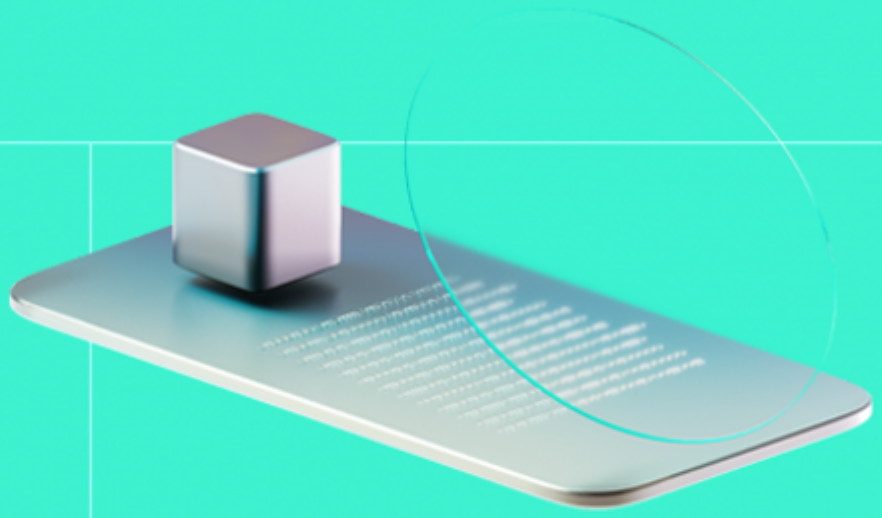
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Credbull

Date: 15/08/2024

We express our gratitude to the Credbull team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Credbull is an ERC-4626 vault system.

Document

Name	Smart Contract Code Review and Security Analysis Report for Credbull
Audited By	Olesia Bilenka, Andy Cho
Approved By	Ataberk Yavuzer
Website	https://credbull.io/
Changelog	26/07/2024 - Preliminary Report
	15/08/2024 - Final Report
Platform	Arbitrum
Language	Solidity
Tags	Vault; Yield Farming
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/credbull/credbull-defi/
Commit	f927c85
Remediation commit	ce1c17e

Audit Summary

The system users should acknowledge all the risks summed up in the risks section of the report

2	1	0	1
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	1
Low	1

Vulnerability	Severity
F-2024-4404 - Centralization Risk in Vault System Leading to Deposit Lock	Medium
F-2024-4396 - Risk of Ownership Control Loss in Owner-Dependent Contracts	Low

Documentation quality

- Functional requirements are provided.
- Technical description is provided.
- NatSpec comments are provided.

Code quality

- The code mostly follows best practices and styleguides.
- The development environment is configured.

Test coverage

Code coverage of the project is **90%** (branch coverage).

- Deployment and basic user interactions are covered with tests.

Table of Contents

System Overview **6**

Privileged Roles 7

Risks **9**

Findings **10**

Vulnerability Details 10

Observation Details 14

Disclaimers 24

Appendix 1. Severity Definitions **25**

Appendix 2. Scope **26**

System Overview

Credbull is an ERC-4626 vault system with the following contracts:

Vault - is an abstract smart contract based on OpenZeppelin's ERC4626, designed to manage ERC20 token deposits with enhanced security. Deposited assets are transferred to a designated CUSTODIAN address, with the contract maintaining records of total assets deposited. The contract includes functions for depositing, minting, withdrawing, and redeeming assets, with built-in error handling for invalid operations. Additionally, it features access control mechanisms and the ability to pause operations for security purposes.

MaturityVault - is an abstract contract extends the `Vault` by adding a maturity feature, preventing further deposits once matured. It tracks whether the vault is matured and includes functions to mature the vault and check maturity status. This ensures that only existing assets and any accrued yield are managed after maturity.

FixedYieldVault - is a contract that extends the `MaturityVault` and integrates additional features from `WhiteListPlugin`, `WindowPlugin`, and `MaxCapPlugin`. It allows deposits and redemptions based on set windows and caps, and it offers a fixed yield to users. The contract includes mechanisms for maturity checking and whitelisting, ensuring that only approved addresses can interact with it. The `FixedYieldVault` also has functionality to pause operations and to withdraw any ERC20 tokens held by the contract.

