



Security Assessment

Credbull

CertiK Assessed on May 13th, 2024





CertiK Assessed on May 13th, 2024

Credbull

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES DeFi	ECOSYSTEM EVM Compatible	METHODS Formal Verification, Manual Review, Static Analysis
LANGUAGE Solidity	TIMELINE Delivered on 05/13/2024	KEY COMPONENTS N/A
CODEBASE https://github.com/credbull/credbull-defi/tree/main/packages/contracts/src View All in Codebase Page	COMMITTS <ul style="list-style-type: none"> da9877c0823663a5fca5f75f3a09b84e040dab8e 9b86f2af463028173079c7180e231c6f14325c56 View All in Codebase Page	

Highlighted Centralization Risks

⚠ Withdraws can be disabled

Vulnerability Summary



0 Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2 Major	2 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2 Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
3 Minor	1 Resolved, 2 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
1 Informational	1 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | CREDBULL

I Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

I Findings

CBV-02 : Centralization Related Risks in `CUSTODIAN` Address

GLOBAL-01 : Centralization Related Risks

SRC-02 : Potential Asset Lock in `FixedYieldVault` and `UpsideVault` Contracts

UVB-01 : Inconsistent Use of `Pausable` Contract in `UpsideVault`

CBV-01 : Tokens Transferred to Vault Will Be Locked

SRC-01 : Incompatibility With Deflationary Tokens (Non-standard ERC20 Token)

SRC-03 : Third-Party Dependencies

UVB-03 : Shadowing State Variable

I Formal Verification

Considered Functions And Scope

Verification Results

I Appendix

I Disclaimer

CODEBASE | CREDBULL

Repository

<https://github.com/credbull/credbull-defi/tree/main/packages/contracts/src>







Commit



- [da9877c0823663a5fca5f75f3a09b84e040dab8e](#)
- [9b86f2af463028173079c7180e231c6f14325c56](#)

AUDIT SCOPE | CREDBULL

30 files audited ● 2 files with Acknowledged findings ● 28 files without findings

ID	Repo	File	SHA256 Checksum
● CBV	credbull/credbull-defi	 base/CredbullBaseVault.sol	746d77413ed9e406ea61e32cde5ae555ff23c112dfd40dc9b73fd6f7f761b7a5
● UVB	credbull/credbull-defi	 vaults/UpSideVault.sol	24b73c346770cd0b044ee45c0c840172517572261d086e73cd474608569c4d44
● CKY	credbull/credbull-defi	 CredbullKYCProvider.sol	1b9228c7f2e6b2289ae5b18cd64373796dc4adf3910bb1ce838f8680ba6329b4
● CFY	credbull/credbull-defi	 CredbullFixedYieldVaultWithUpside.sol	b020e54eb3fb146c8e9d997177a1197b864177693707b66c04dbd567a11d620a
● CFV	credbull/credbull-defi	 CredbullFixedYieldVault.sol	07f7969bf5a99cc944bf92506659663634e7f7a1e9964ada22bd9a395c0b8140
● MVB	credbull/credbull-defi	 extensions/MaturityVault.sol	24619b65444c6bc19e69350145681b8e71b0341b52aaff45fac8d032c2adb864
● CFF	credbull/credbull-defi	 factories/CredbullFixedYieldVaultFactory.sol	7b8658de2db4e7c17dd01bf149ad7b25ed71a098037fc4e68b2a33bb195330a6
● CUV	credbull/credbull-defi	 factories/CredbullUpsideVaultFactory.sol	6eb9b98e837fb1f2d7c61d34ab354e7c401358f860003286531b31c2163e581c
● CVF	credbull/credbull-defi	 factories/CredbullVaultFactory.sol	be881491aa86d2d203146ac91dbd92539a09851b94086d564267fe55dbee729
● MCP	credbull/credbull-defi	 plugins/MaxCapPlug.sol	6a088c4f87a285941f97d1267b848fb0c5cd25f89f9fa2a1b4341514430b7035
● WPI	credbull/credbull-defi	 plugins/WhitelistPlugIn.sol	c3e4faaa4bb6ef3c05abdb6480ffad6bfc04e4d9b545bf3213cc2f53b4aaa370
● WIN	credbull/credbull-defi	 plugins/WindowPlugIn.sol	c79b09843e9b2038710f7a33fa56b8810533d26a9262a597441ed1d05c7fd775
● FYV	credbull/credbull-defi	 vaults/FixedYieldVault.sol	c7aa8297b9288698541a92a95197cecfef88ddcc7dbee8d11cf6d3268c73db7

