



Security Assessment

UnshETH #2

CertiK Verified on Apr 4th, 2023





Certik Verified on Apr 4th, 2023

UnshETH #2

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

Executive Summary

TYPES

DeFi

ECOSYSTEM

Ethereum (ETH)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 04/04/2023

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/UnshETH/unsheth-contracts-v1.5/tree/92efb3b84ccf78052d5f9af3822c5af0de892939>
<https://github.com/UnshETH/unsheth-bridge>
[...View All](#)

COMMITTS

[92efb3b84ccf78052d5f9af3822c5af0de892939](#)
[e74db6733f4363dabdbffd21acc06121088d2038](#)
[...View All](#)

Vulnerability Summary



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.

1 Major

1 Resolved



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.

1 Medium

1 Resolved



Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

1 Minor

1 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.

4 Informational

3 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 | UNSHETH #2

I Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I Review Notes

I Decentralization Efforts

[Description](#)

[Recommendations](#)

[Short Term:](#)

[Long Term:](#)

[Permanent:](#)

I Findings

[RUT-01 : Lack Of `payable` Keyword](#)

[LSV-01 : Potential Overwrite `IsdIndex\[_Isd\]`](#)

[UET-01 : Missing Zero Address Validation](#)

[LSV-02 : `shanghaiTime` may not be modified](#)

[LSV-03 : The Purpose Of the Function `setVdAmm`](#)

[UET-02 : Missing Emit Events](#)

[UET-03 : Usage of `transfer\(\)`/`send\(\)` for sending Ether](#)

I Appendix

I Disclaimer

CODEBASE | UNSHETH #2

Repository

<https://github.com/UnshETH/unsheth-contracts-v1.5/tree/92efb3b84ccf78052d5f9af3822c5af0de892939>

<https://github.com/UnshETH/unsheth-bridge-contracts/tree/e74db6733f4363dabdbffd21acc06121088d2038>








Commit

[92efb3b84ccf78052d5f9af3822c5af0de892939](https://github.com/UnshETH/unsheth-contracts-v1.5/tree/92efb3b84ccf78052d5f9af3822c5af0de892939)

[e74db6733f4363dabdbffd21acc06121088d2038](https://github.com/UnshETH/unsheth-bridge-contracts/tree/e74db6733f4363dabdbffd21acc06121088d2038)

AUDIT SCOPE | UNSHETH #2

7 files audited ● 3 files with Acknowledged findings ● 4 files without findings

ID	File	SHA256 Checksum
● LSV	 src/LSDVault.sol	b88e9d647c11fbc689b2a6f332aad15ce495 1524e35c7f04eb4aa16b175d4a9
● RUT	 src/sgReciever.sol	97894cadea3e16773b64ff97d6dc909df16dda 0d4a292aca362b49bdf314bc12
● SUT	 src/sgSender.sol	a3a7bd57ccb7b610b23c4b8f34f48e940e732 1dad781ca9f25c9a195dc52bf66
● USO	 src/USH-OFT-BSC.sol	0d54a8d2bb5d6dc2fa359cac49cd1d2c230ff0 a23f3abce2827332c4069b796d
● USE	 src/USH-Proxy-ETH.sol	1761f3f4fd39c8eb86400054a41436f2a8f89ac f0f88b9b32cbd9bdd27211e81
● ETO	 src/unshETH-OFT-BSC.sol	61be281c59b88f9dfd981453d1fdb3d1aa6f8a 776981a10cc4fdb654fc751408
● ETE	 src/unshETH-Proxy-ETH.sol	dee81e79cca5e33cd0be005b319cb6eb3c749 fe3fe4f3a0a05d287e629e568f7

APPROACH & METHODS | UNSHETH #2

This report has been prepared for UnshETH to discover issues and vulnerabilities in the source code of the UnshETH #2 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.

REVIEW NOTES | UNSHETH #2

Financial Models

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The protocol offers users the ability to deposit tokens into a vault using various blockchain networks.

1. User can use usdt/weth/bnb to deposit to a vault. The vault can be deployed in the same chain or a different chain.
2. Whenever a user makes a deposit, the system calculates the current ETH value of each UnShETH token. It then uses this value to determine the total amount of UnShETH tokens to be issued to the user based on the formula: the total ETH value of tokens deposited divided by the latest ETH value of each UnShETH token.
3. Then the project minted the UnshETH token and send them to the user.
4. The project also provides a way to exit. Users can exit the protocol and get All kinds of tokens proportionally. A part of the tokens will be deducted from the fees.

Third-Party Dependencies

The contract serves as the underlying entity to interact with third-party protocols like `LayerZero` , `StarGate` , etc. 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.