UpsideVault - is a contract that extends the functionality of the `FixedYieldVault` by integrating collateral management using the Credbull token (CBL). Users must deposit collateral in CBL tokens alongside their main assets, with the required collateral amount determined by a specified percentage of the asset value and the current TWAP (Time-Weighted Average Price). The contract allows for both deposits and withdrawals while ensuring the proper handling and accounting of the collateral. Additionally, the contract has a mechanism to update the TWAP value, which is crucial for calculating the collateral requirements.

VaultFactory - is a factory designed to manage the creation and oversight of vault contracts. It maintains sets of allowed custodians and created vaults, allowing only authorized custodians to interact with the factory. The contract allows administrators to add or remove custodians and provides functions to query the list of created vaults and validate custodians.

WhiteListProvider - is a contract that manages a whitelist of addresses that can be used by other contracts to determine if addresses are authorized. It allows the owner to update the whitelist status of multiple addresses at once. The contract tracks whitelist status through a mapping and provides a function to check if an address is whitelisted.

MaxCapPlugin - is a contract that provides functionality to enforce a maximum cap on asset deposits in a vault. It tracks the maximum allowed cap through the `maxCap` variable and uses a boolean flag, `checkMaxCap`, to enable or disable the cap enforcement. The contract provides internal functions to check if the deposited amount exceeds the cap, update the cap value, and toggle the cap check status.

WhiteListPlugin - is a contract that manages white-listing functionality within a vault system. It enforces that certain addresses must be white-listed to perform operations, such as

deposits, if their transaction amount meets or exceeds a specified threshold (`depositThresholdForWhiteListing`). The white-list status is determined through an external `IWhiteListProvider` contract. The plugin includes internal functions for checking white-list status and toggling the white-list check.

WindowPlugin - is a contract that contract manages deposit and redemption time windows for a vault. It defines time periods during which deposits and redemptions are permitted, using timestamps for opening and closing these windows. The plugin includes functions to check whether operations fall within these time windows, update window timestamps, and toggle the window check functionality.

CredbullFixedYieldVault - is a straightforward implementation of the `FixedYieldVault` contract. It serves as a specific instance of `FixedYieldVault` with no additional functionality or modifications.

CredbullFixedYieldVaultFactory - is a contract that extends the `VaultFactory` and is designed to facilitate the creation of `CredbullFixedYieldVault` instances. It provides functionality for deploying new vaults and managing the list of allowed custodians and vault addresses.

CredbullFixedYieldVaultWithUpside - is a contract that extends the `UpsideVault` and is tailored to integrate additional upside functionality into the fixed yield vault. It leverages the existing features of `UpsideVault`, which itself extends `FixedYieldVault`, to provide enhanced capabilities for handling collateral and deposit-redemption windows.

CredbullUpsideVaultFactory - is a contract that extends `VaultFactory` to create instances of `CredbullFixedYieldVaultWithUpside`. This factory contract provides functionality for deploying new vaults that integrate both fixed yield and upside features.

CredbullWhiteListProvider - is an implementation that extends the `WhiteListProvider` contract. It inherits the functionality for managing a whitelist of addresses from `WhiteListProvider`, including updating whitelist statuses and querying the whitelist status of an address.

Privileged roles

- The `DEFAULT_ADMIN_ROLE` in the `Vault` contract has the authority to pause and unpause contract operations, impacting the ability of users to deposit or withdraw assets. They can also set the custodian address and modify parameters related to asset management, such as asset decimals and share token settings.
- The `DEFAULT_ADMIN_ROLE` in the `MaturityVault` contract has the authority to toggle the maturity check, initiate the vault's maturation process, and manage other critical parameters, affecting the vault's operational status and asset management.
- The `DEFAULT_ADMIN_ROLE` in the `FixedYieldVault` contract has significant control, including the ability to toggle checks for maturity, whitelisting, window periods, and max cap, as well as to update the max cap value and window timestamps. The `OPERATOR_ROLE` can initiate the vault's maturation process and setting the TWAP value (`UpsideVault` contract). These roles hold significant power over the contract's functionality and the management of user assets.