ID	Repo	File	SHA256 Checksum
● ICB	credbull/credbull-defi	 interface/ICredbull.sol	ce476844f5b7b27820becf49288f263d8303468a3710b166dc746880dfab0fe1
● IKY	credbull/credbull-defi	 interface/IKYCProvider.sol	6bdbdec57340f10c318c08f47c9ca20b77290268de80ecc6e9cc9aaf71751bac
● CRE	credbull/credbull-defi	 base/CredbullBaseVault.sol	95d97f2861995ce59308b87f0a5e8b300197c64f660e998bc1d0f1f73dba8182
● MVU	credbull/credbull-defi	 extensions/MaturityVault.sol	24619b65444c6bc19e69350145681b8e71b0341b52aaff45fac8d032c2adb864
● CYF	credbull/credbull-defi	 factories/CredbullFixedYieldVaultFactory.sol	7b8658de2db4e7c17dd01bf149ad7b25ed71a098037fc4e68b2a33bb195330a6
● CUF	credbull/credbull-defi	 factories/CredbullUpsideVaultFactory.sol	6eb9b98e837fb1f2d7c61d34ab354e7c401358f860003286531b31c2163e581c
● CRD	credbull/credbull-defi	 factories/CredbullVaultFactory.sol	be881491aa86d2d203146ac91dbd92539a09851b94086d564267fe55dbee729
● ICU	credbull/credbull-defi	 interface/ICredbull.sol	ce476844f5b7b27820becf49288f263d8303468a3710b166dc746880dfab0fe1
● IKC	credbull/credbull-defi	 interface/IKYCProvider.sol	6bdbdec57340f10c318c08f47c9ca20b77290268de80ecc6e9cc9aaf71751bac
● MAX	credbull/credbull-defi	 plugins/MaxCapPlug.sol	6a088c4f87a285941f97d1267b848fb0c5cd25f89f9fa2a1b4341514430b7035
● WHI	credbull/credbull-defi	 plugins/WhitelistPlugIn.sol	c3e4faaa4bb6ef3c05abdb6480ffad6bfc04e4d9b545bf3213cc2f53b4aaa370
● WID	credbull/credbull-defi	 plugins/WindowPlugIn.sol	c79b09843e9b2038710f7a33fa56b8810533d26a9262a597441ed1d05c7fd775
● FIX	credbull/credbull-defi	 vaults/FixedYieldVault.sol	57b1a685c6050ac8ac95b0f2401145afa765ab8e9d243dbfead354d9d4c5bbfc
● UVU	credbull/credbull-defi	 vaults/UpsideVault.sol	3c3f66d642d7c9837285edb18400bf0581cb9ae019df7edd8b292515317cb58e
● CYV	credbull/credbull-defi	 CredbullFixedYieldVault.sol	07f7969bf5a99cc944bf92506659663634e7f7a1e9964ada22bd9a395c0b8140

ID	Repo	File	SHA256 Checksum
● CFW	credbull/credbull-defi	 CredbullFixedYieldVaultWithUpside.sol	b020e54eb3fb146c8e9d997177a1197b864177693707b66c04dbd567a11d620a
● CKC	credbull/credbull-defi	 CredbullKYCProvider.sol	1b9228c7f2e6b2289ae5b18cd64373796dc4adf3910bb1ce838f8680ba6329b4

APPROACH & METHODS | CREDBULL

This report has been prepared for Credbull to discover issues and vulnerabilities in the source code of the Credbull project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

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 specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. 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 perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance 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.

FINDINGS | CREDBULL



8

Total Findings

0

Critical

2

Major

2

Medium

3

Minor

1

Informational

This report has been prepared to discover issues and vulnerabilities for Credbull. Through this audit, we have uncovered 8 issues ranging from different severity levels. Utilizing the techniques of Static Analysis & Manual Review to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
CBV-02	Centralization Related Risks In <code>CUSTODIAN</code> Address	Centralization	Major	● Acknowledged
GLOBAL-01	Centralization Related Risks	Centralization	Major	● Acknowledged
SRC-02	Potential Asset Lock In <code>FixedYieldVault</code> And <code>UpsideVault</code> Contracts	Logical Issue	Medium	● Resolved
UVB-01	Inconsistent Use Of <code>Pausable</code> Contract In <code>UpsideVault</code>	Access Control	Medium	● Resolved
CBV-01	Tokens Transferred To Vault Will Be Locked	Design Issue	Minor	● Resolved
SRC-01	Incompatibility With Deflationary Tokens (Non-Standard ERC20 Token)	Volatile Code	Minor	● Acknowledged
SRC-03	Third-Party Dependencies	Volatile Code	Minor	● Acknowledged
UVB-03	Shadowing State Variable	Coding Style	Informational	● Resolved

