Security Assessment

Equalizer V1

Aug 10th, 2021

Table of Contents

Summary

This report has been prepared for Equalizer Finance to discover issues and vulnerabilities in the source code of the Equalizer V1 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- 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.

Overview

Project Summary

Project Name	Equalizer V1
Platform	Ethereum, BSC
Language	Solidity
Codebase	https://github.com/Equalizer-Finance/equalizer-smart-contracts-v1-private
Commit	5c841547852215b6c64c1052266b9ea37655c0b4 649313bfecf562582e0973187b1066c06fad01d7 e8b6395dceb9d17825b175220a41683161834691 3f8c763a87077885f2b3a840d1d48562b29a8d17 bc5e29f192c08c43ce9b123ca709cf3396aca1d3

Audit Summary

Delivery Date	Aug 10, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	1	0	0	1	0	0
Major	1	0	0	1	0	0
Medium	1	0	1	0	0	0
Minor	3	0	1	0	2	0
Informational	4	0	0	4	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IER	interfaces/IERC20.sol	139c69be728d147c65269bcc07277f673f10e29a4ca2c96d2cd4c9b111fe3ab0
IEC	interfaces/IERC3156FlashBorrower.sol	ca9f8c32d37644421d1c09efcfc5446a2a754b4d60da93c6034cdc1aca05f663
IEF	interfaces/IERC3156FlashLender.sol	acaa1365c881d807e28ee24f361b25d8524f6d510243c61745ae4569d465bec7
IFL	interfaces/IFlashLoanFeeProvider.sol	e8f3572ccff5f710a7310f90da9ca9e91242c2fe50f563755a2a3ec195a10143
IVC	interfaces/IVault.sol	57e169720e5d3bd11b4bccf4a8e6eb03f1bc18d46934c43b91ef35fde6250952
IVF	interfaces/IVaultFactory.sol	74aea3ec0548aaea32d3e0979d550548346f97269af0f0cfab9daa00cb7a8450
MCK	roles/Moderable.sol	1c870ed692c3e15759021c54bf18c81c58ceb311540bd1faa6c19105af4d1541
CCC	CoreConstants.sol	fa661971fc3f47e0e8ffe1f0435cd81c7319d8cd87f1ad72ba6f19eca2eb5a2c
ERC	ERC20EToken.sol	13c5413c4412bb969f55e71eb9c4f2ab7bfb7c13251d3d4690de701f8a619b65
FLF	FlashLoanFeeProvider.sol	924e3d1a413bcf32b8bf3a6fc4fd930981c8957516b69b4dd505a16c55ed83f7
FLP	FlashLoanProvider.sol	4ef65fa032b82a2cca80f34150558dd48dbac36b7164c9dc18ac0808aa8e093d
VCK	Vault.sol	862e35b9edb216486a85212113e9851e0b1e3bf943eccadf89bd153942f7b92f
VFC	VaultFactory.sol	cf6abd25a6c9493d3bac19ce5b375451f4694e891d37a9bcd2b8cd665ae18924

Review Notes

Dependencies

There are a few depending injection contracts or addresses in the current project:

- `stakedToken`, `treasuryAddress` for the contract `Vault`;
- `ERC20(token)`, `receiver` for the contract `FlashLoanProvider`;
- `tokenToPayInFee`, `stakedToken` for the contract `VaultFactory`.

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

Privilledged Functions

The `moderator` role in the contract `Vault` can operate on the following functions:

- `Vault.setMaxCapacity()` to change the maximum capacity of a vault;
- `Vault.setMinAmountForFlash()` to change the minimum amount for flash loan;
- `Vault.pauseVault()` to pause a vault;
- `Vault.unpauseVault()` to unpause a vault.