- The `DEFAULT_ADMIN_ROLE` in the `VaultFactory` contract has broad control, including adding or removing custodian addresses and managing factory-wide settings. The `OPERATOR_ROLE` does not have direct access to this contract's functions but is important in related contexts where vaults are created and managed.
- The owner is of the `WhiteListProvider` contract is authorized to update the whitelist status of addresses and make changes to the whitelist.
- `OPERATOR_ROLE` role can create new `CredbullFixedYieldVault` instances. It must be a valid custodian to create a vault

Risks

- **Owner's Unrestricted State Modification:** The absence of restrictions on state variable modifications by the owner leads to arbitrary changes, affecting contract integrity and user trust, especially during critical operations like minting phases.
- **Absence of Time-lock Mechanisms for Critical Operations**: Without time-locks on critical operations, there is no buffer to review or revert potentially harmful actions, increasing the risk of rapid exploitation and irreversible changes.
- **Single Points of Failure and Control:** The project is fully or partially centralized, introducing single points of failure and control. This centralization can lead to vulnerabilities in decision-making and operational processes, making the system more susceptible to targeted attacks or manipulation.

Findings

Vulnerability Details

F-2024-4404 - Centralization Risk in Vault System Leading to Deposit Lock - Medium

Description:

The Vault system, inheriting ERC4626 functionality, presents a centralization risk. Upon depositing tokens, the funds are sent to the `CUSTODIAN` address instead of being stored in the vault itself.

```
function _deposit(...) ... {
    ...

    SafeERC20.safeTransferFrom(IERC20(asset()), caller, CUSTODIAN, assets);
    ...

}
```

This setup risks users being unable to withdraw their tokens if the `CUSTODIAN` fails to return them to the vault address. The Credbull team indicated that this design is for security reasons, but it inherently creates a risk for users.

> In general for security reasons we don't want the funds to sit on the vault contract. So, we are transferring it to the custodian, which is a custodial wallet in Circle.

Additionally, the `FixedYieldVault` contract, which inherits from `Vault` and `MaturityVault`, includes the `withdrawERC20` function, allowing the `DEFAULT_ADMIN_ROLE` to withdraw any ERC-20 token from the contract. This setup, as confirmed by the Credbull team, meets their project requirements but introduces significant risk to users.

```
function withdrawERC20(address[] calldata _tokens) public onlyRole(DEFAULT_ADMIN
        _withdrawERC20(_tokens, msg.sender);
    }


function _withdrawERC20(address[] calldata _tokens, address _to) internal {
    for (uint256 i = 0; i < _tokens.length; i++) {
        uint256 balance = IERC20(_tokens[i]).balanceOf(address(this));
        SafeERC20.safeTransfer(IERC20(_tokens[i]), _to, balance);
```

```
    }
}
```

Moreover, `FixedYieldVault` checks whether withdrawals can be made according to specified timeframes, which the `DEFAULT_ADMIN_ROLE` can modify, and whether the contract is sufficiently funded ("matured") to cover current withdrawals and yields. Users are also unable to withdraw their deposits if the contract (FixedYieldVault or UpsideVault) is paused.

```
function _withdraw(...) onWithdrawOrRedeem(caller, receiver, owner, assets, shar
    ...
}

modifier onWithdrawOrRedeem(...)
 ... {
        _checkIsRedeemWithinWindow();
        _checkVaultMaturity();
        _;
}
```

This setup risks locking user deposits.

Assets:

- CredbullFixedYieldVault.sol [https://github.com/credbull/credbull-defi/]
- CredbullFixedYieldVaultWithUpside.sol [https://github.com/credbull/credbull-defi/]
- FixedYieldVault.sol [https://github.com/credbull/credbull-defi/]
- UpsideVault.sol [https://github.com/credbull/credbull-defi/]
- Vault.sol [https://github.com/credbull/credbull-defi/]

Status: Mitigated

Classification

Impact: 5/5

Likelihood: 3/5

Exploitability: Dependent

Complexity: Simple

Severity: Medium

Recommendations

Remediation: To mitigate the centralization risk, consider the following:

1. Implement a mechanism to securely store user funds directly in the vault, minimizing dependency on the `CUSTODIAN`.
2. Limit the `DEFAULT_ADMIN_ROLE`'s ability to withdraw arbitrary tokens and modify withdrawal timeframes. Introduce additional safeguards, such as multi-signature approvals or user consent, for these actions.
3. Ensure transparency and clear communication with users regarding the conditions under which they can withdraw their deposits, especially during paused contract states.

These measures will help reduce the risk to users and increase the system's overall security and trustworthiness.