CBV-02 | CENTRALIZATION RELATED RISKS IN CUSTODIAN ADDRESS

Category	Severity	Location	Status
Centralization	● Major	base/CredbullBaseVault.sol (da9877): 82	● Acknowledged

Description

The `CredbullBaseVault` contract utilizes a designated `CUSTODIAN` address to receive tokens from depositors. This setup centralizes the control of deposited assets to a single account. If this custodian account is compromised, malicious actors could misuse the assets, potentially resulting in the contract not having sufficient funds to fulfill withdrawal requests by legitimate depositors.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (2/3, 3/5) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
OR
- Remove the risky functionality.

I Alleviation

[Credbull Team, 05/13/2024]:

This is implemented as designed. The CUSTODIAN address will be an account with our Virtual Asset Service Provider (e.g. Circle or Coinbase Institutional).

We will use Multi-Sig and/or MPC to ensure this and all sensitive accounts are protected.

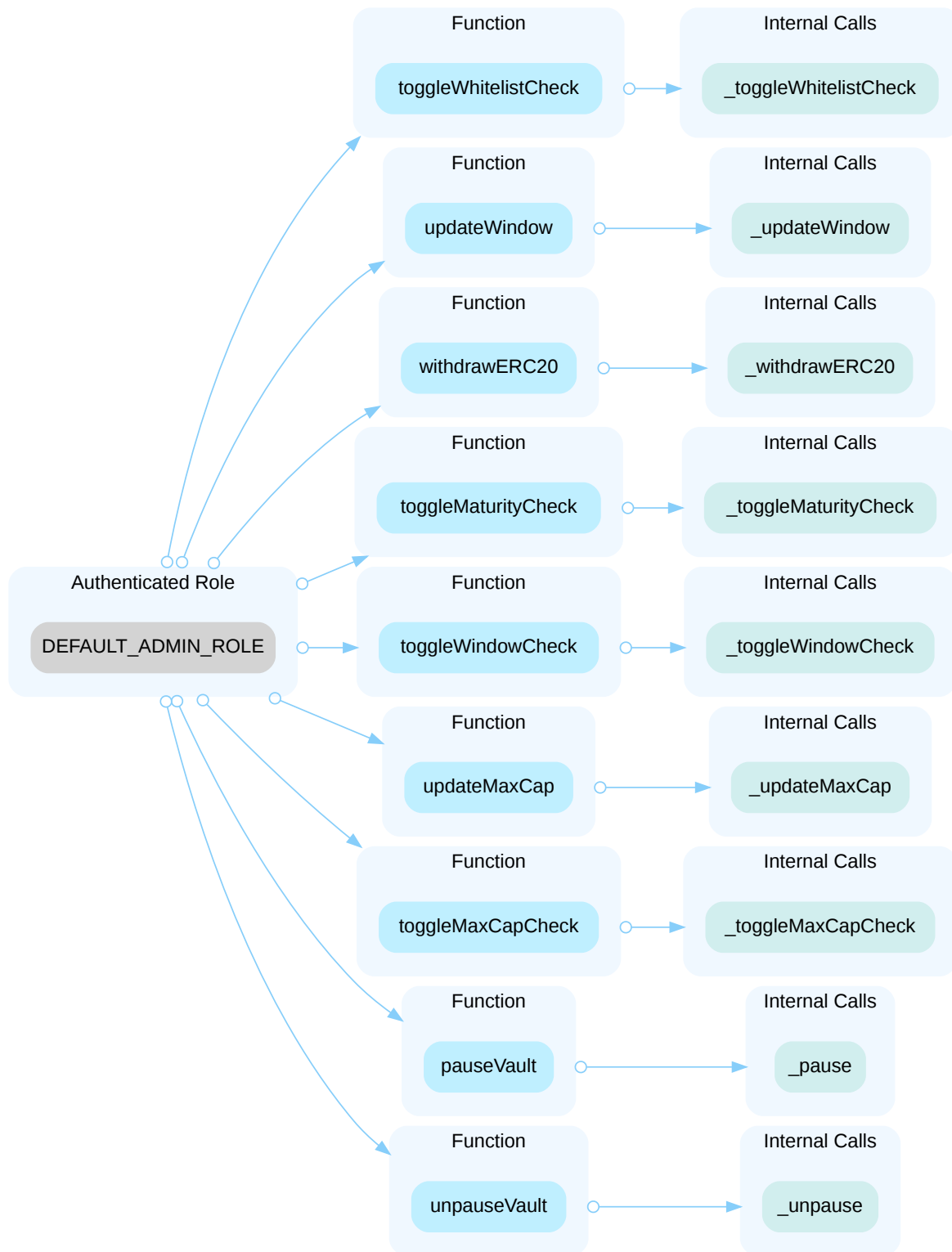
[CertiK, 05/13/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● Major		● Acknowledged

