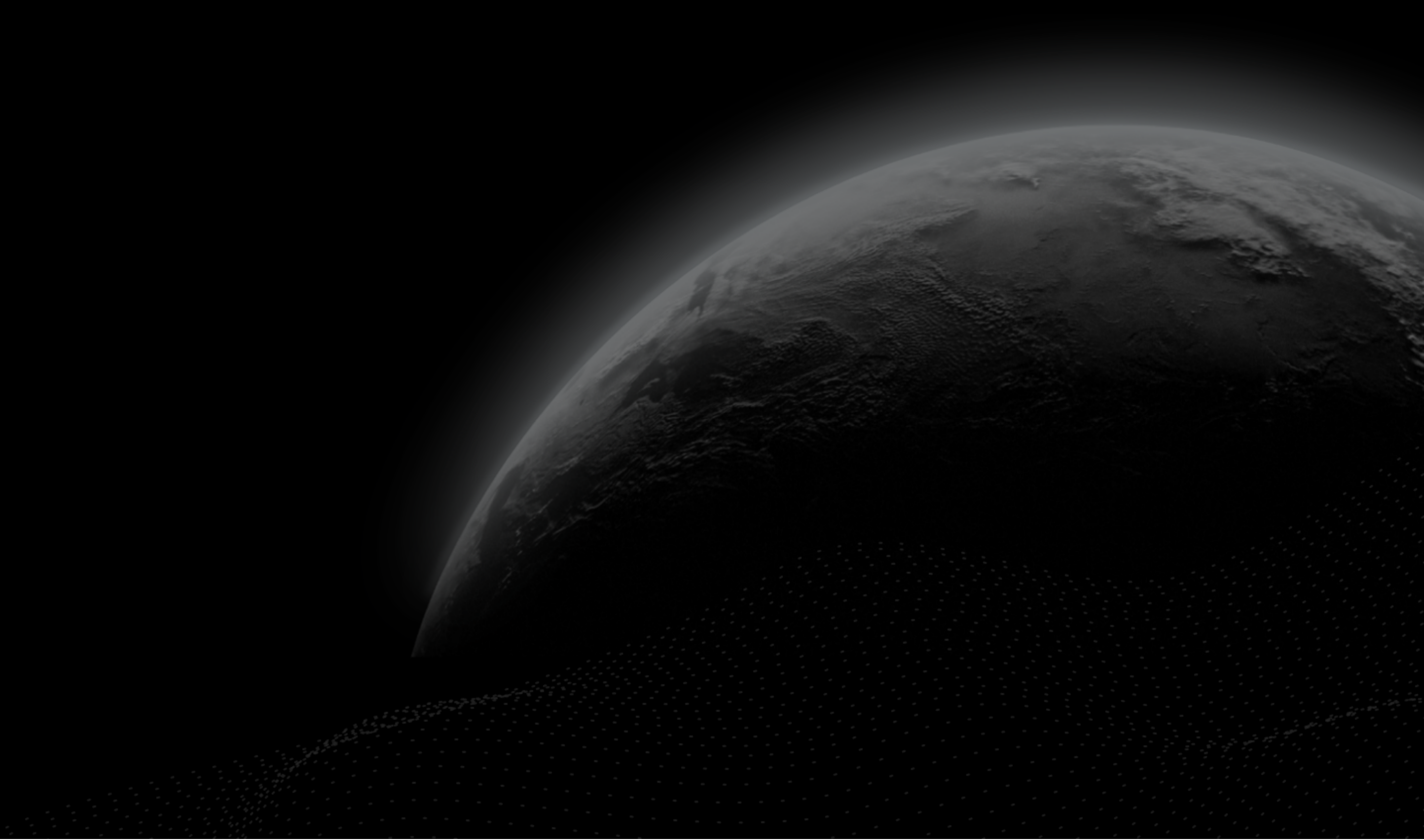
Security Assessment

# Venus - Vaults Audit

CertiK Assessed on Jul 4th, 2023



CertiK Assessed on Jul 4th, 2023

# Venus - Vaults Audit

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

## Executive Summary

| | | |
|---|---|---|
| TYPES | ECOSYSTEM | METHODS |
| DeFi | Ethereum (ETH) | Manual Review, Static Analysis |
| LANGUAGE | TIMELINE | KEY COMPONENTS |
| Solidity | Delivered on 07/04/2023 | N/A |