Resolution: The provided functionality works as intended. The client notice:

> For the Custodian deposit - We are using a VASP (e.g Circle) to custody our funds. We will use the VASPs security and recovery mechanisms

The withdrawal mechanism is present for the cases when someone does direct transfers according to the provided information.

F-2024-4396 - Risk of Ownership Control Loss in Owner-Dependent Contracts - Low

Description: The `WhiteListProvider` contract uses `Ownable` functionality from OpenZeppelin, heavily relying on the owner.

If ownership is mistakenly transferred to the wrong address, functions that are allowed only to the owner may become unreachable or compromised, thereby blocking the real owner from accessing critical administrative functions.

Assets:

- WhiteListProvider.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Classification

Impact: 5/5

Likelihood: 1/5

Exploitability: Dependent

Complexity: Simple

Severity: Low

Recommendations

Remediation: Integrate `Ownable2Step` from OpenZeppelin, which introduces a two-step process for transferring ownership. This process necessitates that the prospective new owner actively accepts the ownership, thereby adding an extra security measure to prevent accidental transfers.

Resolution: The Finding is fixed according to PR#87 [commit **9361eca**]. The `Ownable2Step` was integrated. However, the `Ownable` import was not removed.

Observation Details

F-2024-4397 - Floating Pragma - Info

Description: The project uses floating pragma `^0.8.20`.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Assets:

- CredbullFixedYieldVault.sol [https://github.com/credbull/credbull-defi/]
- CredbullFixedYieldVaultFactory.sol [https://github.com/credbull/credbull-defi/]
- CredbullFixedYieldVaultWithUpside.sol [https://github.com/credbull/credbull-defi/]
- CredbullUpsideVaultFactory.sol [https://github.com/credbull/credbull-defi/]
- CredbullWhiteListProvider.sol [https://github.com/credbull/credbull-defi/]
- VaultFactory.sol [https://github.com/credbull/credbull-defi/]
- MaxCapPlugin.sol [https://github.com/credbull/credbull-defi/]
- WhiteListPlugin.sol [https://github.com/credbull/credbull-defi/]
- WindowPlugin.sol [https://github.com/credbull/credbull-defi/]
- IWhiteListProvider.sol [https://github.com/credbull/credbull-defi/]
- WhiteListProvider.sol [https://github.com/credbull/credbull-defi/]
- FixedYieldVault.sol [https://github.com/credbull/credbull-defi/]
- MaturityVault.sol [https://github.com/credbull/credbull-defi/]
- UpsideVault.sol [https://github.com/credbull/credbull-defi/]
- Vault.sol [https://github.com/credbull/credbull-defi/]

Status: Accepted

Recommendations

Remediation: Consider locking the pragma version and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

F-2024-4398 - Missing Validation for Time Windows in WindowPlugin Contract - Info

Description:

The `WindowPlugin` contract allows the setting of time windows for deposits and redemptions. These time windows are set in the `constructor` and the `_updateWindow` functions.

```
constructor(WindowPluginParams memory params) {
    depositOpensAtTimestamp = params.depositWindow.opensAt;
    depositClosesAtTimestamp = params.depositWindow.closesAt;
    redemptionOpensAtTimestamp = params.redemptionWindow.opensAt;
    redemptionClosesAtTimestamp = params.redemptionWindow.closesAt;
    checkWindow = true;
}

function _updateWindow(uint256 _depositOpen, uint256 _depositClose, uint256 _rede
    depositOpensAtTimestamp = _depositOpen;
    depositClosesAtTimestamp = _depositClose;
    redemptionOpensAtTimestamp = _redeemOpen;
    redemptionClosesAtTimestamp = _redeemClose;
}
```

However, there is no validation to ensure that the start time is earlier than the end time, nor that the deposit window precedes the redemption window.

This lack of validation can lead to incorrect contract configurations, potentially causing operational issues.

Assets:

- WindowPlugin.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation:

Consider implementing validation checks in the `constructor` and `_updateWindow` functions to ensure that the start time is less than the end time and that the deposit window occurs before the redemption window. This will help prevent incorrect configurations and ensure that the contract operates as intended.

