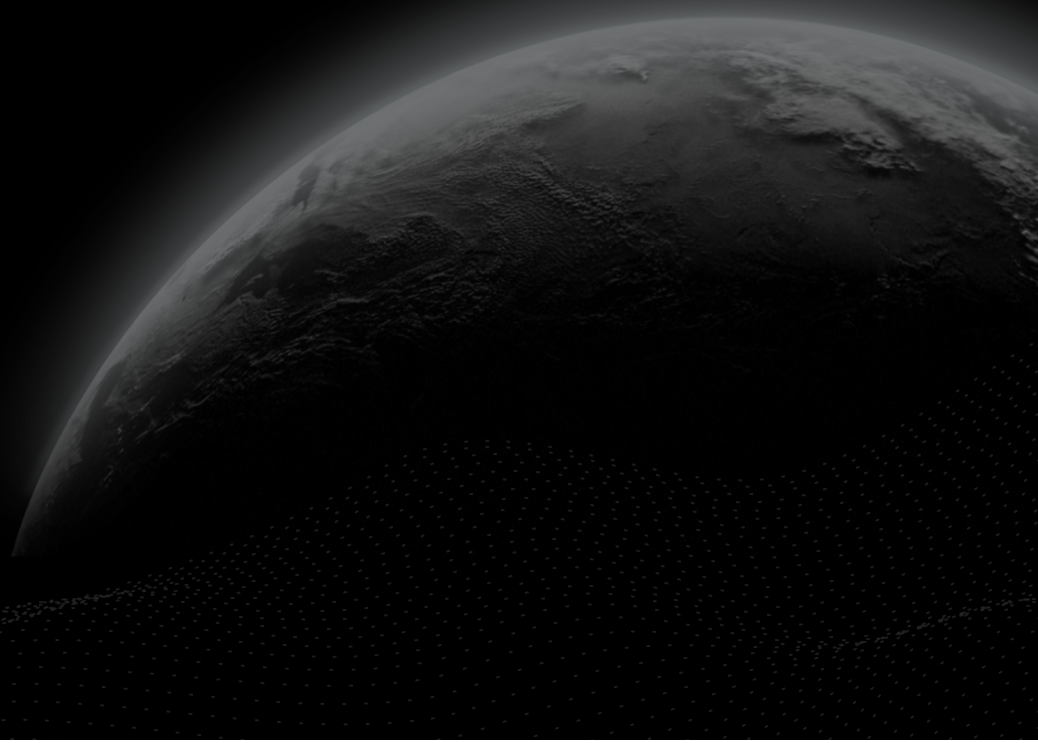
CERTIK

Security Assessment

Formation Finance

CertiK Verified on Sept 2nd, 2021



CertiK Verified on Sept 2nd, 2021

Formation Finance

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Staking	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 09/02/2021	N/A

CODE BASE

github.com/Formation-Finance/stable-vaults/blob/ma...

COMMITS

5e89d9cbb4e4fc55cf5e4dbaf4e89383c568da9f 5d...

https:

Vulnerability Summary



15	6	0	0	9	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

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.
3	Major	3 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
3	Medium	2 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
6	Minor	2 Resolved, 4 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.
3	Informational	2 Resolved, 1 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 | FORMATION FINANCE

Summary

- Executive Summary
- Vulnerability Summary
- Code Base
- Audit Scope
- Approach & Methods

Findings

- FTF-01 : Centralization Risk
- LPT-01 : Centralization Risk
- VFF-07 : Centralization Risk
- FSF-01 : Logical issue of function `withdrawYield()`
- LPF-01 : Logical issue of function `withdrawYield()`
- VFF-11 : Cross chain transaction atomicity
- VFF-03 : Lack of Input Validation
- VFF-04 : Return Value Not Checked
- VFF-05 : Logical issue about `token`
- VFF-06 : Shares could be transferred
- VFF-08 : Unknown implementations
- VFF-09 : Potential insufficient liquidity
- VFF-01 : Address type in event could be indexed
- VFF-02 : Variable `decimals` could be declared in `uint8` type
- VFF-10 : Logical issue in the function `refund`