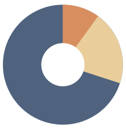
CODEBASE

d4f48ad6a032f1764e607c4cf0a9d8769230f2a0

a158f8c335d0cfad71f1d2c27af6b0d92f4abe41

View All in Codebase Page

## Vulnerability Summary



| 10 | 3 | 1 | 2 | 4 | 0 |
|---|---|---|---|---|---|
| Total Findings | Resolved | Mitigated | Partially Resolved | Acknowledged | Declined |

| | | | | |
|---|---|---|---|---|
| 🟥 | 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. |
| 🟧 | 1 | Major | 1 Mitigated | 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. |
| 🟨 | 0 | Medium | | Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform. |
| 🟨 | 2 | Minor | 1 Partially Resolved, 1 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. |
| 🟦 | 7 | Informational | 3 Resolved, 1 Partially Resolved, 3 Acknowledged | 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 | VENUS - VAULTS AUDIT

# CODEBASE | VENUS - VAULTS AUDIT

## Repository

d4f48ad6a032f1764e607c4cf0a9d8769230f2a0

a158f8c335d0cfad71f1d2c27af6b0d92f4abe41

## Deployed contracts

### Proxys:

- `VRTVaultProxy` : 0x98bf4786d72aaef6c714425126dd92f149e3f334
- `VAIVaultProxy` : 0x0667eed0a0aab930af74a3dfedd263a73994f216
- `XVSVaultProxy` : 0x051100480289e704d20e9db4804837068f3f9204

### Implementations:

- `VRTVault` : 0xea98e94d35120b23f9f9f20a7314804d4ab491f1
- `VAIVault` : 0xa52f2a56abb7cbdd378bc36c6088fafeaf9ac423
- `XVSVault` : 0x0cf9a22e790d89b8e58469f217b50bb4c3ab068c
- `XVSStore` : 0x1e25cf968f12850003db17e0dba32108509c4359

# AUDIT SCOPE | VENUS - VAULTS AUDIT

7 files audited ● 4 files with Acknowledged findings ● 1 file with Resolved findings ● 2 files without findings

| ID | File | SHA256 Checksum |
|---|---|---|
| ● VRT | contracts/VRTVault/VRTVault.sol | 0e6562c8fb43ed61cb52be1cbb9858917b9ed66c884e2c27ae677c18dda477b0 |
| ● VAI | contracts/Vault/VAIVault.sol | f676b94a7d4b6023d1888efa2b52ad1926c7bb5f038ffbd65eb12d203407342c |
| ● XVS | contracts/XVSVault/XVSStore.sol | 31e037224032384a188c118dfdf05b45be4310733c27baab3697f265030f0891 |
| ● XVV | contracts/XVSVault/XVSVault.sol | 1fc39155c8b48d3d1fa9881b104c37c2a391681b263b4d00f8cb1dafe718bc8c |
| ● XVX | contracts/XVSVault/XVSVaultStorage.sol | c6f18cd787ee4ce780adcd149c6b6f0614620fd75fbee153838673519706570b |
| ● VRV | contracts/VRTVault/VRTVaultStorage.sol | cd25bc855dee462c79ee6621e495686b3cf9e27b5700ce6e619db4be27c15477 |
| ● VAV | contracts/Vault/VAIVaultStorage.sol | fa020a29acddc9f9968d2aef37172d961879fbdc4b96c407d507316fda61aa53 |

# APPROACH & METHODS | VENUS - VAULTS AUDIT

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

# REVIEW NOTES | VENUS - VAULTS AUDIT

In this audit report, we have focused on reviewing the changes made to three distinct vaults within Venus protocol:

- `VRTVault`
- `VAIVault`
- `XVSVault`

Our primary emphasis was on the differences introduced by our client to these vault contracts to ensure no safety issues are introduced in the new versions. It is a more specialized examination as opposed to a complete re-evaluation, allowing us to provide a precise and efficient review.

## VRTVault

The `VRTVault` serves as a depository for users to stake their `VRT` tokens. These staked tokens earn `VRT` rewards proportionate to the volume of tokens deposited and the duration of their stay in the vault.

The recent updates encompass modifications to the pausing mechanism and privileged role management. Additionally, a new state variable, `lastAccruingBlock` , was introduced. This variable denotes the block number beyond which the accrual of interest ceases.

## VAIVault

The `VAIVault` operates as a farming platform where users can stake `VAI` tokens to earn `XVS` tokens. This is made possible through an implemented share mechanism.

In the latest changes, the option to transfer or remove the admin role has been eliminated. Additionally, the management of privileged roles has been updated. A new pausing mechanism has also been introduced which, when activated in case of an emergency, prevents users from performing actions such as depositing, withdrawing, or claiming rewards.

