Code Security Assessment

Swell Network

Mar 11th, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- CKP-01 : Missing Zero Address Validation
- CKP-02 : Centralization Related Risks
- CKP-03 : Variables That Could Be Declared as Immutable
- NFT-01 : Third Party Dependencies
- NFT-02 : Missing Emit Events
- NFT-03 : Incorrect `require` Statement
- NFT-04 : Potential Gas Exhaustion
- NFT-05 : Check Effect Interaction Pattern Violated
- NFT-06 : No Function to Withdraw Funds
- NFT-07 : Withdraw Tokens Without Reward
- NFT-08 : User Can Deposit More Tokens than The Position Value
- NFU-01 : Potential Gas Exhaustion On function `stake`
- NFU-02 : Lack of Validation for `msg.value`
- SCK-01 : Missing Error Messages

Appendix

Disclaimer

About

Summary

This report has been prepared for Swell Network to discover issues and vulnerabilities in the source code of the Swell Network 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	Swell Network
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/SwellNetwork/v2-core
Commit	① 6bb73be7ebc7bb8b8ca79bff61159aab66ce356f ② 1b67128b5c0b9904bebf596c32fa5040d72eb10f

Audit Summary

Delivery Date	Mar 11, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

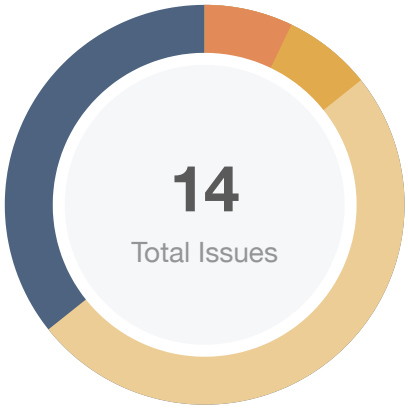
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	1	0	0	0
Medium	1	0	0	1	0	0	0
Minor	7	0	0	2	0	0	5
Informational	5	0	0	3	0	0	2
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SCK	contracts/Strategy.sol	ca8fb2be2c83358351b75f7aaa1643ff356bbed7079cc77f1eea299d81133900
CKP	contracts/helpers.sol	0dc62ddc286b43e6c2e847631f0ae66489fab849557ed3fa77bab1c611b342e7
DAO	contracts/swDAO.sol	3acf343795b9a55785584f305761cdb1c0f929eccb217491bd0f1da071ae3819
ETH	contracts/swETH.sol	52ac6ce60108428af75b121967b979528c2109e70a5c7c7c3dde9b1a04f89ba9
NFT	contracts/swNFTUpgrade.sol	b5b1a71acde5a17bbb3d8bb7f99c550d1820fa3cd9ee17b3e0e46ef575f96d21

Findings



Critical	**0** (0.00%)
Major	**1** (7.14%)
Medium	**1** (7.14%)
Minor	**7** (50.00%)
Informational	**5** (35.71%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Missing Zero Address Validation	Volatile Code	Minor	Resolved
CKP-02	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
CKP-03	Variables That Could Be Declared as Immutable	Gas Optimization	Minor	Resolved
NFT-01	Third Party Dependencies	Volatile Code	Informational	Acknowledged
NFT-02	Missing Emit Events	Coding Style	Informational	Resolved
NFT-03	Incorrect `require` Statement	Logical Issue	Minor	Resolved
NFT-04	Potential Gas Exhaustion	Volatile Code	Minor	Acknowledged
NFT-05	Check Effect Interaction Pattern Violated	Logical Issue	Minor	Resolved
NFT-06	No Function to Withdraw Funds	Logical Issue	Informational	Acknowledged
NFT-07	Withdraw Tokens Without Reward	Logical Issue	Informational	Acknowledged
NFT-08	User Can Deposit More Tokens than The Position Value	Logical Issue	Medium	Acknowledged
NFU-01	Potential Gas Exhaustion On function `stake`	Gas Optimization	Minor	Acknowledged
NFU-02	Lack of Validation for `msg.value`	Logical Issue	Minor	Resolved
SCK-01	Missing Error Messages	Coding Style	Informational	Resolved

CKP-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/Strategy.sol: 16 contracts/swETH.sol: 15	Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

Recommend adding zero address checks in the functions.