The `moderator` role in the contract `VaultFactory` can operate on the following functions:

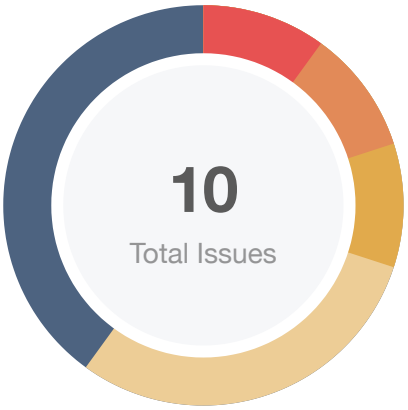
- `VaultFactory.setTreasuryAddress()` to change treasury address;
- `VaultFactory.setFeeToPublishVault()` to change fee for publishing a vault;
- `VaultFactory.createVault()` to create a vault as a moderator;
- `VaultFactory.withdraw()` to withdraw funds which are payed as tax for listing vaults.

The `moderator` role in the contract `FlashLoanFeeProvider` can operate on the following functions:

- `FlashLoanFeeProvider.setFee()` to set the flash loan fee rate;
- `FlashLoanFeeProvider.setTreasuryFeePercentage` to set the treasury fee percentage.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings



Critical	**1** (10.00%)
Major	**1** (10.00%)
Medium	**1** (10.00%)
Minor	**3** (30.00%)
Informational	**4** (40.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
FLF-01	Centralization Risks	**Centralization / Privilege**	**Minor**	**Partially Resolved**
FLF-02	Lack of Input Validation	Coding Style	Informational	Resolved
FLP-01	Variables Could Be Declared as `immutable`	Gas Optimization, Language Specific	Informational	Resolved
VCK-01	Incorrect Amount Calculation	Logical Issue	Critical	Resolved
VCK-02	Undistributed Fees	Logical Issue	Major	Resolved
VCK-03	Centralization Risks	**Centralization / Privilege**	**Minor**	**Acknowledged**
VCK-04	Incompatibility with Deflationary Tokens	Logical Issue	Minor	Acknowledged
VCK-05	Redundant Code	Logical Issue	Informational	Resolved
VFC-01	Centralization Risks	**Centralization / Privilege**	**Medium**	**Partially Resolved**
VFC-02	Lack of Event Emissions for Significant Transactions	**Centralization / Privilege**	**Informational**	**Resolved**

FLF-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	FlashLoanFeeProvider.sol: 18, 37	**Partially Resolved**

Description

The `moderator` is an important role in the contract `FlashLoanFeeProvider`. The `moderator` can operate on the following functions:

- `FlashLoanFeeProvider.setFee()` to change flash loan fee related state variables;
- `FlashLoanFeeProvider.setTreasuryFeePercentage()` to change treasury fee percentage.

Recommendation

We recommend the client carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt the `Timelock` contract with a reasonable delay to allow users to withdraw their funds, Multisig with community-selected 3-party independent co-signer, and/or DAO with transparent governance with the project's community in the project to manage the sensitive role accesses.

Alleviation

[Equalizer Team]: The issue will be resolved with the governance model and the governance smart contracts that will be put in place after the V1 release. In the first version, all critical operations are managed using a multi-sig wallet (Gnosis safe), requiring 4-out-of-5 signatures by the Equalizer core team members, taking into consideration the off-chain voting of the community.

[08/10/2021]: The team provided references for multi-sig address as below:

Gnosis safe for the Equalizer Vaults (Treasury)
https://etherscan.io/address/0xA49174859aA91E139b586F08BbB69BceE847d8a7

FLF-02 | Lack of Input Validation

Category	Severity	Location	Status
Coding Style	Informational	FlashLoanFeeProvider.sol: 19~20, 38	Resolved

Description

The inputs `_flashFeePercentage` and `_flashFeeAmountDivider` in the function `FlashLoanFeeProvider.setFee()`, and `_treasuryFeePercentage` in the function `FlashLoanFeeProvider.setTreasuryFeePercentage()` should have proper input validations in case of unwanted mis-inputs.

Recommendation

We recommend adding necessary precautions to check the validity of the inputs, `_flashFeePercentage` and `_flashFeeAmountDivider` in the function `FlashLoanFeeProvider.setFee()`, and the maximum capacity `_treasuryFeePercentage` in the function `FlashLoanFeeProvider.setTreasuryFeePercentage()`.