Appendix

Disclaimer

CODE BASE | FORMATION FINANCE

Repository

https://github.com/Formation-Finance/stable-vaults/blob/master/contracts/Vault.sol

https://github.com/Formation-Finance/form-staking/tree/master/contracts

Commit

5e89d9cbb4e4fc55cf5e4dbaf4e89383c568da9f 5dbee67598b33086acbb2eea5a8efb587118770a

AUDIT SCOPE | FORMATION FINANCE

6 files audited ● 3 files with Acknowledged findings ● 2 files with Resolved findings ● 1 file without findings

ID	File	SHA256 Checksum
FTF	FormToken.sol	5a4c887c20c832876d4ed154ab68b86d5057b6c1b7e71124afb091c7ce7df467
LPT	LPToken.sol	74d84b8caf1bf65b5045ffb7214ef968f0997c519fe81c94ec764bd026ea57e6
VFF	Vault.sol	b70b0dc0e339b8044ed72d67f4efde0331bd6cfd5039d07d7e6ccfb1a0f4d599
FSF	FormStake.sol	aa80876104faca3fee30266b2395d3dc7c692585d330424c06de4df4d419f60e
LPF	LPFarm.sol	573530bbc77b4d148fd5e5262a535d438a2bedfd172caf355536d0596ca5664b
FFE		418abfc6590b1fbe7605798dab3d38d5567454c6b76e251806a59279c1b78239

APPROACH & METHODS | FORMATION FINANCE

This report has been prepared for Formation Finance to discover issues and vulnerabilities in the source code of the Formation Finance 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.

FINDINGS | FORMATION FINANCE



15	0	3	3	6	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Formation Finance . Through this audit, we have uncovered 15 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
FTF-01	**Centralization Risk**	**Centralization / Privilege**	**Major**	**Acknowledged**
LPT-01	**Centralization Risk**	**Centralization / Privilege**	**Major**	**Acknowledged**
VFF-07	**Centralization Risk**	**Centralization / Privilege**	**Major**	**Acknowledged**
FSF-01	Logical Issue Of Function `withdrawYield()`	Logical Issue	Medium	Resolved
LPF-01	Logical Issue Of Function `withdrawYield()`	Logical Issue	Medium	Resolved
VFF-11	Cross Chain Transaction Atomicity	Volatile Code	Medium	Acknowledged
VFF-03	Lack Of Input Validation	Volatile Code	Minor	Resolved
VFF-04	Return Value Not Checked	Logical Issue	Minor	Resolved
VFF-05	Logical Issue About `token`	Logical Issue	Minor	Acknowledged
VFF-06	Shares Could Be Transferred	Logical Issue	Minor	Acknowledged
VFF-08	Unknown Implementations	Volatile Code	Minor	Acknowledged

ID	Title	Category	Severity	Status
VFF-09	Potential Insufficient Liquidity	Logical Issue	Minor	Acknowledged
VFF-01	Address Type In Event Could Be Indexed	Logical Issue	Informational	Resolved
VFF-02	Variable `decimals` Could Be Declared In `uint8` Type	Gas Optimization	Informational	Resolved
VFF-10	Logical Issue In The Function `refund`	Logical Issue	Informational	Acknowledged

FTF-01 | FINDING DETAILS

Finding Title

Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**FormToken.sol (base2): 11**	**Acknowledged**

Description

In the contract `FormToken` and `LPToken`, the role `owner` has the authority over the following function:

- `mint()`, to mint any amount of tokens to any address
- `_transferOwnership()`, to transfer the owner role

Any compromise to the `owner` account may allow the hacker to take advantage of this and mint any amount of tokens to any accounts.