Description

In the contract `FixedYieldVault` the role `DEFAULT_ADMIN_ROLE` has authority over the functions shown in the diagram below. Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow the hacker to take advantage of this authority and enable/disable security checks, pause/unpause deposits and withdrawals, and update critical variables. Moreover, the account can withdraw tokens from the contract at any time, it could lead to insufficient tokens available for depositors when attempting to withdraw.



In the contract `FixedYieldVault` the role `OPERATOR_ROLE` has authority over the following function:

- `mature()`

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and permit withdrawals.

In the contract `UpsideVault` the role `OPERATOR_ROLE` has authority over the following function:

- `setTWAP()`

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and set `twap`.

In the contract `AccessControl` the role `adminRole` has authority over the following functions:

- `grantRole()`
- `revokeRole()`

Any compromise to the `adminRole` account may allow the hacker to take advantage of this authority and grant associated role to any account or revoke the role from any account. Note that `DEFAULT_ADMIN_ROLE` is the admin role for all roles.

In the contract `AccessControl` the role `role` has authority over the following function:

- `renounceRole()`

Any compromise to the `role` account may allow the hacker to take advantage of this authority and renounce corresponding privileges to functions within other contracts.

In the contract `CredbullKYCProvider` the role `_owner` has authority over the following function:

- `updateStatus()`

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and add/remove addresses from the whitelist.

In the contract `Ownable` the role `_owner` has authority over the following functions:

- `transferOwnership()`
- `renounceOwnership()`

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and transfer or renounce the ownership.

In the contract `CredbullFixedYieldVaultFactory` the role `OPERATOR_ROLE` has authority over the following function:

- `createVault()`

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and deploy a new `CredbullFixedYieldVault` contract.

In the contract `CredbullUpsideVaultFactory` the role `OPERATOR_ROLE` has authority over the following function:

- `createVault()`

Any compromise to the `OPERATOR_ROLE` account may allow a hacker to take advantage of this authority and deploy a new `CredbullFixedYieldVaultWithUpside` contract.

In the contract `CredbullVaultFactory` the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- `allowCustodian()`
- `removeCustodian()`

Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow a hacker to take advantage of this authority and add/remove an account from `allowedCustodians`.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (2/3, 3/5) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
- AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
- OR
- Remove the risky functionality.

I Alleviation

[Credbull Team, 05/13/2024]:

To protect sensitive accounts, Credbull uses wallets with advanced security features such as Multi-Sig and MPC.

We will also add a Time-lock with reasonable latency for privileged operations.

Once we have deployed to Mainnet, Credbull will share relevant information for transparency.

[CertiK, 05/13/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

SRC-02 | POTENTIAL ASSET LOCK IN `FixedYieldVault` AND `UpsideVault` CONTRACTS

Category	Severity	Location	Status
Logical Issue	● Medium	base/CredbullBaseVault.sol (da9877): 103; vaults/UpsideVault.sol (da9877): 92	● Resolved

Description

The `FixedYieldVault` and `UpsideVault` contracts use the `totalAssetDeposited` variable to track the total assets deposited into the vault. This variable is adjusted upward during deposits and downward during withdrawals. However, if assets are directly transferred into the vault without using the `_deposit()` function (e.g., through donations or direct transfers), these assets are not recorded in `totalAssetDeposited`. they will be locked in the contract indefinitely.

Although this issue is somewhat mitigated since both contracts inherit from the `MaturityVault` contract, which includes a `_mature()` function that adjusts `totalAssetDeposited` to match the current balance, this adjustment is not guaranteed to occur before withdrawals are permitted. Moreover, tokens transferred after `_mature()` has been called would still be locked.