Alleviation

The team heeded our advice and resolved this issue in commit `95b31eea33ef584a029f50508cea0509e58a7ae1`.

CKP-02 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/Strategy.sol: 25, 33 contracts/swDAO.sol: 15, 19 contracts/swETH.sol: 24, 28 contracts/swNFTUpgrade.sol: 89, 96, 104, 115, 115	Acknowledged

Description

In the contract `Strategy`, the role `swNFT` has authority over the following functions:

- function enter(): to deposit `swETH` tokens to the `Strategy` contract.
- function exit(): to withdraw `swETH` tokens from the `Strategy` contract.

Any compromise to the `swNFT` account may allow a hacker to take advantage of this authority.

In the contract `SWDAO`, the role `_owner` has authority over the following functions:

- function burn(): to burn the `swDAO` tokens from the owner address.
- function mint(): to mint `swDAO` tokens to the owner address.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority.

In the contract `SWETH`, the role `minter` has authority over the following functions:

- function mint(): to mint `swETH` tokens to the minter address.
- function burn(): to burn `swETH` tokens from the minter address.

Any compromise to the `minter` account may allow a hacker to take advantage of this authority.

In the contract `SWNFTUpgrade`, the role `_owner` has authority over the following functions:

- function setswETHAddress(): to set the `swETH` token address.
- function addStrategy(): to add a new strategy.
- function removeStrategy(): to remove a strategy.
- function addWhiteList(): to add a new validator into whiteList.
- funciton setFeePool(): to set fee pool address.

Any compromise to the `_owner` account 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

The team acknowledged the issue and stated they will transfer the owner to Protocol DAO Gnosis multisign.

CKP-03 | Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	Minor	contracts/swETH.sol: 11 contracts/Strategy.sol: 14	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The team heeded our advice and resolved this issue in commit `95b31eea33ef584a029f50508cea0509e58a7ae1`.

NFT-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Informational	contracts/swNFTUpgrade.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `IDepositContract` protocols. The scope of the audit treats 3rd party entities as black boxes and assumes 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 the contract requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and they stated the following:

"As IDepositContract is the official deposit contract, which is static code and address that is not going to change. And worst case as swNFT is an upgradable contract that we could still update from there".

NFT-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/swNFTUpgrade.sol: 89	Resolved

Description

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

Recommendation

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

Alleviation

The team heeded our advice and resolved this issue in commits `95b31eea33ef584a029f50508cea0509e58a7ae1` and `bc2756fb84a680c4630f0e8c1bb73f0d5c550b91`.

NFT-03 | Incorrect `require` Statement

Category	Severity	Location	Status
Logical Issue	Minor	contracts/swNFTUpgrade.sol: 105, 208, 231	Resolved

Description

The correct require check should ensure strategy index is in the range of the array `strategies` length.

Recommendation

We advise the client to recheck the function.

Alleviation

The team heeded our advice and resolved this issue in commit `bc2756fb84a680c4630f0e8c1bb73f0d5c550b91`.

NFT-04 | Potential Gas Exhaustion

Category	Severity	Location	Status
Volatile Code	Minor	contracts/swNFTUpgrade.sol: 246	Acknowledged

Description

The `for` loop in the function `batchAction()` takes the unbounded array's length as the maximum iteration times. If the size of the array `Action` grows large, iterating through the entire array could be an expensive operation considering there are external calls in the `for` loop. Even worse, if the computation for each iteration is extremely complex, it could lead to gas insufficiency.

Recommendation

We recommend setting constraints to the length of the array `Action`.