Alleviation

The team heeded the advice and resolved this issue in the commit `3f8c763a87077885f2b3a840d1d48562b29a8d17`.

FLP-01 | Variables Could Be Declared as `immutable`

Category	Severity	Location	Status
Gas Optimization, Language Specific	Informational	FlashLoanProvider.sol: 15	Resolved

Description

State variable, `vaultFactory`, that never changed after constructor can be declared as `immutable`.

Recommendation

We recommend declaring the aforementioned variable as `immutable`.

Alleviation

The team heeded the advice and resolved this issue in the commit `649313bfecf562582e0973187b1066c06fad01d7`.

VCK-01 | Incorrect Amount Calculation

Category	Severity	Location	Status
Logical Issue	Critical	Vault.sol: 118	Resolved

Description

According to the logic implemented in the function `Vault.provideLiquidity()`, the function `Valut.getNrOfETokensToMint()` should calculate the amount of eToken to be minted for the liquidity provider given the amount of liquidity token deposited as follows:

$$(\text{amount deposited}) \cdot \frac{(\text{total eToken})}{(\text{total deposited})}$$

Considering the function `Vault.getRatio()` returns the ratio of (total deposited) / (total eTokens), the implementation of `Valut.getNrOfETokensToMint()` is incorrect:

```
118     function getNrOfETokensToMint(uint256 amount) internal view returns (uint256) {
119         return (amount * getRatio()) / RATIO_MULTIPLY_FACTOR;
120     }
```

Recommendation

We recommend revising the calculation formula in the aforementioned function to accommodate the correct logic.

Alleviation

The team heeded the advice and resolved this issue in the commit `e8b6395dceb9d17825b175220a41683161834691`.

VCK-02 | Undistributed Fees

Category	Severity	Location	Status
Logical Issue	Major	Vault.sol: 126, 155, 203, 218, 240	Resolved

Description

The function `FlashLoanProvider.flashLoan()` implemented in the file FlashLoanProvider.sol borrows some amount of the token from the contract `Vault` and returns borrowed amount plus fees to `Vault`. After fees are transferred to `Vault`, 20% of fees are transferred to `Vault.treasuryAddress` by `Vault.splitFees`:

```
    function flashLoan(
        IERC3156FlashBorrower receiver,
        address token,
        uint256 amount,
        bytes calldata data
    ) external override returns (bool) {
        ...
        require(vault.transferToAccount(address(receiver), amount),
'FLASH_LENDER_TRANSFER_FAILED');
        ...
        require(
            vault.transferFromAccount(address(receiver), amount + fee),
            'FLASH_LENDER_REPAY_FAILED'
        );
        vault.splitFees(fee);
        ...
    }
```

However, the rest (80%) of fees are not handled and will be stuck in the contract account.

Moreover, `totalAmountDeposited` is only updated through the functions `Vault.provideLiquidity()` and `Vault.removeLiquidity()`, so the function `Vault.getRatio()` would always return a constant value, which means users can only withdraw at most the same amount of liquidity they provided to `Vault`. They will not be benefitted from flash loan fees.

We hope to check with the Equalizer team and confirm if this is the intended design.

Alleviation

[Equalizer Team]: The fees will be distributed to users automatically when the price of eToken will increase. The price of eToken increases when fees are accumulated in the vault (from flash loans service).

The price of eToken will always increase, it never goes down. In the worst case, it can be constant in case fees are not generated with the respective vault. The fixed version of `getRatio()` function also takes the fees into account. It is the same problem as VCK-01.

VCK-03 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	Vault.sol: 102, 110, 184, 192	**Acknowledged**

Description

The `moderator` is an important role in the contract `Vault`. The `moderator` can operate on the following functions:

- `Vault.setMaxCapacity()` to change the maximum capacity of a vault;
- `Vault.setMinAmountForFlash()` to change the minimum amount for flash loan;
- `Vault.pauseVault()` to pause a vault;
- `Vault.unpauseVault()` to unpause a vault.

Recommendation

We recommend the client carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt the `Timelock` contract with a reasonable delay to allow users to withdraw their funds, Multisig with community-selected 3-party independent co-signer, and/or DAO with transparent governance with the project's community in the project to manage the sensitive role accesses.