Resolution:

The Finding is fixed according to PR#87 [commit **dedf91d**]. The `validateWindows` modifier was added to the `constructor` and `_updateWindow` functions,

```
modifier validateWindows(uint256 _depositOpen, uint256 _depositClose, uint256 _r
    if (!(_depositOpen < _depositClose && _depositClose < _redeemOpen && _redeem
        revert WindowPlugin__IncorrectWindowValues(_depositOpen, _depositClose, 
    }
    _;
}
```

F-2024-4399 - Redundant Fallback and Receive Functionality - Info

Description:

The `Vault` contract contains the receive and `fallback` functions which is designed to revert the native token payments.

```
receive() external payable {
    revert CredbullVault__NativeTransferNotAllowed();
}


fallback() external payable {
    revert CredbullVault__NativeTransferNotAllowed();
}
```

However, to establish the contract as non-receptive to payments, the inclusion of `receive` and `fallback` functions is not mandatory. Avoiding the creation of `receive` and `fallback` functions can result in Gas savings at the time of contract deployment.

Assets:

- Vault.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation: Consider removing the `receive` and `fallback` functions to design the contract as one that does not accept payments.

Resolution: The Finding is fixed according to PR#87 [commit **7226f47**]. The `receive` and `fallback` functions were removed.

F-2024-4400 - Lack of Configuration Events - Info

Description:

In the `FixedYieldVault` and `VaultFactory` contract there is a partial lack of configuration events.

This might result in decreased transparency and increased difficulty in monitoring changes effectively.

This issue affects the following functions:

`FixedYieldVault` contract:

- `toggleMaturityCheck`
- `toggleWhiteListCheck`
- `toggleWindowCheck`
- `toggleMaxCapCheck`
- `updateMaxCap`
- `updateWindow`
- `mature`

`VaultFactory` contract:

- `allowCustodian`
- `removeCustodian`

Assets:

- FixedYieldVault.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation: Consider implementing additional event logging for critical configuration changes to enhance monitoring and ensure operational transparency in the Strategy contract.

Resolution: The finding is fixed according to PR#87 [commits **e9ca043** and **a43861a**] The additional event logging was implemented.

F-2024-4403 - State Variables That Should Be Immutable - Info

Description:

In the `FixedYieldVault` contract, the `_fixedYield` variable is only updated in the constructor.

The variables to be assigned only in the contract's constructor could be set immutable to reduce gas cost when reading them.

Same for the `whiteListProvider` and `depositThresholdForWhiteListing` variables for the `WhiteListPlugin` contract.

Assets:

- WhiteListPlugin.sol [https://github.com/credbull/credbull-defi/]
- FixedYieldVault.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation: Make the `_fixedYield` , `whiteListProvider` , and `depositThresholdForWhiteListing` variables immutable.

Resolution: The Finding is fixed according to PR#87 [commit **1259db9**]. The mentioned variables are marked immutable.

F-2024-4414 - Missing Zero Addresses Validations Leading to Misconfiguration - Info

Description:

In the `WhiteListProvider` contract, the `updateStatus` function does not validate whether `_addresses` are non-zero.

```
function updateStatus(...) ... {
    ...
    for (uint256 i; i < length;) {

        isWhiteListed[_addresses[i]] = _statuses[i];

        unchecked {
            ++i;
        }
    }
}
```

Additionally, in the `FixedYieldVault` contract, the `constructor` does not verify that `params.roles.owner` and `params.roles.operator` are non-zero addresses.

```
constructor(FixedYieldVaultParams memory params) ... {
    _grantRole(DEFAULT_ADMIN_ROLE, params.roles.owner);
    _grantRole(OPERATOR_ROLE, params.roles.operator);

    ...
}
```

In the `VaultFactory` contract, the `owner`, operator and `custodians` are not verified for being non-zero. In the `allowCustodian` function, the `_custodian` is not verified for being non-zero.

```
constructor(address owner, address operator, address[] memory custodians) {
    _grantRole(DEFAULT_ADMIN_ROLE, owner);
    _grantRole(OPERATOR_ROLE, operator);

    bool[] memory result = new bool[](custodians.length);

    for (uint256 i = 0; i < custodians.length; i++) {
        result[i] = allowedCustodians.add(custodians[i]);
    }
}

function allowCustodian(address _custodian) public onlyRole(DEFAULT_ADMIN_ROLE)
```

```
        return allowedCustodians.add(_custodian);
    }
```

This oversight can lead to misconfiguration issues, potentially causing functionality failures or security vulnerabilities.

