CERTIK

Security Assessment

FSTSWAP (Farm.sol)

Apr 28th, 2022

Table of Contents

Summary

This report has been prepared for FSTSWAP (Farm.sol) to discover issues and vulnerabilities in the source code of the FSTSWAP (Farm.sol) 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	FSTSWAP (Farm.sol)
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/FstSwapDex/contract_farm
Commit	eb15fb1d08ae8e75b2d9a9be5f2679333add9f74

Audit Summary

Delivery Date	Apr 28, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

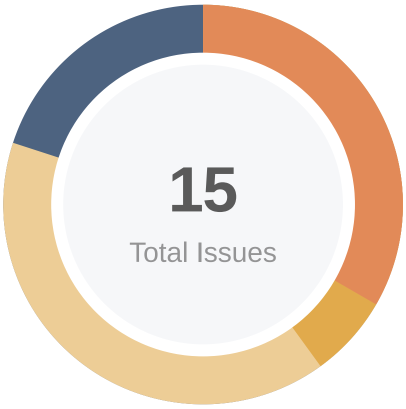
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	5	0	0	2	0	0	3
● Medium	1	0	0	1	0	0	0
● Minor	6	0	0	1	0	0	5
● Informational	3	0	0	1	0	0	2
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
FFS	Farm.slo	fd0b8ad107a0d9f9d475d41c720d589cde20d15cd7eead79ea04f5f23c7f4d57

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**5** (33.33%)
🟨 **Medium**	**1** (6.67%)
▨ **Minor**	**6** (40.00%)
🟦 **Informational**	**3** (20.00%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
FFS-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
FFS-02	Delegation Not Moved Along With Transfer	Logical Issue	● Major	✓ Resolved
FFS-03	Initial Token Distribution	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
FFS-04	Incorrect Delegation Flow	Logical Issue	● Major	✓ Resolved
FFS-05	Logic Flaw In `emergencyWithdraw()`	Logical Issue	● Major	✓ Resolved
FFS-06	Uncertain Income Source Of Reward Token	Logical Issue	● Medium	ⓘ Acknowledged
FFS-07	Incompatibility With Deflationary Tokens(Farming)	Volatile Code	● Minor	ⓘ Acknowledged
FFS-08	`add()` Function Not Restricted	Logical Issue	● Minor	✓ Resolved
FFS-09	Recommended Explicit Pool Validity Checks	Logical Issue	● Minor	✓ Resolved
FFS-10	Missing Update Pools	Logical Issue	● Minor	✓ Resolved
FFS-11	Check Effect Interaction Pattern Violated	Logical Issue	● Minor	✓ Resolved
FFS-12	Over-transferred Tokens	Logical Issue	● Minor	✓ Resolved
FFS-13	Public Function That Could Be Declared External	Gas Optimization	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
FFS-14	Missing Emit Events	Coding Style	● Informational	✓ Resolved
FFS-15	Inconsistent Comments And Code	Coding Style	● Informational	✓ Resolved

FFS-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	Farm.slo	ⓘ Acknowledged

Description

In the contract `Ownable`, the role `owner` has authority over the following functions:

- function renounceOwnership()
- function transferOwnership(address newOwner)

In the contract `BEP20`, the role `owner` has authority over the following functions:

- function mint(uint256 amount)

In the contract `FarmReward`, the role `owner` has authority over the following functions:

- function mint(address _to, uint256 _amount)
- function burn(address _from ,uint256 _amount)
- function safeFonvityTransfer(address _to, uint256 _amount)

In the contract `Farm`, the role `owner` has authority over the following functions:

- function updateMultiplier(uint256 multiplierNumber)
- function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate)
- function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate)

In the contract `Farm`, the role `daoaddr` has authority over the following functions:

- function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate)

Any compromise to these accounts may allow a hacker to take advantage of this authority.

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., multi-signature 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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Team]: Issue acknowledged. We won't make any changes for the current version.