Recommendation

We advise the client to carefully manage the `owner` account's private key to 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 smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 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

The team acknowledged this issue and they stated the following:

This contract is created for testing purposes only. Vault contract will mine LP tokens and FORM tokens are already minted.

LPT-01 | FINDING DETAILS

Finding Title

Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**LPToken.sol (base2): 11**	**Acknowledged**

Description

In the contract `FormToken` and `LPToken`, the role `owner` has the authority over the following function:

- `mint()`, to mint any amount of tokens to any address
- `_transferOwnership()`, to transfer the owner role

Any compromise to the `owner` account may allow the hacker to take advantage of this and mint any amount of tokens to any accounts.

Recommendation

We advise the client to carefully manage the `owner` account's private key to 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 smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 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

The team acknowledged this issue and they stated the following:

This contract is created for testing purposes only. Vault contract will mine LP tokens and FORM tokens are already minted.

VFF-07 | FINDING DETAILS

Finding Title

Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**Vault.sol (base)**	**Acknowledged**

Description

In the contract `Vault`, the role `owner` or `bridge` has the authority over the following function:

- `setEmergencyShutdown()`, to pause `deposit()` or `withdraw()`,
- `withdrawTreasury()`, to transfer `treasuryAmount` to owner
- `swapOut()` and `refund()`, to transfer funds and impact shares.

Any compromise to the `owner` account may allow the hacker to exploit it.

Recommendation

We advise the client to carefully manage the `owner` account's private key to 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 smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 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

The team acknowledged the issue and stated the following:

They need to `swapOut()` and `refund()` using a cloud-based application which is going to be triggered by incoming events. Therefore, they need to have a single wallet capable of calling these functions.

FSF-01 | FINDING DETAILS

Finding Title

Logical Issue Of Function `withdrawYield()`

Category	Severity	Location	Status
Logical Issue	Medium	FormStake.sol (base2)	Resolved

Description

```
92      function withdrawYield() public {
93          uint256 yieldEarned = _calculateYieldTotal(msg.sender);
94          require(yieldEarned > 0, "Nothing to withdraw");
95
96          uint256 transferValue = yieldEarned;
97
98          formToken.transfer(msg.sender, transferValue);
99
100          startTime[msg.sender] = block.timestamp;
101          yieldBalance[msg.sender] = 0;
102
103          emit YieldWithdraw(msg.sender, transferValue);
104      }
```

The function `withdrawYield()` releases temporary yield rewards, but clears the user's `yieldBalance`. This leads to the loss of user rewards.

Recommendation

We recommend checking the logic and fixing the issue.

Alleviation

[Formation Team]:

The team disagreed with this issue and they stated the following:

This is how it should work. Once `yield` is withdrawn, `yieldBalance` of a user is being cleared.

[CertiK Reply]:

The function `_calculateYieldTime()` calculates the elapsed time for updated rewards. Then the function `_calculateYieldTotal()` calculates the updated rewards based on the results returned by the function `_calculateYieldTime()`.

```
function _calculateYieldTime(address user) public view returns(uint256) {
    uint256 end = block.timestamp;
    uint256 totalTime = end - startTime[user];
    return totalTime;
}
```

Then according to the logic in the function `stake()`, we think the `yieldBalance` represents the total accumulated yield for the users.

```
    uint256 yieldEarned = _calculateYieldTotal(msg.sender);
    yieldBalance[msg.sender] += yieldEarned;
```

However, in the functions `unstake/withdrawYield`, only the updated rewards were distributed, the `yieldBalance` is ignored.

Hence we think the `yieldEarnd` does not seem to contain the accumulated rewards.

[Formation Team]:

The team heeded our advice and fixed the issue in commit `45af34f694e29ee64ad2c90803f612390dfa30e8` and `e35d608a2c9992978886484c9d020df01cd46fd0`.