## XVSVault

The `XVSVault` enables users to deposit tokens into pre-established pools, which can be added by privileged accounts. When users deposit `XVS` tokens, they receive delegates that grant them participation rights in the Venus protocol's governance mechanism.

With the recent updates, additional checks have been implemented to existing functions, aimed at preventing minor errors. Furthermore, a new function has been added to improve the transfer of rewards. The ability to transfer or remove the admin role has been taken away, and there have been adjustments to how certain privileged roles are managed. A new pause mechanism has been introduced, which can prevent, in case of an emergency, users from depositing tokens, making withdrawals, claiming rewards, and delegating votes. In addition, modifications to the storage infrastructure have been implemented, ensuring consistency in the reward mechanism both pre and post-upgrade.

# DEPENDENCIES | VENUS - VAULTS AUDIT

## Out Of Scope Dependencies

The protocol is serving as the underlying entity to interact with out-of-scope dependencies. The out-of-scope dependencies that the contracts interact with are:

```
    IBEP20 public vrt;
```

- The contract `VRTVaultStorage` interacts with `IBEP20` interface via `vrt`.

```
    IBEP20 public xvs;
```

- The contract `VAIVaultStorage` interacts with `IBEP20` interface via `xvs`.

```
    IBEP20 public vai;
```

- The contract `VAIVaultStorage` interacts with `IBEP20` interface via `vai`.

```
    function safeRewardTransfer(address token, address _to, uint256 _amount) external onlyOwner {
```

- The function `XVSStore.safeRewardTransfer` interacts with `IBEP20` interface via `token`.

```
    function emergencyRewardWithdraw(address _tokenAddress, uint256 _amount) external onlyOwner {
```

- The function `XVSStore.emergencyRewardWithdraw` interacts with `IBEP20` interface via `_tokenAddress`.

```
    function _transferReward(address rewardToken, address userAddress, uint256 amount) internal {
```

- The function `XVSVault._transferReward` interacts with `IBEP20` interface via `rewardToken`.

`XVSVault` , `VAIVault` , and `VRTVault` rely on `@venusprotocol/governance-contracts/contracts/Governance/AccessControlledV5.sol` to handle certain priviledged functions.

The scope of the audit treats out-of-scope dependencies as black boxes and assumes their functional correctness.

## Assumptions

Within the scope of the audit, assumptions are made about the intended behavior of the protocol in order to inspect consequences based on those behaviors. Assumptions made within the scope of this audit include:

- `vrt` , `xvs` , `vai` are meant to be tokens from the Venus protocol;
- other tokens are valid, trusted, non-deflatianory `BEP20` contracts;
- `@venusprotocol/governance-contracts/contracts/Governance/AccessControlledV5.sol` has no vulnerabilities.

## Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced. Additionally, we recommend all out-of-scope dependencies are carefully vetted to ensure they function as intended. Last, we recommend all assumptions about the behavior of the project are thoroughly reviewed and, if the assumptions do not match the intention of the protocol, documenting the intended behavior for review.

# FINDINGS | VENUS - VAULTS AUDIT

| 10 | 0 | 1 | 0 | 2 | 7 |
|---|---|---|---|---|---|
| Total Findings | Critical | Major | Medium | Minor | Informational |

This report has been prepared to discover issues and vulnerabilities for Venus - Vaults Audit. Through this audit, we have uncovered 10 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