FFS-02 | Delegation Not Moved Along With Transfer

Category	Severity	Location	Status
Logical Issue	● Major	Farm.slo: 955, 1211	✓ Resolved

Description

The voting power of delegation is not moved from token sender to token recipient along with the `transfer()` and `transferFrom()`. Current `transfer()` and `transferFrom()`are from `BEP20` protocol and don't invoke `_moveDelegates()`.

Recommendation

We advise the client to consider moving delegation along with these functions. For example, override `transfer()`/`transferFrom()` in `FON` like `mint()`, and override `transfer()`/`transferFrom()` in `FarmReward` like `mint()/burn()`.

```
function transfer(address recipient, uint256 amount) public override returns (bool) {
    super.transfer(recipient, amount);
    _moveDelegates(_delegates[_msgSender()], _delegates[recipient], amount);
    return true;
}
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public override returns (bool) {
    super.transferFrom(sender, recipient, amount);
    _moveDelegates(_delegates[sender], _delegates[recipient], amount);
    return true;
}
```

Reference: https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAM.sol#L108

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-03 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	Farm.slo: 968~977	ⓘ Acknowledged

Description

All of the `FON` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute all tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[Team]: We'll change the owner to `FarmReward` contract after `FarmReward` contract is deployed.

FFS-04 | Incorrect Delegation Flow

Category	Severity	Location	Status
Logical Issue	● Major	Farm.slo: 1228~1229	✓ Resolved

Description

Whenever new FRT tokens are minted, new delegates are moved from the zero address to the recipient of the minting process. However, whenever tokens are burned, new delegates are once again moved from the zero address to the recipient whereas delegates should be moved in the opposite way.

Recommendation

We advise that the address(0) and _from variable orders are swapped on L1228 to alleviate this issue. At its current state, it breaks the delegate mechanism and can also lead to a user being unable to mint/burn tokens in case the upper limit of a uint256 is reached due to the SafeMath utilization on L1430.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-05 | Logic Flaw In `emergencyWithdraw()`

Category	Severity	Location	Status
Logical Issue	● Major	Farm.slo: 1726	✓ Resolved

Description

When `msg.sender` calls `enterStaking()`, `FRT` token will be minted to `msg.sender` when `pool.lpToken` is staked in the contract. However, if the `msg.sender` calls `emergencyWithdraw()`, the `pool.lpToken` can be transferred back to the `msg.sender` but the `FRT` token that has been minted to the `msg.sender` will not be burnt. Therefore, `msg.sender` can call `enterStaking()` and `emergencyWithdraw()` repeatedly to ultimately mint a huge amount of FRT token, with just the same amount of pool.lpToken

Recommendation

We advise the client to burn the same amount of `FRT` along with the withdraw of `pool.lpToken` when calling the `emergencyWithdraw()`. i.e:

```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    if(_pid == 0) {
        frt.burn(msg.sender, user.amount);
    }
    uint256 amount = user.amount;
    user.amount = 0;
    user.rewardDebt = 0;
    pool.lpToken.safeTransfer(address(msg.sender), amount);
    emit EmergencyWithdraw(msg.sender, _pid, amount);
}
```

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-06 | Uncertain Income Source Of Reward Token

Category	Severity	Location	Status
Logical Issue	● Medium	Farm.slo: 1600, 1623	ⓘ Acknowledged

Description

The rewards tokens are all sent from contract `FarmReward`, so the users may not get the full amount of rewards when the balance in this contract is insufficient.

Recommendation

We advise the client to ensure the reward token is enough for all users.

Alleviation

[Team] The reward token will store at `FarmReward` contract after it is deployed.

FFS-07 | Incompatibility With Deflationary Tokens(Farming)

Category	Severity	Location	Status
Volatile Code	● Minor	Farm.slo	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user stakes 100 deflationary tokens (with a 10% transaction fee) in a MasterChef, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

The MasterChef takes the pool token balance(the `lpSupply`) into account when calculating the users' reward. An attacker can repeat the process of deposit and withdraw to lower the token balance(`lpSupply`) in a deflationary token pool and cause the contract to increase the reward amount.

Reference: https://thoreum-finance.medium.com/what-exploit-happened-today-for-gocerberus-and-garuda-also-for-lokum-ybear-piggy-caramelswap-3943ee23a39f

Recommendation

We advise the client to regulate the set of pool tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Team]: We will check the token is standard ERC20 before adding.