LPF-01 | FINDING DETAILS

Finding Title

Logical Issue Of Function `withdrawYield()`

Category	Severity	Location	Status
Logical Issue	Medium	LPFarm.sol (base2)	Resolved

Description

```
92      function withdrawYield() public {
93          uint256 yieldEarned = _calculateYieldTotal(msg.sender);
94          require(yieldEarned > 0, "Nothing to withdraw");
95
96          uint256 transferValue = yieldEarned;
97
98          formToken.transfer(msg.sender, transferValue);
99
100          startTime[msg.sender] = block.timestamp;
101          yieldBalance[msg.sender] = 0;
102
103          emit YieldWithdraw(msg.sender, transferValue);
104      }
```

The function `withdrawYield()` releases temporary yield rewards, but clears the user's `yieldBalance`. This leads to the loss of user rewards.

Recommendation

We recommend checking the logic and fixing the issue.

Alleviation

[Formation Team]:

The team disagreed with this issue and they stated the following:

This is how it should work. Once `yield` is withdrawn, `yieldBalance` of a user is being cleared.

[CertiK Reply]:

The function `_calculateYieldTime()` calculates the elapsed time for updated rewards. Then the function `_calculateYieldTotal()` calculates the updated rewards based on the results returned by the function `_calculateYieldTime()`.

```
function _calculateYieldTime(address user) public view returns(uint256) {
    uint256 end = block.timestamp;
    uint256 totalTime = end - startTime[user];
    return totalTime;
}
```

Then according to the logic in the function `stake()`, we think the `yieldBalance` represents the total accumulated yield for the users.

```
    uint256 yieldEarned = _calculateYieldTotal(msg.sender);
    yieldBalance[msg.sender] += yieldEarned;
```

However, in the functions `unstake/withdrawYield`, only the updated rewards were distributed, the `yieldBalance` is ignored.

Hence we think the `yieldEarnd` does not seem to contain the accumulated rewards.

[Formation Team]:

The team heeded our advice and fixed the issue in commit `45af34f694e29ee64ad2c90803f612390dfa30e8` and `e35d608a2c9992978886484c9d020df01cd46fd0`.

VFF-11 | FINDING DETAILS

Finding Title

Cross Chain Transaction Atomicity

Category	Severity	Location	Status
Volatile Code	Medium	Vault.sol (base)	Acknowledged

Description

Cross-chain token transfer depends on third party cross-chain protocol, for example, PolyNetwork. If the cross-chain protocol is hacked, the Formation Finance protocol in the source chain cannot confirm the transaction status correctly. As a result, the user deposits tokens in the source chain but cannot receive them in the target chain.

Recommendation

We recommend transferring transaction data by two different cross-chain protocols to ensure the transaction atomicity.

Alleviation

The team acknowledged the issue and stated the following.

They are going to store all of the transactions with statuses in their internal database and if there is some issue where they can not send swapped funds, then they will return the original deposit using refund function. (after 3 retries)

VFF-03 | FINDING DETAILS

Finding Title

Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	Vault.sol (base): 75	Resolved

Description

There is no check if `amount <= token.balanceOf(msg.sender) && amount <= token.allowance(msg.sender, address(this))`.

Recommendation

We recommend adding sanity checks to save gas.

Alleviation

The team resolved the issue in commit 17ca80cf57d71fe47fe498d21220ebcbabbfee54.

VFF-04 | FINDING DETAILS

Finding Title

Return Value Not Checked

Category	Severity	Location	Status
Logical Issue	Minor	Vault.sol (base): 86, 112	Resolved

Description

The return value of invocation 'token.transferFrom()' and `token.transfer()` is not checked.

Recommendation

We recommend checking the retuned result of the function. Using `safetransferFrom` is better.

Alleviation

The team resolved the issue in commit 17ca80cf57d71fe47fe498d21220ebcbabbfee54.

VFF-05 | FINDING DETAILS

Finding Title

Logical Issue About `token`

Category	Severity	Location	Status
Logical Issue	Minor	Vault.sol (base): 86	Acknowledged

Description

Interface `IERC20Metadata` has no function `transferFrom()`.