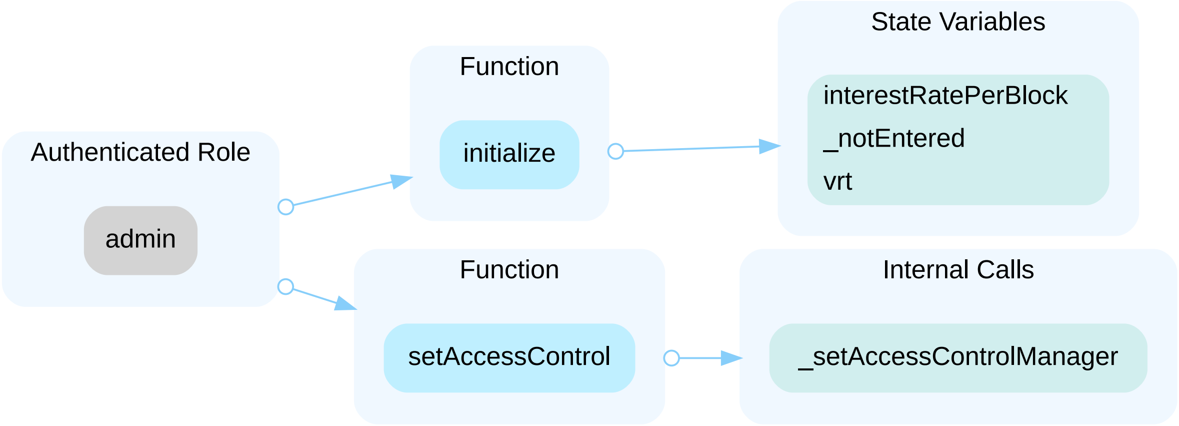
| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **GLOBAL-01** | **Centralization Related Risks** | **Centralization** | **Major** | ● **Mitigated** |
| VPB-05 | Missing Upper Bound | Volatile Code | Minor | ● Partially Resolved |
| XVS-02 | Missing Zero Address Validation | Volatile Code | Minor | ● Acknowledged |
| VPB-02 | Comparison To Boolean Constant | Coding Style | Informational | ● Partially Resolved |
| VPB-03 | Missing Emit Events | Coding Style | Informational | ● Acknowledged |
| VRT-03 | Unused Event | Coding Style | Informational | ● Resolved |
| VRT-05 | Typo | Coding Style | Informational | ● Resolved |
| XVV-02 | Check Effect Interaction Pattern Violated (Out-Of-Order Events) | Volatile Code | Informational | ● Acknowledged |
| XVX-01 | Unused Library | Coding Style | Informational | ● Acknowledged |
| XVX-02 | Possible Overflow | Logical Issue | Informational | ● Resolved |

# GLOBAL-01 | CENTRALIZATION RELATED RISKS

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization | ● Major | | ● Mitigated |

## Description

In the contract `VRTVault` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and modify the access control.



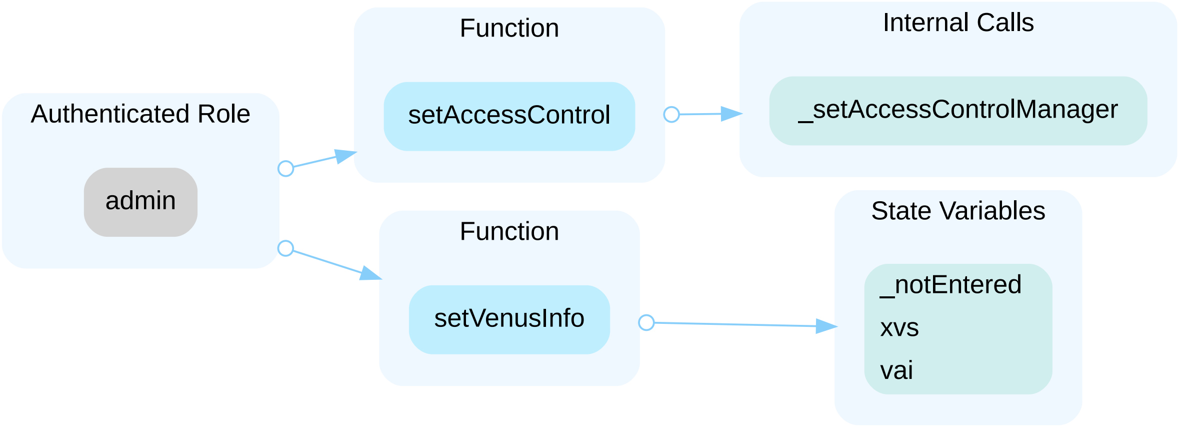
Other privileged accounts have authority over the functions:

- `pause()`;
- `resume()`;
- `withdrawBep20()`;
- `setLastAccruingBlock()`;

Any compromise to one of these accounts may allow the hacker to take advantage of this authority and:

- pause the contract, preventing users from withdrawing their tokens;
- transfer any token from the contract to an address they control;
- set an extreme value to `lastAccruingBlock` so interests keep being accrued.

In the contract `VAIVault` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and modify the access control.



