner or any address granted this role, undermining user autonomy and exposing the system to potential misuse or abuse.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol

Impacts
The owner or a compromised API address can impersonate users, alter their financial and identity data, and perform irreversible operations on their behalf without approval or notification.

Remediation
Decentralize authority by requiring per-user authorization and signed approvals for sensitive actions instead of relying solely on a global API role.

Retest

Client's comment: The Vault is a centralized deployment for any Token Issuer / Operator. It is like a centralized exchange, where the vault operator holds all privileges in regards to the assets and users. The Vault is not a multi-tenant system, which will host multiple token issuers and users, each token issuer (vault operator) will have their own instance of the Vault, including Contracts, APIs, and other layers, such their own DB and IPFS instances.

Bug ID #C002 [Fixed]

Any User Can Change Password of Any Account

Vulnerability Type
Sensitive Data Exposure (SCWE-021)

Severity
Critical

Description
The UserTemplate.changePassword function relies on plaintext password verification using Strings.equal(oldPassword, _password) with both values exposed in the transaction. Since all onchain calldata is publicly accessible, any observer can read the oldPassword argument from the transaction history and use it to call changePassword, bypassing authentication and taking over the account. Moreover, the function lacks a binding between users and passwords, so any observer can impersonate any user with a known password.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/UserTemplate.sol#L43-L46

Impacts
Any attacker monitoring onchain transactions can extract a user's password, call changePassword, and permanently lock the user out of their account.

Remediation
It is recommended to implement onlyOwner modifier in changePassword function to make sure it is only called the userContract.

Retest
This issue has been fixed by adding onlyOwner to the function.

Bug ID #H001 [Fixed]

Missing Ownership Validation Before Attachment

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
High

Description
The documentAttach function in TarmiizVaultE calls documentsContract.attach without validating whether the document is already attached to the same or a different owner. In Documents, the attach function unconditionally overwrites _documentOwner[documentId] and appends the documentId to _ownerDocuments[owner], even if the document was already attached. This can lead to ownership overwrites, duplication in the owner's document list, and potential data inconsistency due to repeated attachments or reassignments without checks.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L237

Impacts
Allows repeated or unauthorized reassignment of document ownership, leading to duplicated references and broken ownership tracking.

Remediation
To mitigate this issue add a validation in documentAttach to ensure the document is not already attached to an owner, or to the same owner, before proceeding with the attach call.

Retest
This issue has been fixed by introducing below validation.

```
+    require(documentsContract.getOwner(documentCID) == address(0), "Document is
     attached to an asset or user");
```

Bug ID #H002 [Fixed]

Large Dividend Distributions Can Exceed Gas Limits and Revert

Vulnerability Type
Denial of Service (SCWE-058)

Severity
High

Description
The TarmiizVaultE.dividendAdd function, along with its dependencies in AssetsContract and DividendTokenTemplate, relies on multiple unbounded for loops to iterate over all token holders and distribute dividends. These loops execute in a single transaction and grow linearly with the number of holders. As the user base scales, the cumulative gas cost can exceed block gas limits, causing the entire transaction to revert and making it impossible to distribute dividends.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L184-L219

Impacts
Dividends cannot be distributed once the number of holders grows beyond a certain threshold, effectively freezing the payout mechanism and halting expected earnings for users.

Remediation
It is recommended to refactor the dividend logic to use a batching mechanism that processes payments across multiple transactions.

Retest
This issue has been fixed by distributing dividends in batches.

Bug ID #H003 [Fixed]

Inconsistent Owner Validation Blocks Document Removal

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
High

Description
The documentRemove function in the TarmiivalutE contract includes a validation that requires the document to not be attached to any owner (require(documentsContract.getOwner(documentCID) == address(0))) before it can be removed. However, in the documentsContract.remove function, the removal logic depends on the document being attached to an owner, as it attempts to locate the document within the _ownerDocuments mapping to delete it. This creates a logical contradiction: the outer function requires no owner, while the internal function requires an owner to proceed with removal. As a result, valid document removals are blocked, and transactions revert due to conflicting conditions, effectively breaking document cleanup functionality.

Affected Code
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L229

Impacts
Document removal becomes impossible, leading to broken functionality.

Remediation
It is recommended to align the owner validation logic between TarmiizvalutE.documentRemove and documentsContract.remove. Specifically, revise the outer require condition to allow removal only when a valid owner exists, matching the internal logic of the remove function.

Retest
This issue has been fixed by validating if the owner exists.