Alleviation

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

NFT-05 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Minor	contracts/swNFTUpgrade.sol: 173, 190, 206, 229	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

It is recommended to follow checks-effects-interactions pattern for cases like this. It shields public functions from re-entrancy attacks. It's always a good practice to follow this pattern. `checks-effects-interaction` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Alleviation

The team heeded our advice and resolved this issue in commit `1b67128b5c0b9904bebf596c32fa5040d72eb10f`.

NFT-06 | No Function To Withdraw Funds

Category	Severity	Location	Status
Logical Issue	Informational	contracts/swNFTUpgrade.sol: 143~149	Acknowledged

Description

In the function `stake()`, ETH is deposited to the contract `IDepositContract`, however, there is no function provided in contract `SWNFTUpgrade` to withdraw ETHs, we would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We advise the client to revisit the design and ensure it is intended.

Alleviation

The team acknowledged this issue and they stated that is intended and that's how Ethereum official POS works.

NFT-07 | Withdraw Tokens Without Reward

Category	Severity	Location	Status
Logical Issue	Informational	contracts/swNFTUpgrade.sol: 190, 229	Acknowledged

Description

In the contract `SWNFTUpgrade`, the users can deposit `swETH` to the contract, however, they do not receive any reward when withdrawing `swETH` from the contract. Similarly when the users exit the strategy they do not get any reward. We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We advise the client to revisit the design and ensure it is intended.

Alleviation

The team acknowledged this issue and stated they plan to convert it to a vault and provide reward in the next version.

NFT-08 | User Can Deposit More Tokens Than The Position Value

Category	Severity	Location	Status
Logical Issue	Medium	contracts/swNFTUpgrade.sol: 179, 215	Acknowledged

Description

In the function `deposit()`, the **require** check is used to ensure the total deposit amount is not more than the position value, however, does not consider the enter strategy amount. So the user still can deposit more tokens (transferred from other uses) than the position value if the user calls the function `enterStrategy()` to enter strategy before depositing `swETH` into position.

Recommendation

We recommend reviewing the logic again.

Alleviation

The team removed the **require** check in commit `1b67128b5c0b9904bebf596c32fa5040d72eb10f` and stated users could just deposit unlimited `swETH` but withdraw will still check the balance.

NFU-01 | Potential Gas Exhaustion On Function `stake`

Category	Severity	Location	Status
Gas Optimization	Minor	contracts/tests/swNFTUpgradeTestnet.sol	Acknowledged

Description

The `for` loop in the function `stake()` takes the unbounded array's length as the maximum iteration times. If the size of the array `Action` grows large, iterating through the entire array could be an expensive operation considering there are external calls in the `for` loop. Even worse, if the computation for each iteration is extremely complex, it could lead to gas insufficiency.

Recommendation

We recommend setting constraints to the length of the array `stakes`.

Alleviation

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

NFU-02 | Lack Of Validation For `msg.value`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/tests/swNFTUpgradeTestnet.sol	Resolved

Description

There is no validation to ensure `msg.value` equals the total stake amount of ETH.

```
  function stake(Stake[] calldata stakes) external payable returns (uint[] memory ids) {
      require(msg.value >= 1 ether, "Must send at least 1 ETH");
      require(msg.value % ETHER == 0, "stake value not multiple of Ether");
      ids = new uint[](stakes.length);
      uint totalAmount = msg.value;
      for(uint i = 0; i < stakes.length; i++){
          ids[i] = _stake(stakes[i].pubKey, stakes[i].signature,
stakes[i].depositDataRoot, stakes[i].amount);
          totalAmount -= stakes[i].amount;
      }
  }
```

Recommendation

Consider adding the validation to avoid the remaining ETH staying in the contract.

Alleviation

The team heeded our advice and resolved this issue in commit `bc2756fb84a680c4630f0e8c1bb73f0d5c550b91`.

SCK-01 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	contracts/Strategy.sol: 21	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

The team heeded our advice and resolved this issue in commit `ef2a10118b96224331976890967d2ffa55022f36`.

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.