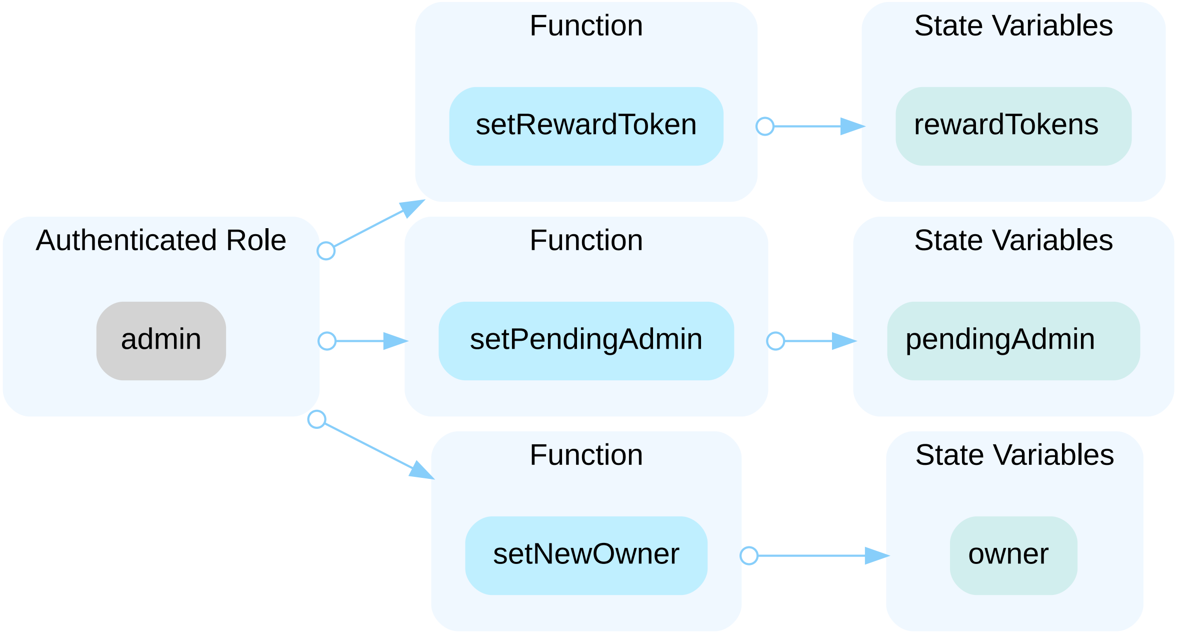
Other privileged accounts have authority over the functions:

- `pause()` ;
- `resume()` .

Any compromise to one of the privileged addresses may allow the hacker to take advantage of this authority and pause or unpause the contract.

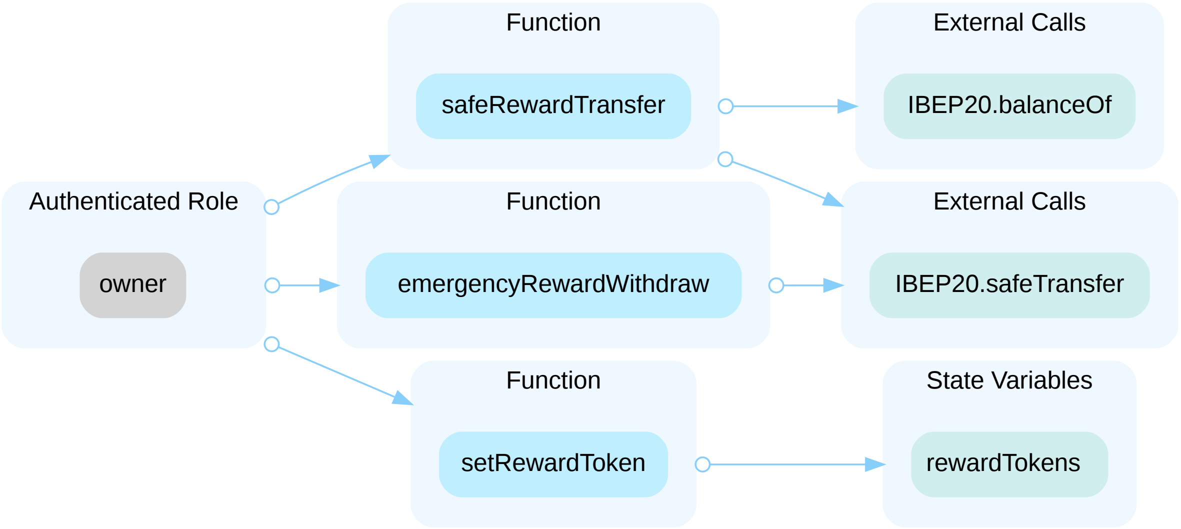
In the contract `XVSStore` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and :

- set an address they control as the new `pendingAdmin` ;
- set an address they control as the new `owner` ;
- set any malicious token contract as a reward token.