Bug ID #M001 [Fixed]

Missing Validation for Dual Inputs (values & tokens) in orderMake

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
Medium

Description
The orderMake function does not validate against both tokens and value being non-zero simultaneously, even though its logic assumes only one will be provided. If both are supplied, neither conditional branch executes, resulting in requestTokens and requestValue remaining zero. This causes the subsequent call to ordersContract.make to revert due to the enforced condition require(tokens > 0, "Tokens must be > 0") in the make function. Consequently, user orders will unexpectedly fail at a later stage rather than being rejected upfront, leading to poor user experience and wasted gas.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L114-L170

Impacts
Causes failed transactions and wasted gas when both inputs are non-zero, due to delayed validation in downstream logic.

Remediation
To mitigate this issue, it is recommended to add an explicit validation in orderMake function that ensures only one of the tokens or values is non-zero, preventing invalid input combinations before further processing.

Retest
This issue has been fixed by introducing below validation.

```
+       require((tokens != 0 && value == 0) || (value != 0 && tokens == 0), "Either tokens or value
      must be supplied");
```

Bug ID #M002 [Fixed]

Unnecessary Array Initialization Can Trigger Compilation Errors

Vulnerability Type
Compilation Error

Severity
Medium

Description
The contract contains redundant zero-length checks that attempt to initialize dynamic storage arrays (assets, priceHistroy, and holders) using new Type , which is unsupported in Solidity and leads to compilation errors. Solidity automatically initializes dynamic arrays in storage, and manual reinitialization is unnecessary and incorrect.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/UserTemplate.sol#L50-L52
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/BasicTokenTemplate.sol#L83-L85
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/BasicTokenTemplate.sol#L141-L143

Impacts
The contract fails to compile, preventing deployment and blocking all dependent functionality.

Remediation
Remove all redundant length == 0 checks and associated array initialization logic.

Retest
This issue has been fixed by removing the array initialization logic.

Bug ID #M003 [Fixed]

Missing Ownership Validation in documentDetach Function

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
Medium

Description
The documentDetach function in TarmiizVaultE does not verify whether the specified documentCID is actually attached to the provided owner before attempting to detach it. As a result, the detach function in the Documents contract proceeds to delete ownership mappings and remove entries from the owner's document list regardless of whether the document was associated with that owner. This lack of validation allows unauthorized detachment of unrelated or non-existent document associations, violating access control assumptions and potentially leading to inconsistent state and data loss.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L243-L249

Impacts
May allow detachment of documents not owned by the specified owner, leading to data integrity issues and unauthorized document state changes.

Remediation
To mitigate this issue, it is advisable to validate that the documentId is currently attached to the specified owner before proceeding with detachment. This check can be performed by verifying the ownership mapping prior to deletion.

Retest
This issue has been fixed by introducing below validation.

```
+    require(documentsContract.getOwner(documentCID) == owner, "Document is not
     attached to given owner");
```

Bug ID #M004 [Won't Fix]

Privileged Actor Can Mint or Burn Tokens for Inactive Assets

Vulnerability Type
Missing Input Validation (SC04-Lack Of Input Validation)

Severity
Medium

Description
In the mint and burn functions, the contract does not validate the state of the asset before updating its token supply. These functions can be executed by authorized actors via onlyRole(_role), yet they bypass checks on the Asset.state field. Assets marked as Suspended (3) or Deactivated (4) should be considered inactive and excluded from further minting or burning actions. The absence of this validation undermines the system's integrity rules around asset lifecycle management.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Assets.sol#L79-L87

Impacts
An authorized operator can mint or burn tokens tied to assets that are administratively suspended or deactivated, violating expected lifecycle constraints and potentially leading to governance or compliance issues.

Remediation
It is recommended to add checks to ensure that only assets in Active (2) state can be minted or burned.

Retest

Client's comment: The reason that tokens can be minted or burned at any given state is that the Vault operator can change the amount of tokens at any given state. An operator might want to "suspend" the token before minting or burning.

Bug ID #L001 [Fixed]

Redundant Execution Check in orderMake Function

Vulnerability Type
Code With No Effects (SCWE-062)

Severity
Low

Description
The orderMake function includes require statements to check the return values of _orderBuy and _orderSell, expecting them to indicate whether the order execution succeeded. However, both of these internal functions always return true unconditionally after performing their operations. As a result, the require(_executed, "Order execution failed") statements are redundant and serve no practical purpose, unnecessarily consuming gas and increasing contract complexity without providing functional benefit.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L150
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L158

Impacts
Causes unnecessary gas consumption due to ineffective validation and increases code complexity.

Remediation
It is advisable to remove the redundant require statements checking _executed, and eliminate the return value from _orderBuy and _orderSell to streamline execution and improve gas efficiency.

Retest
This issue has been fixed.