Recommendation

It is recommended to prevent tokens from being locked in the contract.

Alleviation

[Credbull Team, 05/13/2024]: The team heeded the advice and resolved the issue in commit: [9b86f2af463028173079c7180e231c6f14325c56](https://github.com/credbull/9b86f2af463028173079c7180e231c6f14325c56).

UVB-01 | INCONSISTENT USE OF Pausable CONTRACT IN UpsideVault

Category	Severity	Location	Status
Access Control	● Medium	vaults/UpsideVault.sol (da9877): 52, 80	● Resolved

Description

The `UpsideVault` contract inherits from the `Pausable` contract, which provides functionality to pause and unpaue contract operations. However, the `UpsideVault` does not utilize the `whenNotPaused` modifier from the `Pausable` contract in its `_deposit()` and `_withdraw()` functions. This oversight nullifies the benefits of having a pausable mechanism.

Recommendation

It is recommended to apply `whenNotPaused` modifier in the `UpsideVault`'s `_deposit()` and `_withdraw()` functions.

Alleviation

[Credbull Team, 05/07/2024]: The team heeded the advice and resolved the issue in commit:

9b86f2af463028173079c7180e231c6f14325c56.

CBV-01 | TOKENS TRANSFERRED TO VAULT WILL BE LOCKED

Category	Severity	Location	Status
Design Issue	● Minor	base/CredbullBaseVault.sol (da9877): 132, 140	● Resolved

Description

The comments of the `transfer()` and `transferFrom()` functions in the `CredbullBaseVault` contract indicate that tokens sent to the contract should allow for asset redemption. However, there appears to be no mechanism in the provided code to handle the tokens once they reach the contract. This could lead to tokens being locked within the contract with no clear method for users to retrieve the corresponding assets.

```
    /// @notice The share token should be soul bound. Should be transferable only to
    vault to receive assets back
    function transfer(address to, uint256 value) public override(ERC20, IERC20)
    returns (bool) {
        if (to != address(this)) revert CredbullVault__TransferOutsideEcosystem();
        address owner = _msgSender();
        _transfer(owner, to, value);
        return true;
    }

    /// @notice The share token should be soul bound. Should be transferable only to
    vault to receive assets back
    function transferFrom(address from, address to, uint256 value) public
    override(ERC20, IERC20) returns (bool) {
        if (to != address(this)) revert CredbullVault__TransferOutsideEcosystem();
        address spender = _msgSender();
        _spendAllowance(from, spender, value);
        _transfer(from, to, value);
        return true;
    }
```

Recommendation

It is recommended to revert any call to `transfer()` or `transferFrom()`.

Alleviation

[Credbull Team, 05/07/2024]: The team heeded the advice and resolved the issue in commit:

9b86f2af463028173079c7180e231c6f14325c56.

SRC-01 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS (NON-STANDARD ERC20 TOKEN)

Category	Severity	Location	Status
Volatile Code	● Minor	base/CredbullBaseVault.sol (da9877): 82, 106; vaults/UpsideVault.sol (da9877): 71, 72, 94, 97	● Acknowledged

Description

The vault contracts are not equipped to handle non-standard ERC20 tokens, including deflationary or rebase tokens, which may apply transaction fees or adjust balances dynamically. The contract's reliance on `transferFrom()` and `transfer()` methods without validating the actual transferred amounts could result in discrepancies between expected and actual token balances. This issue is particularly critical because it could lead to insufficient tokens available for depositors when attempting to withdraw, potentially disrupting contract functionality and user transactions.

Scenario

When transferring deflationary ERC20 tokens, the input amount may not equal the received amount due to the charged transaction fee. For example, if a user sends 100 deflationary tokens (with a 10% transaction fee), only 90 tokens actually arrive to the contract. However, a failure to discount such fees may allow the same user to withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