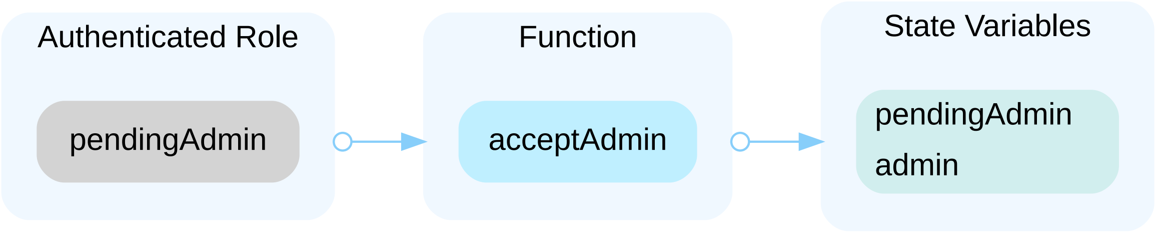


In the contract `XVSStore` the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority and:

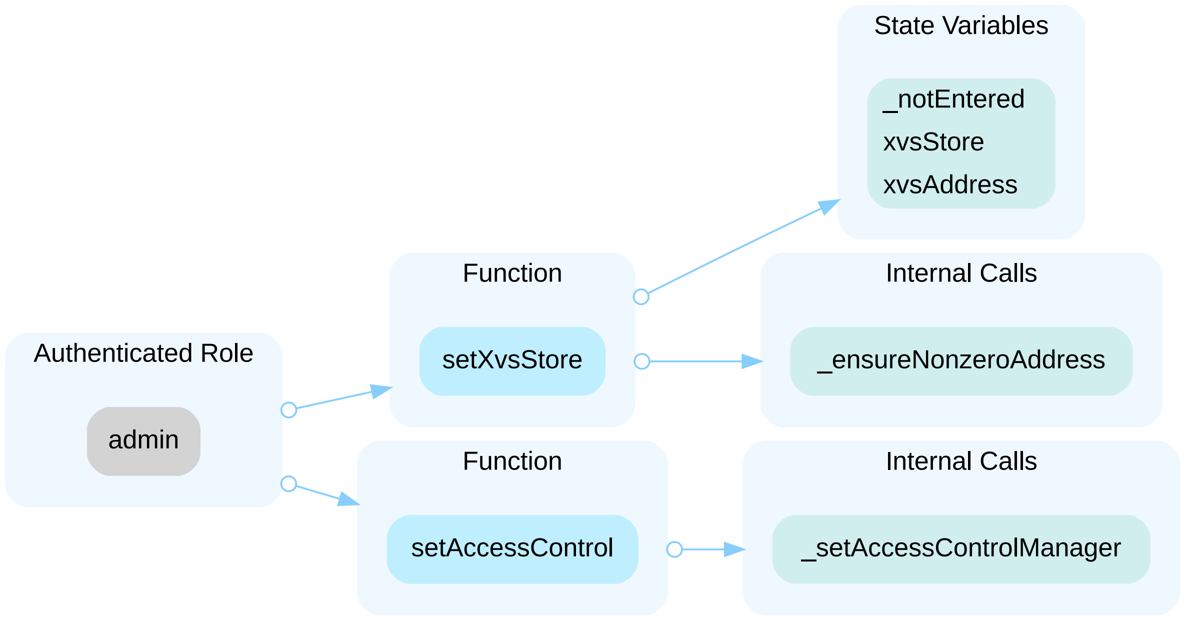
- transfer reward tokens from the contract to an address they control;
- transfer any token to its own address.
- set any malicious token contract as a reward token.



In the contract `XVSStore` the role `pendingAdmin` has authority over the functions shown in the diagram below. Any compromise to the `pendingAdmin` account may allow the hacker to take advantage of this authority and become the new admin.



In the contract `XVSVault` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and modify the access control.



Other privileged accounts have authority over the functions:

- `pause()`
- `resume()`
- `add()`
- `set()`
- `setRewardAmountPerBlock()`
- `setWithdrawalLockingPeriod()`

Any compromise to one of the privileged addresses may allow the hacker to take advantage of this authority and - pause or unpause the contract;

- add a new pool with malicious tokens and set its reward allocation very high to steal reward tokens;
- set a significantly high reward amount per block for certain tokens, allowing the hacker to earn more reward tokens than expected.
- set the `lockPeriod` of a specific pool extremely high so users cannot request withdrawal anymore.

## 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 (⅔, ⅗) 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.

## Alleviation

`[Venus]` :

**ACM :**

Regarding the ACM instance, we'll set and user in every case this contract: 0x4788629abc6cfca10f9f969efdeaa1cf70c23555

In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal Timelock) has the `DEFAULT_ADMIN_ROLE`. And this contract is a Timelock contract used during the Venus Improvement Proposals.