Bug ID #L002 [Won't Fix]

Stale Document Data Not Deleted from Storage

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
Low

Description
The remove function in the Documents contract marks a document as deleted by setting its state to 2 but does not delete the corresponding entry in the _documentById mapping. This means that although the document is logically marked as deleted, all its metadata remains accessible through public view functions such as getById, getByCID, and getAll. This behavior may unintentionally expose data that is expected to be removed and increases on-chain storage usage over time, especially in high-volume document environments.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Documents.sol#L45-L63

Impacts
Deleted document metadata remains accessible, potentially exposing sensitive information and increasing unnecessary storage costs.

Remediation
To mitigate this issue, it is recommended to delete the document entry from the _documentById mapping when removing a document, in addition to updating its state.

Retest

Client's comment: It is important to keep track of the documents uploaded, and hence keeping the document mappings, all except the _ownerDocuments, which directly returns the list of documents for the given owner address, used in some lookups at the API layer

Bug ID #L003 [Fixed]

Unreachable Order Cancellation Logic

Vulnerability Type
Code With No Effects (SCWE-062)

Severity
Low

Description
The orderCancel function is designed to cancel an open or partially filled order, but in the current implementation of orderMake, every order is immediately marked as filled (state 3) upon creation. As a result, the precondition require(_order.state != 3, "Order is already filled") in orderCancel will always fail, making it impossible to cancel any order and the orderCancel function behaves as dead code.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L172-L180

Impacts
Prevents any order from being canceled, rendering the cancellation feature non-functional and misleading.

Remediation
To mitigate this issue, it is recommended to either revise the order lifecycle to allow cancelable states or remove the orderCancel function entirely if order cancellation is not intended to be supported, in order to avoid maintaining non-functional code.

Retest
This issue has been fixed by removing the orderCancel function.

Bug ID #L004 [Fixed]

Event Emission After Invalid State Change

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
Low

Description
In the documentRemove function, the document is first deleted using documentsContract.remove(documentCID), which clears the CID mapping. Immediately afterward, the function attempts to retrieve the documentId via getByCID(documentCID).id, which returns zero since the mapping no longer exists. As a result, the emitted event contains an incorrect documentId, reflecting a flawed operation sequence that leads to inaccurate event logging and potential misinterpretation in off-chain systems.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L231

Impacts
Emits incorrect event data, which may mislead off-chain consumers and monitoring system

Remediation
To fix this issue, it is advisable to retrieve and store the documentId before calling the remove function, ensuring accurate event emission and consistent data handling.

Retest
This issue has been fixed by reordering the valuable assignment before deleting the data.

Bug ID #L005 [Fixed]

Role Entry Not Removed from Roles List

Vulnerability Type
Business Logic (SC03-LogicErrors)

Severity
Low

Description
The _removeRoleFromList function only removes the specified account's index from the _roleIndices mapping for the given role but fails to delete the corresponding entry in the _rolesList array. This results in stale entries persisting in _rolesList, meaning that a removed role is still visible in role listings even though the account no longer possesses the role. This leads to inconsistencies between role state in _callers and data shown through view functions like listAllRoles, potentially misleading administrators or users relying on role listings.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Control.sol#L94-L111

Impacts
May cause outdated role data to persist, resulting in incorrect role listings and misleading administrative output.

Remediation
To mitigate this issue, it is recommended to also clear or nullify the corresponding Role entry in _rolesList when removing the role, ensuring consistency across all role-related data structures.

Retest
This issue has been fixed by removing the role entry from _rolesList mappings.

Bug ID #L006 [Won’t Fix]

Function Contains Unused Transfer Logic with No External Effects

Vulnerability Type
Code With No Effects (SCWE-062)

Severity
Low

Description
The transfer function updates internal balances and transaction logs but is never invoked anywhere in the contract. Functions and parameters that are not used or externally exposed should be removed prior to mainnet deployment to maintain code clarity and reduce audit surface.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Credit.sol#L57-L71

Impacts
This does not affect the contract’s security but may cause confusion for developers or auditors reviewing the code. It also increases contract size unnecessarily, which can affect gas efficiency.

Remediation
To fix this issue, it is recommended that if transfer is not intended for use, consider removing it from the codebase.

Retest

Client’s comment: Noted, it is there in case a manual credit transfer is needed to be done by the operator for one account to another.

Bug ID #L007 [Fixed]

Missing Events in Important Functions

Vulnerability Type
Missing Best Practices (SCWE-063)

Severity
Low

Description
Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain. These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.

The contract was found to be missing these events on certain critical functions which would make it difficult or impossible to track these transactions off-chain.

Affected Code
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultQ.sol#L255
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultQ.sol#L260
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultQ.sol#L265
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultQ.sol#L270
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L322
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L327

Impacts
Events are used to track the transactions off-chain and missing these events on critical functions makes it difficult to audit these logs if they're needed at a later stage.

Remediation
Consider emitting events for important functions to keep track of them.