Proof of Concept

Foundry test:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.7.5;
pragma abicoder v2;

import "forge-std/Test.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract DeflationaryToken is ERC20 {
    address deadAddress = 0x0000000000000000000000000000000000000000000000000000000000000000dEaD;
    constructor () ERC20 ("Test", "TEST") {
        _mint(msg.sender, 1000000);
    }

    function _transfer(address sender, address recipient, uint256 amount) internal
    override {
        uint256 burnAmount = 10 * amount / 100;
        uint256 transferAmount = amount - burnAmount;
        super._transfer(sender, deadAddress, burnAmount);
        super._transfer(sender, recipient, transferAmount);
    }
}

contract VictimContract {
    address public token;
    mapping(address => uint256) public stakedAmount;

    constructor (address _token) {
        token = _token;
    }

    function stake(uint256 amount) public {
        ERC20(token).transferFrom(msg.sender, address(this), amount);
        stakedAmount[msg.sender] += amount;
    }

    function unstake() public {
        uint256 amount = stakedAmount[msg.sender];
        ERC20(token).transfer(msg.sender, amount);
        stakedAmount[msg.sender] = 0;
    }
}

contract DeflationaryTokenTest is Test {
    DeflationaryToken public deflationaryToken;
    VictimContract public victimContract;
    address public user = vm.addr(1);

    function setUp() public {
        deflationaryToken = new DeflationaryToken();
    }
}
```

```
victimContract = new VictimContract(address(deflationaryToken));
deflationaryToken.transfer(address(victimContract), 10000);
deflationaryToken.transfer(user, 10000);
}

function testIssue() public {
    vm.startPrank(user);
    deflationaryToken.approve(address(victimContract), 1000);

    uint256 victimBalanceBefore =
deflationaryToken.balanceOf(address(victimContract)); // balance before staking
    victimContract.stake(1000);
    victimContract.unstake();

    uint256 victimBalanceAfter =
deflationaryToken.balanceOf(address(victimContract)); // balance after unstaking

    vm.stopPrank();

    require(victimBalanceAfter < victimBalanceBefore, "error"); // victim
contract lost tokens
}
}
```

Recommendation

We advise the client to regulate the set of tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support non-standard ERC20 tokens.

Alleviation

[Credbull Team, 05/07/2024]: Acknowledged, our Vaults will be using well-known ERC-20 stablecoins only, e.g. USDC and USDT.

SRC-03 | THIRD-PARTY DEPENDENCIES

Category	Severity	Location	Status
Volatile Code	● Minor	base/CredbullBaseVault.sol (da9877): 55; vaults/UpsideVault.sol (da9877): 16	● Acknowledged

Description

The contract is serving as the underlying entity to interact with third-parties asset token and collateral token. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

Recommendation

We recommend that the project team constantly monitor the functionality of these contracts to mitigate any side effects that may occur when unexpected changes are introduced.

Alleviation

[Credbull Team, 05/07/2024]:

Understood and agreed regarding third-party dependencies overall. For more context:

- Asset Token will always be a USDC or USDT stablecoin.
- Collateral Token will only be our own Credbull (\$CBL) ERC20 token. Credbull will develop this contract from OpenZeppelin standards.

UVB-03 | SHADOWING STATE VARIABLE

Category	Severity	Location	Status
Coding Style	● Informational	vaults/UpsideVault.sol (da9877): 23	● Resolved

Description

The state variable `_balances` in the `Upsidevault` contract is shadowing the same named component in the parent contract `ERC20`. This means that when the derived contract accesses the state variable by its name, it will use the one defined in the derived contract, not the one in the parent contract. This can lead to confusion.

Recommendation

It is suggested to rename the state variable that shadows another definition.

Alleviation

[Credbull Team, 05/07/2024]: The team heeded the advice and resolved the issue in commit: 9b86f2af463028173079c7180e231c6f14325c56.

FORMAL VERIFICATION | CREDBULL

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-renounce-ownership-is-permanent	Once Renounced, Ownership Cannot be Regained
ownable-transferownership-correct	Ownership is Transferred.
ownable-renounceownership-correct	Ownership is Removed.
ownable-owner-succeed-normal	<code>owner</code> Always Succeeds

Verification Results

For the following contracts, formal verification established that each of the properties that were in scope of this audit (see scope) are valid:

**Detailed Results For Contract CredbullKYCProvider
(packages/contracts/src/CredbullKYCProvider.sol) In Commit
da9877c0823663a5fca5f75f3a09b84e040dab8e**

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounce-ownership-is-permanent	● True	
ownable-renounceownership-correct	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

APPENDIX | CREDBULL

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Access Control	Access Control findings are about security vulnerabilities that make protected assets unsafe.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed Ownable Properties

Properties related to function `renounceOwnership`

ownable-renounce-ownership-is-permanent

The contract must prohibit regaining of ownership once it has been renounced.

Specification:

```
constraint \old(owner()) == address(0) ==> owner() == address(0);
```

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to `address(0)`.

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

Properties related to function `owner`**ownable-owner-succeed-normal**

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