The idea is to grant 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 to execute the mentioned functions. Moreover, we'll allow:

- 0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02 (Fast-track) and
- 0x213c446ec11e45b15a6E29C1C1b402B8897f606d (Critical)

also to execute pause() and resume(). These are the Timelock contracts to execute VIP's with a shorter delay. Specifically, the current config for the three Timelock contracts are:

- normal: 24 hours voting + 48 hours delay
- fast-track: 24 hours voting + 6 hours delay
- critical: 6 hours voting + 1 hour delay

**Admin/owners :**

Regarding the admin/owners:

- `VRTVault.admin` : 0x1c2cac6ec528c20800b2fe734820d87b581eaa6b. Multisig wallet, which will be replaced by the Normal Timelock contract 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 during the VIP where we'll release these changes:
- `VAIVault.admin` : 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396. It's the Normal Timelock used in the Governance processes;
- `XVSStore` :
  - `XVSStore.admin` : 0x1c2cac6ec528c20800b2fe734820d87b581eaa6b, Multisig wallet. We will transfer the admin role in the `XVSStore` to the Normal Timelock contract 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396
  - `XVSStore.pendingAdmin` . 0x0. Used during the transfer of the admin role;
  - `XVSStore.owner` : 0x051100480289e704d20e9db4804837068f3f9204, the `XVSVaultProxy` ;
  - `XVSVault.admin` : 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396. It's the Normal Timelock used in the Governance processes.

`[CertiK]` : The mitigation strategy given above should mitigate the centralization risk.

**2023/06/15 - 00:15 UTC, at block height: 29108290**

Currently, none of these three contracts:

- 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396
- 0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02 (Fast-track)
- 0x213c446ec11e45b15a6E29C1C1b402B8897f606d (Critical)

are granted to execute any of the functions.

The admin of `XVSVaultProxy` , `VRTVaultProxy` and `VAIVaultProxy` is 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396.

**2023/06/26 - 17:21:35 UTC, at block height: 29444597**

`XVSStore.admin` has been transferred to the Normal Timelock in this transaction: 0xc0c3f761cdd06a80df30922a0fe22bf8c91f3d5cadf89181f05fa03a13c029da.

# VPB-05 | MISSING UPPER BOUND

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Minor | contracts/VRTVault/VRTVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 280~281; contracts/XVSVault/XVSVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 222~223, 238~239 | ● Partially Resolved |

## Description

In `VRTVault`, the function `setLastAccruingBlock()` allows setting `lastAccruingBlock` to an arbitrary value.

---

In `XVSVault`:

- the function `setRewardAmountPerBlock()` allows setting `rewardTokenAmountsPerBlock[_rewardToken]` to an arbitrary value.
- the function `setWithdrawalLockingPeriod()` allows setting `pool.lockPeriod` to an arbitrary value.

## Recommendation

We recommend introducing a reasonable upper limit in the setting functions in order to prevent excessively high values that could potentially disrupt the protocol.

## Alleviation

`[Venus]`:

- `VRTVault`: we have defined a constant with a block number close to year 3,000 (assuming 3 seconds per block), and used it as an upper bound
- `XVSVault.setRewardAmountPerBlock`: we won't limit it, because the limit would depend on the reward token, and we prefer to limit the number of changes in this release
- `XVSVault.setWithdrawalLockingPeriod`: we require _newPeriod < 60 * 60 * 24 * 365 * 10, so we wouldn't allow lock periods greater than 10 years.

Commit: a158f8c335d0cfad71f1d2c27af6b0d92f4abe41.

# XVS-02 | MISSING ZERO ADDRESS VALIDATION

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Minor | contracts/XVSVault/XVSStore.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 60 | ● Acknowledged |

## Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
60          pendingAdmin = _admin;
```

- `_admin` is not zero-checked before being used.

## Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

## Alleviation

`[Venus]` : `XVSStore` will not be changed in this release, so no updates to `XVSStore` contract will be done

# VPB-02 | COMPARISON TO BOOLEAN CONSTANT

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | contracts/VRTVault/VRTVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 76; contracts/Vault/VAIVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 56; contracts/XVSVault/XVSStore.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 46; contracts/XVSVault/XVSVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 108 | ● Partially Resolved |

## Description

Boolean constants can be used directly and do not need to be compared to true or false.