Retest
This issue has been fixed by emitting an event in the state-changing functions.

Bug ID #L008 [Fixed]

Outdated Pragma

Vulnerability Type
Outdated Compiler Version (SCWE-061)

Severity
Low

Description
The smart contract is using an outdated version of the Solidity compiler specified by the pragma directive i.e. 0.8.26. Solidity is actively developed, and new versions frequently include important security patches, bug fixes, and performance improvements. Using an outdated version exposes the contract to known vulnerabilities that have been addressed in later releases. Additionally, newer versions of Solidity often introduce new language features and optimizations that improve the overall security and efficiency of smart contracts.

Affected Code
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/BasicTokenTemplate.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/UserTemplate.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Users.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Control.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Orders.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Credit.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Assets.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/templates/DividendTokenTemplate.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultQ.sol#L2
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L2

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Documents.sol#L2

Impacts
The use of an outdated Solidity compiler version can have significant negative impacts. Security vulnerabilities that have been identified and patched in newer versions remain exploitable in the deployed contract.
Furthermore, missing out on performance improvements and new language features can result in inefficient code execution and higher gas costs.

Remediation
It is suggested to use the 0.8.29 pragma version.
Reference: https://scs.owasp.org/SCWE/SCSVS-CODE/SCWE-061/

Retest
This issue has been addressed by updating the pragma version to 0.8.29.

Bug ID #L009 [Fixed]

Missing Zero Address Validations

Vulnerability Type
Missing Input Validation (SC04-Lack Of Input Validation)

Severity
Low

Description:
The contracts were found to be setting new addresses without proper validations for zero addresses.
Address type parameters should include a zero-address check otherwise contract functionality may become inaccessible or tokens burned forever.
Depending on the logic of the contract, this could prove fatal and the users or the contracts could lose their funds, or the ownership of the contract could be lost forever.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Control.sol#L61
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L74
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L87
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L92
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L97
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L102
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L114
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L172
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L184
- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L235

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/TarmiizVaultE.sol#L243

Impacts
If address type parameters do not include a zero-address check, contract functionality may become unavailable or tokens may be burned permanently.

Remediation
Add a zero address validation to all the functions where addresses are being set.

Retest
This issue has been fixed by implementing the zero address validation.

Bug ID #G001 [Fixed]

Missing Break Statement in detach Loop

Vulnerability Type
Gas Optimization

Severity
Gas

Description
The detach function iterates through the _ownerDocuments array to find and remove the specified documentId. However, after successfully removing the document, the loop continues iterating through the rest of the array unnecessarily. This results in wasted gas, especially for owners with many documents. The loop should terminate immediately after the target document is found and removed, but it currently lacks a break or return statement to achieve this.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Documents.sol#L71-L81

Impacts
Leads to unnecessary gas consumption during document detachment due to continued iteration after removal.

Remediation
To fix this issue, it is recommended to insert a break or return statement immediately after the document is removed to exit the loop and prevent redundant iterations.

Retest
This issue has been fixed by breaking the loop once the document is removed.

Bug ID #G002 [Fixed]

Inefficient Array Element Removal in Role Index

Vulnerability Type
Gas Optimization

Severity
Gas

Description
The _removeRoleFromList function removes an account's index from the _roleIndices array by shifting elements one by one to fill the gap and then popping the last element. This element-shifting approach is inefficient and results in unnecessary gas consumption, especially when the array is large. A more gas-efficient method, commonly known as the swap-and-pop technique, swaps the element to be removed with the last element and then pops it, thereby reducing the number of writes and saving gas.

Affected Code

- https://github.com/Tarmiiz/Vault-Contracts/blob/4dac763f7291f64fbff28d729cceefc53e07c770/contracts/utils/Control.sol#L104-L110

Impacts
Wastes more gas because the array is handled in an inefficient way when removing a role.

Remediation
If order does not matter: swap-and-pop is preferable for gas efficiency.
If order matters current shifting logic should be kept, and the gas cost accepted.

Retest
This issue has been fixed by updating the data removal to the swap-and-pop method.

6. The Disclosure ----------------------

The Reports provided by CredShields are not an endorsement or condemnation of any specific project or team and do not guarantee the security of any specific project. The contents of this report are not intended to be used to make decisions about buying or selling tokens, products, services, or any other assets and should not be interpreted as such.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. CredShields does not provide any warranty or representation about the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business. The report is not intended to be used as investment advice and should not be relied upon as such.

CredShields Audit team is not responsible for any decisions or actions taken by any third party based on the report.

YOUR SECURE FUTURE STARTS HERE

At CredShields, we're more than just auditors. We're your strategic partner in ensuring a secure Web3 future. Our commitment to your success extends beyond the report, offering ongoing support and guidance to protect your digital assets

Audited by