Assets:

- WhiteListProvider.sol [https://github.com/credbull/credbull-defi/]
- FixedYieldVault.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation: Consider implementing checks in the `updateStatus` function of the `WhiteListProvider` contract to ensure `_addresses` are non-zero. Similarly, in the `FixedYieldVault` contract, validate that `params.roles.owner` and `params.roles.operator` are non-zero addresses during the constructor's execution.

Resolution: The Finding is fixed according to PR#87 [commit **0a8f658** and **4a001d6**]. Addresses checks for being non-zero were implemented.

F-2024-4416 - Missing Validation for collateralPercentage and twap in UpsideVault Contract - Info

Description:

In the `UpsideVault` contract, the `constructor` does not check whether `params.collateralPercentage` is less than or equal to the `MAX_PERCENTAGE` value.

This validation is also missing in the `setTWAP` function when setting the `twap` value.

```
constructor(...) ... {

    collateralPercentage = params.collateralPercentage;
    ...

}

function setTWAP(uint256 _twap) public onlyRole(OPERATOR_ROLE) {
    twap = _twap;
}
```

Lack of these checks can lead to misconfigurations, potentially causing incorrect contract behavior.

Assets:

- UpsideVault.sol [https://github.com/credbull/credbull-defi/]

Status: Fixed

Recommendations

Remediation:

Consider adding validation checks in the `UpsideVault` contract's constructor to ensure that `params.collateralPercentage` does not exceed `MAX_PERCENTAGE`. Similarly, implement checks in the `setTWAP` function to verify that the `twap` value is within acceptable limits. These validations will help prevent misconfigurations and ensure the contract operates as intended.

Resolution:

The Finding is fixed according to PR#87 [commit **4a001d6**] The validation for `params.collateralPercentage` was added. `twap` value can be any according to the information provided by the team.

F-2024-4419 - Violation of Checks-Effects-Interactions Pattern - Info

Description:

In the contracts, there are certain functions that do not adhere to the Checks-Effects-Interactions (CEI) pattern.

`Vault` contract:

- `_deposit`
- `_withdraw`

`UpsideVault` contract:

- `_deposit`
- `_withdraw`

The absence of this pattern can lead to vulnerabilities where interactions with external contracts occur before all checks and state changes have been completed, potentially compromising contract integrity.

Assets:

- UpsideVault.sol [https://github.com/credbull/credbull-defi/]
- Vault.sol [https://github.com/credbull/credbull-defi/]

Status: Accepted

Recommendations

Remediation:

Consider implementing the Checks-Effects-Interactions (CEI) pattern for the functions identified in the contracts. Adopting this pattern will help prevent reentrancy attacks and ensure that all validations and state updates are completed before any external interactions, thereby enhancing the security and reliability of the contracts.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details	
Repository	https://github.com/credbull/credbull-defi/
Commit	f927c850b628a63dd71e5c6a1e8888e89a6262e5
Remediation commit	ce1c17ea94494d985c516f0ac3be379cb2fef8e3
Whitepaper	Credbull High Level Architecture (SHA256: 13117d79fe1da3f456fad5b0c0d24c956bb8552b1e9c40b97c2d90c6f50cf4f6)
	https://docs.credbull.io/docs/litepaper

Contracts in Scope
./packages/contracts/src/factory/VaultFactory.sol
/packages/contracts/src/plugin/MaxCapPlugin.sol
/packages/contracts/src/plugin/WhiteListPlugin.sol
/packages/contracts/src/plugin/WindowPlugin.sol
/packages/contracts/src/provider/whiteList/IWhiteListProvider.sol
/packages/contracts/src/provider/whiteList/WhiteListProvider.sol
/packages/contracts/src/vault/FixedYieldVault.sol
/packages/contracts/src/vault/MaturityVault.sol
/packages/contracts/src/vault/UpsideVault.sol
/packages/contracts/src/vault/Vault.sol
/packages/contracts/src/CredbullFixedYieldVault.sol
/packages/contracts/src/CredbullFixedYieldVaultFactory.sol
/packages/contracts/src/CredbullFixedYieldVaultWithUpside.sol
/packages/contracts/src/CredbullUpsideVaultFactory.sol
/packages/contracts/src/CredbullWhiteListProvider.sol