```
76          require(vaultPaused == false, "Vault is paused");
```

```
56          require(vaultPaused == false, "Vault is paused");
```

```
46          require(rewardTokens[token] == true, "only reward token can");
```

```
108          require(vaultPaused == false, "Vault is paused");
```

## Recommendation

We recommend removing the equality to the boolean constant.

## Alleviation

`[Venus]` : Fixed for `VRTVault` , `XVSVault` and `VAIVault` . `XVSStore` won't be upgraded in this release, so, we didn't change it

Commit: 1a47e51eae5cf2180b0034f61159cf1fa412e37f.

# VPB-03 | MISSING EMIT EVENTS

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | contracts/VRTVault/VRTVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 56, 311; contracts/Vault/VAIVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 235; contracts/XVSVault/XVSStore.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 45, 83, 88; contracts/XVSVault/XVSVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 845 | ● Acknowledged |

## Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

## Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

## Alleviation

`[Venus]` : Issue acknowledged. I won't make any changes for the current version.

`VRTVault.sol#L311` - `_setAccessControlManager()` already emit an event

`VAIVault.sol#L235` - `_setAccessControlManager()` already emit an event

For the rest of the suggestions, we won't change the codebase because:

- `XVSStore` : we are not upgrading it in this release
- `VRTVault.initialize` won't be used anymore

# VRT-03 | UNUSED EVENT

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | contracts/VRTVault/VRTVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 19 | ● Resolved |

## Description

Some events are never emitted, which can lead to confusion and code maintainability issues.

```
19      event AdminTransfered(address indexed oldAdmin, address indexed newAdmin);
```

- `AdminTransfered` is declared in `VRTVault` but never emitted.

## Recommendation

It is recommended to remove the unused events or emit them in the intended functions to improve code clarity and maintainability.

## Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding, commit: df23556727d2b5f13326e6deffcec7637270f642.

# VRT-05 | TYPO

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | contracts/VRTVault/VRTVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 97 | ● Resolved |

## Description

In `VRTVault.sol` , the error message from the modifier `userHasPosition()` is missing a space.

## Recommendation

We recommend correcting the typo.

## Alleviation

`[CertiK]` : The team heeded the advice and resolved the finding, commit: 6b7b8b71f9a93613b11ff881cf5a52ff8ef6931b.

# XVV-02 | CHECK EFFECT INTERACTION PATTERN VIOLATED (OUT-OF-ORDER EVENTS)

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | ● Informational | contracts/XVSVault/XVSVault.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 193, 195 | ● Acknowledged |

## Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

*This finding is considered minor because the reentrancy only causes out-of-order events.*

### External call(s)

```
193         IXVSStore(xvsStore).setRewardToken(_rewardToken, true);
```

### Events emitted after the call(s)

```
195         emit PoolAdded(_rewardToken, poolInfo.length - 1, address(_token),
_allocPoint, _rewardPerBlock, _lockPeriod);
```

## Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

## Alleviation

`[Venus]` : Issue acknowledged. I won't make any changes for the current version.

`XVSStore` is already set in the `XVSVault` , and cannot be changed in the deployed contract.

# XVX-01 | UNUSED LIBRARY

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | contracts/XVSVault/XVSStore.sol (for_tools): 6 | ● Acknowledged |

## Description

In the contract `XVSStore`, the library `SafeMath` is declared on a using-for directive but its functionalities are never used, which can lead to unnecessary complexity and reduced maintainability.

## Recommendation

It is advised to ensure that all necessary contracts and libraries are used, and remove redundant constracts and libraries.

## Alleviation

`[Venus]`: XVSStore will not be changed in this release, so no updates to `XVSStore` contract will be included.

# XVX-02 | POSSIBLE OVERFLOW

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | ● Informational | contracts/XVSVault/XVSVaultStorage.sol (d4f48ad6a032f1764e607c4cf0a9d8769230f2a0): 58~62 | ● Resolved |

## Description

The following struct in `XVSVaultStorage` was changed between implementations.

From

```
struct WithdrawalRequest {
       uint256 amount;
       uint256 lockedUntil;
   }
```

to the struct

```
struct WithdrawalRequest {
        uint256 amount;
        uint128 lockedUntil;
        uint128 afterUpgrade;
    }
```

Note that both structs will only take up 2 storage slots, so there will not be any storage collisions. However the second slot now packs two `uint128` values together. If `lockedUntil` is greater than the maximum `uint128`, then it will overflow when the implementation is upgraded so that the new `lockedUntil` will be a smaller value. It will also make `afterUpgrade` a nonzero value to start.

We give this an informational severity as it is very unlikely any of these values will be this large as it would require a `pool.lockPeriod` to be set to an extremely large value.

## Recommendation

We recommend ensuring that a `pool.lockPeriod` was never set to a large enough value during the lifetime of the proxy so that the `block.timestamp` plus this value would exceed the maximum `uint128`.

## Alleviation

`[Venus]` : There is only one pool. The lockPeriod was set when the pool was created, to 604800.

# APPENDIX | VENUS - VAULTS AUDIT

## 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. |
| 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. |

## 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.

# 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.