Security Assessment

Alchemix Protocol

May 18th, 2021

Table of Contents

Summary

This report has been prepared for Alchemix Protocol smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract 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:

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

Overview

Project Summary

Project Name	Alchemix Protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/alchemix-finance/alchemix-protocol/
Commits	2099ed3b81e8727289fc11994b4389ea118577a2

Audit Summary

Delivery Date	May 18, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

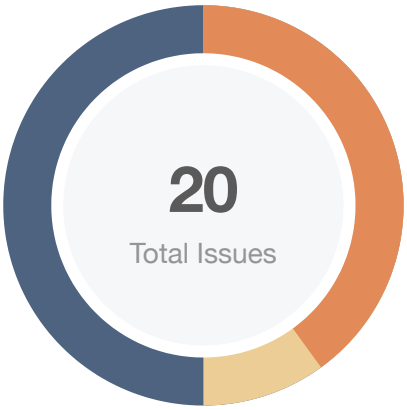
Vulnerability Summary

Total Issues	20
Critical	0
Major	8
Medium	0
Minor	2
Informational	10
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
ATE	AlToken.sol	b6233067dff833fe9f0b72b280e25d13fcc5ab35cb195d4ae9582c2b2131c54d
ALC	Alchemist.sol	8da1fc58c2ed7cc7b23c2783aa730497c871fd5f66d6fa8faddd51174ec28986
ATC	AlchemixToken.sol	75b1cad51636f733775606e69d5b1812353a7e14054993170c76e3af4a108d6a
MSW	MultiSigWallet.sol	10e5e7781958a4cc734bae848ce89f551e33da513f7cfc6e8958691170295ba2
MST	MultiSigWalletWithTimelock.sol	f92db429228a4427660af4d61575f78f8795d6c89f5f9a3baa615bf84b39611a
SPE	StakingPools.sol	af90115a5add77caf89360adec3427f5f1e5e5648d3b076bc7b0e9757eaa0e46
TTE	TimeToken.sol	f59363fe3abbb8a930840078ed550c51fe32d0947246c116b0c8ebd1ccff93f7
TRA	Transmuter.sol	a6543305cecdd8eef07953626d33319f7121303fad5729e343bd3bd4d15ee430
TBE	TransmuterB.sol	63e53a19807009afeefe8dec025c1efb9e072801812e61115d9030eb0c3df9eb
YVA	adapters/YearnVaultAdapter.sol	c3ac1fec1f7f229b6248a0931116de5f17c2b066a40cbb91c05849b61cdc37c0

Findings



Critical	0 (0.00%)
Major	8 (40.00%)
Medium	0 (0.00%)
Minor	2 (10.00%)
Informational	10 (50.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
ALC-01	Single Source of Price Oracle	**Centralization / Privilege**	**Informational**	**Acknowledged**
ALC-02	Missing Emit Event	Coding Style	Informational	Acknowledged
ALC-03	Recommended Explicit Vault Validity Checks	Logical Issue	Informational	Acknowledged
ALC-04	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
ATC-01	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
ATE-01	Inaccurate Comment	Inconsistency	Minor	Acknowledged
ATE-02	Centralized Risk to Sensitive Functions	**Centralization / Privilege**	**Major**	**Acknowledged**
MSW-01	Lack of Input Validation	Logical Issue	Minor	Acknowledged
SPE-01	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
TBE-01	No Log In `require()` Check	Coding Style	Informational	Acknowledged
TBE-02	Comment Typo	Coding Style	Informational	Acknowledged
TBE-03	Minimize The Scope of Access To The Function	Control Flow	Informational	Acknowledged

ID	Title	Category	Severity	Status
TBE-04	Recommended Explicit Vault Validity Checks	Logical Issue	Informational	Acknowledged
TBE-05	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
TBE-06	Centralized Risk To Sensitive `withdraw`	**Centralization / Privilege**	**Major**	**Acknowledged**
TRA-01	Comment Typo	Coding Style	Informational	Acknowledged
TRA-02	Minimize The Scope of Access To The Function	Control Flow	Informational	Acknowledged
TTE-01	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
YVA-01	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
YVA-02	Lack of Input Validation	Volatile Code	Informational	Acknowledged

ALC-01 | Single Source of Price Oracle

Category	Severity	Location	Status
Centralization / Privilege	**Informational**	Alchemist.sol: 675	**Acknowledged**

Description

Chainlink is the only price oracle that provides the price in the Alchemix project. If the single price oracle provides an incorrect price, this error will dominate the price and cause single point of failure by affecting the token price.