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

DECENTRALIZATION EFFORTS | UNSHETH #2

Description

In the contract `USDTSGReciever` the role `_owner` has authority over the functions:

- `setSrcChainIdAndAddress` : change the `srcChainId` and `srcAddress` which is very important in the retry logic.
- `set_unsheth_gas_cost` : set the amount of gas used during the sending token process.
- `rescue_eth` : take out all the ETH. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

In the contract `BNBUnshethMinter` the role `_owner` has authority over the functions:

- `setPoolParams` : change the basic configuration of the contract.
- `setPaused` : pause/unpause the contract main functions.
- `setStargateGasAmount` : set the amount of the gas used during the swap process.

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

In the contract `LSDVault` the role `_owner` has authority over the functions:

- `setUnshethAddress` : change the address of the UnShETH token.
- `setAdmin` : set the address of the `admin` .
- `addLSD` : add basic information of an LSD token that can be deposited in this contract.
- `setLSDConfigs` : set the math parameters of the LSD tokens added in the contract.
- `enableLSD` : activate the LSD token.
- `enableAllLSDs` : activate all the LSD tokens.
- `disableLSD` : disable the LSD token.
- `toggleWeightCaps` : toggle the weight caps to make the contract check/uncheck the amount whether more than weighted caps.
- `toggleAbsoluteCaps` : toggle the absolute caps to make the contract check/uncheck the amount whether more than absolute caps.
- `toggleV1VaultAssetsForCaps` : control the `balanceUnderlying` whether count the V1 Vault Assets or not.
- `unpauseDeposits` : unpause the function `deposit` .
- `setRedeemFee` : set the fees when exiting the protocol.
- `createTimeLockProposal` : propose a proposal needed votes.
- `cancelTimeLockProposal` : cancel the proposal.
- `updateShanghaiTime` : update the `shanghaiTime` which is a key variable controls most function running.
- `pauseWithdrawals/unpauseWithdrawals` : pause/unpause the function `exit` .
- `disableVdAmm` : set the `swapperAddress` to 0.

- `withdrawAllETH` : withdraw all the ETH in the contract.

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

In the contract `LSDVault` the role `admin` has authority over the functions:

- `updateShanghaiTime` : update the `shanghaiTime` which is a key variable controls most function running.
- `pauseWithdrawals/unpauseWithdrawals` : pause/unpause the function `exit` .
- `disableVdAmm` : Set the `swapperAddress` to 0.
- `withdrawAllETH` : withdraw all the ETH in the contract.

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

Recommendations

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 recommend carefully managing 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 ($2/3$, $3/5$) 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 of privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being 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.

FINDINGS | UNSHETH #2



7

Total Findings

0

Critical

1

Major

1

Medium

1

Minor

4

Informational

This report has been prepared to discover issues and vulnerabilities for UnshETH #2. Through this audit, we have uncovered 7 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
RUT-01	Lack Of <code>payable</code> Keyword	Logical Issue	Major	● Resolved
LSV-01	Potential Overwrite <code>1sdIndex[_1sd]</code>	Logical Issue	Medium	● Resolved
UET-01	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
LSV-02	<code>shanghaiTime</code> May Not Be Modified	Logical Issue	Informational	● Resolved
LSV-03	The Purpose Of The Function <code>setVdAmm</code>	Logical Issue	Informational	● Resolved
UET-02	Missing Emit Events	Coding Style	Informational	● Acknowledged
UET-03	Usage Of <code>transfer()</code> / <code>send()</code> For Sending Ether	Volatile Code	Informational	● Resolved

RUT-01 | LACK OF payable KEYWORD

Category	Severity	Location	Status
Logical Issue	● Major	src/sgReciever.sol (e74db6733f4363dabdbffd21acc06121088d2038): 97, 138	● Resolved

Description

The `payable` keyword is missing from the `sgReceive()` function, making it impossible to receive ETH and pay the gas fee to execute the function `IOFTCore(proxyUnshethAddress).sendFrom` to send the unshETH back to `_toAddr` on BSC.

```
IOFTCore(proxyUnshethAddress).sendFrom{value:unsheth_gas_cost}( //TODO really think about this
    address(this), //current owner of the unsheth
    _chainId, //chain Id where the proxy of the unsheth exists (we recieve this in sgRecieve)
    abi.encode(toAddr), //the address we want the unsheth to end up in
    unshethMinted, //the amount of unsheth to send
    payable(address(this)), //the refund address if something goes wrong or excess
    address(0x0), //the ZRO Payment Address
    abi.encode("") //Adapter Params //TODO check on this
);
```

Recommendation

We recommend adding the `payable` keyword on the function.