Refer:https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/IERC20Metadata.sol

Recommendation

We recommend checking the retuned result of the function. Using `safetransferFrom` is better.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

VFF-06 | FINDING DETAILS

Finding Title

Shares Could Be Transferred

Category	Severity	Location	Status
Logical Issue	Minor	Vault.sol (base)	Acknowledged

Description

We notice that the `Vault` contract allow users to transfer shares.
We cannot find the necessity for this feature in current logic. We would like to confirm with the client if this feature would be expanded in subsequent development?

Alleviation

The team acknowledged this issue and they stated the following:

Shares are LP tokens in return of the deposit. Needed for withdrawal to exchange LP tokens for assets.

VFF-08 | FINDING DETAILS

Finding Title

Unknown Implementations

Category	Severity	Location	Status
Volatile Code	Minor	Vault.sol (base)	Acknowledged

Description

There are several unknown implementations in this protocol.

- There are two important functions `swapOut()` and `refund()`, only can be called by the `bridge`. We are not sure the `bridge` is an EOA or a contract.
- In addition, there is no messaging mechanism found for the cross-chain swap process in this contract.

The contract `Vault` is serving as the underlying entity to interact with third party `bridge`. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of this protocol requires interaction with `bridge`. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged the issue and stated the following.

Bridge API is a cloud-based application which intercepts events from the blockchain, matches with database entries (created by the website) and perform a swap on the other chain.

VFF-09 | FINDING DETAILS

Finding Title

Potential Insufficient Liquidity

Category	Severity	Location	Status
Logical Issue	Minor	Vault.sol (base)	Acknowledged

Description

According to the logic of this contract, the liquidity provider uses the function `deposit()` to add liquidity, and uses the function `withdraw()` to remove liquidity. In case the liquidity is insufficient in the vault, the users' cross-chain transactions could be waiting for the liquidity, thus bring the worse user experience.

Recommendation

We recommend to bring more liquidity provider solutions.

Alleviation

The team acknowledged the issue and stated the following.

They will implement mechanisms like max single transaction and max transaction per address per day (off chain) which are going to minimize the ratio differences between vaults, but there might be some unavailability for withdrawals for a short period of time.

VFF-01 | FINDING DETAILS

Finding Title

Address Type In Event Could Be Indexed

Category	Severity	Location	Status
Logical Issue	Informational	Vault.sol (base): 13~15	Resolved

Description

Address type in event declaration could be indexed by adding identifier `indexed` for convenient searching.

Recommendation

We recommend adding keyword `indexed` for address type in event declarations.

Alleviation

The team resolved the issue in commit 17ca80cf57d71fe47fe498d21220ebcbabbfee54.

VFF-02 | FINDING DETAILS

Finding Title

Variable `decimals` Could Be Declared In `uint8` Type

Category	Severity	Location	Status
Gas Optimization	Informational	Vault.sol (base): 19	Resolved

Description

Variable `decimals` could be declared in `uint8` type to save gas.

Alleviation

The team resolved the issue in commit 17ca80cf57d71fe47fe498d21220ebcbabbfee54.

VFF-10 | FINDING DETAILS

Finding Title

Logical Issue In The Function `refund`

Category	Severity	Location	Status
Logical Issue	Informational	Vault.sol (base): 287~302	Acknowledged

Description

In the aforementioned line, the function `refund()` will refund `amount+fee` to the user and redistribute the gas fee to the `treasury`.

According to the logic in function `swapIn()`, the users transfer `amount+fee` into the current contract. We would like to confirm with the client who will pay for the gas fee.

Alleviation

The team acknowledged this issue and they stated the following:

Only Bridge API can trigger refund and gas fee is going to be payed by the team because it is ETH or BNB, and the cost will be returned to the team in the function `refund()` in the form of the IERC20Metadata(token).

APPENDIX | FORMATION FINANCE

Finding Categories

Categories	Description
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.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
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.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.