Recommendation

In order to prevent the single point of failure issue and protect the Alchemix project from the fluctuation of the price caused by price oracle, we advise the client to adopt multiple price oracles as token price references.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

ALC-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	Informational	Alchemist.sol: 278, 331	Acknowledged

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setOracleAddress()`
- `setFlushActivator()`

Recommendation

We advise the client to consider adding events for sensitive actions, and emit them in the corresponding functions.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

ALC-03 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	Alchemist.sol: 715	Acknowledged

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

ALC-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Alchemist.sol: 705	**Acknowledged**

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hacker who compromise the account with `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role `governor`'s account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the `governance` role was given to the Alchemix Multisig. The Alchemix Multisig consists of 4 core Alchemix team members and 4 trusted community members. The official AlchemixDAO is currently under development and will offer an even more decentralized approach to governance of the Alchemix protocol and its contracts.

ATC-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	AlchemixToken.sol: 45	**Acknowledged**

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

We advise the client to carefully manage the `MINTER_ROLE` role account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the MINTER_ROLE role was given to the StakingPools contract, and revoked from the deploying EOA.

ATE-01 | Inaccurate Comment

Category	Severity	Location	Status
Inconsistency	Minor	AlToken.sol: 60, 64	Acknowledged

Description

The comment in L60 shows that only the caller that has the minter role can call the function `mint()`, which is not accurate as there's no `Minter_Role` in the contract `AlToken`.

Recommendation

We advise the client to add `Minter_Role` and corresponding modifier to restrict the access to the function `mint()`.

Alleviation

`[Alchemix]`: While the comment is inaccurate, the intended restriction functionality still exists: the `onlyWhitelisted` modifier ensures that only a whitelisted address is able to call the mint function. The only whitelisted address is the Alchemist.

ATE-02 | Centralized Risk to Sensitive Functions

Category	Severity	Location	Status
Centralization / Privilege	**Major**	AlToken.sol: 77, 84, 102	**Acknowledged**

Description

The owner of the account `owner` can update the ceiling of a token that is allowed to mint, add the account to which the minted token can be transferred, and grant `SENTINEL_ROLE` to any address in the contract `AlToken()`

Recommendation

We advise the client to carefully manage the `owner` account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the `owner` role was given to the Alchemix Multisig. The Alchemix Multisig consists of 4 core Alchemix team members and 4 trusted community members. The official AlchemixDAO is currently under development and will offer an even more decentralized approach to governance of the Alchemix protocol and its contracts.

MSW-01 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	MultiSigWallet.sol: 157	Acknowledged

Description

The value of `newOwner` argument is not validated as non-zero value. An invalid owner address will prevent any fund withdraw from its wallet.

Recommendation

We advise the client to add a argument validator to check if the value of `newOwner` is set as address(0)

Alleviation

`[Alchemix]`: The `ownerDoesNotExist` modifier prevents address(0) from being declared as an owner more than once. The Alchemix Multisig is a 5/8 multisig, so even if 1 of the owners were changed to address(0), there would still be 7 valid signers.

SPE-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	StakingPools.sol: 118	**Acknowledged**

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hacker who compromise the account with `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role `governor`'s account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the `governance` role was given to the Alchemix Multisig. The Alchemix Multisig consists of 4 core Alchemix team members and 4 trusted community members. The official AlchemixDAO is currently under development and will offer an even more decentralized approach to governance of the Alchemix protocol and its contracts.

TBE-01 | No Log In `require()` Check

Category	Severity	Location	Status
Coding Style	Informational	TransmuterB.sol: 303	Acknowledged

Description

No log message is added in the `require()` check. Log is essential message for debugging purpose and tracking the transaction. Adding log to `require()` can also increase the readability and overall quality of the codebase.