FFS-08 | `add()` Function Not Restricted

Category	Severity	Location	Status
Logical Issue	● Minor	Farm.slo: 1578	✓ Resolved

Description

When the same LP token is added into a pool more than once in function `add()`, the total amount of reward in function `updatePool()` will be incorrectly calculated. The current implementation is relying on the operation correctness to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

We recommend adding the check for ensuring whether the given pool for addition is a duplicate of an existing pool so that the pool addition is only successful when there is no duplicate. This can be done by using a mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-09 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	● Minor	Farm.slo	✓ Resolved

Description

There's no sanity check to validate if a pool is existing.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `set()`, `pendingFonvity()`, `updatePool()`, `deposit()`, `withdraw()` and `emergencyWithdraw()`.

```
modifier validatePoolByPid(uint256 _pid) {
    require (_pid < poolInfo.length , "Pool does not exist") ;
    _;
}
```

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-10 | Missing Update Pools

Category	Severity	Location	Status
Logical Issue	● Minor	Farm.slo: 1560	✓ Resolved

Description

When updating `BONUS_MULTIPLIER`, the reward for each block will change, the interval for which the reward is not calculated before the update should still be calculated based on the old reward for each block.

Recommendation

We advise the client to update the pools when updating `BONUS_MULTIPLIER`.

```
function updateMultiplier(uint256 multiplierNumber) public onlyOwner {
    massUpdatePools();
    BONUS_MULTIPLIER = multiplierNumber;
}
```

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-11 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	● Minor	Farm.slo	✓ Resolved

Description

In functions deposit()/withdraw()/enterStaking()/leaveStaking()/emergencyWithdraw() of the contract, the Checks Effects Interaction Pattern is not strictly followed. Using interfaces, the implementation of safeTransfer or safeTransferFrom are unknown and may have a malicious logical implementation that calls back to the function deposit(). This is dangerous for the calculation for example the user's balance, the pool's totalAmount, etc.

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 avoid reentrancy and potential assets lost.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-12 | Over-transferred Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	Farm.slo: 1636~1637	✓ Resolved

Description

`updatePool()` function transfers an additional reward about 17.6% to `devaddr`.

Recommendation

We advise the client to fix the block reward as 100% instead of about 117.6%.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-13 | Public Function That Could Be Declared External

Category	Severity	Location	Status
Gas Optimization	● Informational	Farm.slo	ⓘ Acknowledged

Description

Following public functions that are never called by the contract internally should be declared with `external` visibility to save gas.

contract: `BEP20`

- `transfer()`
- `approve()`
- `transferFrom()`
- `increaseAllowance()`
- `decreaseAllowance()`

contract: `FON`

- `mint()` in the contract

contract: `FarmReward`

- `mint()`
- `burn()`
- `safeFonvityTransfer()`

contract: `Farm`

- `updateMultiplier()`
- `add()`
- `set()`
- `deposit()`
- `withdraw()`
- `enterStaking()`
- `leaveStaking()`
- `emergencyWithdraw()`
- `dev()`

- `dao()`

Recommendation

We advise using the `external` attribute for the visibility of the listed functions as they are never called from the contract internally.

Alleviation

[Team]: Issue acknowledged. We won't make any changes for the current version.

FFS-14 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	Farm.slo: 1560	✓ Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `updateMultiplier()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

FFS-15 | Inconsistent Comments And Code

Category	Severity	Location	Status
Coding Style	● Informational	Farm.slo: 1635~1636	✓ Resolved

Description

```
  1635          // devaddr got 15%
  1636          frt.safeFonvityTransfer(devaddr,
fonvityReward.mul(17647058823529413).div(1e17));
```

Referring to line 1635 comments, the `devaddr` fee is 15%. But currently, the fee is about 17.64%.

Recommendation

We advise the client to double-check this to improve the code readability.

Alleviation

The development team resolved this issue in commit `326c71809aab73c28d944d28c9725d0d414b1a1f`.

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.

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.

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.

CERTIK