Alleviation

[Equalizer Team]: The issue will be resolved with the governance model and the governance smart contracts that will be put in place after the V1 release. In the first version, all critical operations are managed using a multi-sig wallet (Gnosis safe), requiring 4-out-of-5 signatures by the Equalizer core team members, taking into consideration the off-chain voting of the community.

VCK-04 | Incompatibility with Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	Vault.sol: 126	Acknowledged

Description

The contract `Vault` operates as the main entry for the interaction for users. Users can deposit `stakedToken` to provide liquidity to `Vault`. Also, users can withdraw their assets (remove liquidity) from the vault. In this process, `Vault.provideLiquidity()` and `Vault.removeLiquidity()` may be involved in transferring users' assets into or out of the Equalizer protocol.

When transferring deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistency.

Recommendation

We recommend keeping regulating the set of tokens supported by the Equalizer Protocol, and if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

Alleviation

[Equalizer Team]: We don't support deflationary tokens and we won't list such tokens in the V1. We will evaluate the possibility for supporting deflationary tokens in V2. In the V1, the listing of the new vaults will be done only by the core team, after individual and manual validation of each token.

In case a Deflationary Token will be listed by mistake, the core team has the possibility to Inactivate the vault.

VCK-05 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	Vault.sol: 142~144	Resolved

Description

In the code snippet below, `lastDepositBlockNr[msg.sender]` is planned to be updated only if `msg.sender` hasn't registered in `lastDepositBlockNr`.

```
142         if (lastDepositBlockNr[msg.sender] == 0) {
143             lastDepositBlockNr[msg.sender] = block.number;
144         }
```

However, On Line 148, `lastDepositBlockNr[msg.sender]` is assigned to be the `block.number` on both conditions.

Recommendation

We recommend removing the redundant code on Line 142-144, or fully implementing the logic on the aforementioned lines.

Alleviation

The team heeded the advice and resolved this issue by removing the redundant code in the commit `649313bfecf562582e0973187b1066c06fad01d7`.

VFC-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	VaultFactory.sol: 39, 64, 73, 121	**Partially Resolved**

Description

The `moderator` is an important role in the contract `VaultFactory`. The `moderator` can operate on the following functions:

- `VaultFactory.setTreasuryAddress()` to change treasury address;
- `VaultFactory.setFeeToPublishVault()` to change fee for publishing a vault;
- `VaultFactory.createVault()` to create a vault as a moderator;
- `VaultFactory.withdraw()` to withdraw funds that are payed as tax for listing vaults.

Recommendation

We recommend the client carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt the `Timelock` contract with a reasonable delay to allow users to withdraw their funds, Multisig with community-selected 3-party independent co-signer, and/or DAO with transparent governance with the project's community in the project to manage the sensitive role accesses.

Alleviation

[Equalizer Team]: The issue will be resolved with the governance model and the governance smart contracts that will be put in place after the V1 release. In the first version, all critical operations are managed using a multi-sig wallet (Gnosis safe), requiring 4-out-of-5 signatures by the Equalizer core team members, taking into consideration the off-chain voting of the community.

[08/10/2021]: The team provided more address references as below:

Gnosis safe for the Equalizer Vaults (Treasury)
https://etherscan.io/address/0xA49174859aA91E139b586F08BbB69BceE847d8a7

VFC-02 | Lack of Event Emissions for Significant Transactions

Category	Severity	Location	Status
Centralization / Privilege	**Informational**	VaultFactory.sol: 39, 64	**Resolved**

Description

The functions that affect the status of sensitive variables should be able to emit events as notifications to the users. For example,

- `VaultFactory.setTreasuryAddress()` to change treasury address;
- `VaultFactory.setFeeToPublishVault()` to change fee for publishing a vault.

Recommendation

We recommend emitting events for all the essential state variables that are possible to be changed during the runtime.

Alleviation

The team heeded the advice and resolved this issue in the commit `649313bfecf562582e0973187b1066c06fad01d7`.

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.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

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 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.

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.