Recommendation

We advise the client to add log message to the `require()` check with similar snippet as following:

```
1 require(realisedTokens[sender] > 0, "no realisedToken balance for sender");
```

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TBE-02 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	TransmuterB.sol: 386	Acknowledged

Description

There's a typo in the comment, where `surlus` should be `surplus`

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TBE-03 | Minimize The Scope of Access To The Function

Category	Severity	Location	Status
Control Flow	Informational	TransmuterB.sol: 474	Acknowledged

Description

As the comment indicates in LXX `This function is meant to be called by the Alchemist contract for when it is sending yield to the transmuter.` , the function `distribute()` should only be called by `Alchemist` contract. However, current a whitelist is adopted to restrict the accesses to the `distribute()` function, which may have potential to add non-Alchemist address into it.

Recommendation

We advise the client to stored the `Alchemist` contract addresses in immutable variables and initialized them in the constructor of the `TransmuterB` contract.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TBE-04 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	TransmuterB.sol: 620	Acknowledged

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TBE-05 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	TransmuterB.sol: 799	Acknowledged

Description

The owner of the account with the `governor` role can migrate `migratableFunds` amount of token to another contract which is implemented based upon `ITransmuter` interface. Any compromise to this account may allow the hacker to take advantage of this function and eventually drain the majority tokens from the current `TransmuterB` contract. Although there `s a` pause `mechanism trying to protect any such hacks, if the hacker owns the` governor `role, the hacker can reset the` pause `by calling` setPause() `function, and bypass the` pause `check in the function` migrateFunds()` .

Recommendation

We advise the client to carefully manage the role `governor`'s account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the `governance` role was given to the Alchemix Multisig. The Alchemix Multisig consists of 4 core Alchemix team members and 4 trusted community members. The official AlchemixDAO is currently under development and will offer an even more decentralized approach to governance of the Alchemix protocol and its contracts.

TBE-06 | Centralized Risk To Sensitive `withdraw`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	TransmuterB.sol: 644, 662	**Acknowledged**

Description

The owner of the account with the `governance` role or `sentinel` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example, owner can all `recallAllFundsFromVault()` function or `recallFundsFromVault()` function to recall planted funds from a target vault to address of `TransmuterB` contract Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

We advise the client to carefully manage the role `governor`'s account private key and role `sentinel`'s account private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO / governance/voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the `governance` role was given to the Alchemix Multisig. The Alchemix Multisig consists of 4 core Alchemix team members and 4 trusted community members. The official AlchemixDAO is currently under development and will offer an even more decentralized approach to governance of the Alchemix protocol and its contracts.

TRA-01 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	Transmuter.sol: 277	Acknowledged

Description

There's a typo in the comment, where `surlus` should be `surplus`

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TRA-02 | Minimize The Scope of Access To The Function

Category	Severity	Location	Status
Control Flow	Informational	Transmuter.sol: 363	Acknowledged

Description

As the comment indicates in LXX `This function is meant to be called by the Alchemist contract for when it is sending yield to the transmuter.` , the function `distribute()` should only be called by `Alchemist` contract. However, current a whitelist is adopted to restrict the accesses to the `distribute()` function, which may have potential to add non-Alchemist address into it.

Recommendation

We advise the client to stored the `Alchemist` contract addresses in immutable variables and initialized them in the constructor of the `TransmuterB` contract.

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

TTE-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	TimeToken.sol: 43	**Acknowledged**

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

We advise the client to carefully manage the `MINTER_ROLE` role account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: Immediately after deployment, the MINTER_ROLE role was given to the Alchemix Multisig contract, and revoked from the deploying EOA.

YVA-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	adapters/YearnVaultAdapter.sol: 72	**Acknowledged**

Description

The owner of the account `owner` can withdraw an arbitrary amount of token from vault to an arbitrary address `_recipient` by calling function `withdraw()`

Recommendation

We advise the client to carefully manage the `owner` account's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[Alchemix]`: During deployment, the `admin` role was given to the Alchemist contract. There is no way to transfer the admin role to any other contract or EOA.

YVA-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	adapters/YearnVaultAdapter.sol: 33~34	Acknowledged

Description

The assigned values to `vault` and `admin` in the constructor of `YearnVaultAdapter.sol` should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values. Example:

```
require(address(_vault) != address(0), "_vault address is a zero address");
require(_admin != address(0), "_admin is a zero address");
```

Alleviation

`[Alchemix]`: We will take all the informational findings into account as we develop the protocol further

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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

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 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 without CertiK's prior written consent.

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.

About

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.