Alleviation

[Certik]: The team heeded the advice and added the function `receive` to receive the ETH to resolved the finding in the commit [ddf21ec6e8a664ce4066992585f0e484bb976df4](https://github.com/Unsheth/Unsheth/commit/ddf21ec6e8a664ce4066992585f0e484bb976df4).

LSV-01 | POTENTIAL OVERWRITE `lsdIndex[_lsd]`

Category	Severity	Location	Status
Logical Issue	● Medium	src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 307 ~309, 456~457, 496~497	● Resolved

Description

After adding the first `_lsd`, the `lsdIndex[_lsd]` variable retains its initial value of 0 since the length of the `supportedLSDs` array is 1. Consequently, if the same `_lsd` is added again, the `require` statement in the `addLSD` function will evaluate to true, causing the `lsdIndex[_lsd]` variable can be overwritten.

```

194     function addLSD(address _lsd) public onlyOwner onlyWhenPaused {
195         require(lsdIndex[_lsd] == 0, "Lsd has already been added"); //fyi fails
on the first lsd being duplicated since it has actual index 0
196         supportedLSDs.push(_lsd);
197         lsdIndex[_lsd] = supportedLSDs.length-1; //reverse mapping of
supportedLSDs indices
198         isEnabled[_lsd] = false;
199         lsdConfigs[_lsd] = LSDConfig(0, 0, 0);
200         emit LSDAdded(_lsd);
201     }

```

```

448     function exit(uint256 amount) external nonReentrant {
449         require(migrated = false, "Already migrated, use v2 vault to exit");
450         require(block.timestamp > shanghaiTime, "Cannot exit until
shanghaiTime");
451         require(!withdrawalsPaused || block.timestamp > withdrawalUnpauseTime,
"Withdrawals are paused");
452         require(IERC20(unshETHAddress).balanceOf(msg.sender) >= amount,
"Insufficient unshETH");
453         uint256 shareOfUnsheth =
1e18*amount/IERC20(unshETHAddress).totalSupply();
454         uint256 fee = shareOfUnsheth*redeemFee/10000; //redeem fees are 100%
retained by remaining unshETH holders
455         IunshETH(unshETHAddress).minter_burn_from(msg.sender, amount);
456         for (uint256 i = 0; i < supportedLSDs.length; i = unchkIncr(i)) {
457             uint256 lsdBalance =
IERC20(supportedLSDs[i]).balanceOf(address(this));
458             uint256 amountPerLsd = (shareOfUnsheth-fee)*lsdBalance/1e18;
459             IERC20(supportedLSDs[i]).safeTransfer(msg.sender, amountPerLsd);
460         }
461     }

```

Recommendation

We recommend reviewing the logic and fixing the issue.

Alleviation

[UnshETH]: We're managing that through our pre-configuration checks - in any case the first index is set in the constructor which we did do correctly during deployment and cannot be overwritten, so the protocol logic is working as intended now.

UET-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	src/SgReceiver.sol (e74db6733f4363dabdbffd21acc06121088d2038): 47-51, 66; src/SgSender.sol (e74db6733f4363dabdbffd21acc06121088d2038): 55-61, 84; src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 122-123, 175, 180	● Acknowledged

Description

The cited address input is missing a check that it is not `address(0)`.

Recommendation

We recommend adding a check the passed-in address is not `address(0)` to prevent unexpected errors.

Alleviation

`[UnshETH]`: These issues are handled in our deploy scripts.

LSV-02 | `shanghaiTime` MAY NOT BE MODIFIED

Category	Severity	Location	Status
Logical Issue	● Informational	src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 531~532	● Resolved

Description

As per the validation of `_newTime` below, the variable `shanghaiTime` can be updated when `block.timestamp < _newTime < shanghaiTime + 4 weeks`.

```
530     function updateShanghaiTime(uint256 _newTime) external onlyOwnerOrAdmin {
531         require(_newTime < shanghaiTime + 4 weeks, "Cannot extend more than 4
weeks" );
532         require(_newTime > block.timestamp, "Cannot set shanghaiTime in the
past" );
533         shanghaiTime = _newTime;
534         emit ShanghaiTimeUpdated(shanghaiTime);
535     }
```

However, if the `block.timestamp >= shanghaiTime + 4weeks`, there is no way to set the `_newTime` as the `shanghaiTime` no matter what the `_newTime` is.

Recommendation

We recommend reviewing the logic again and ensuring it is as intended.

Alleviation

[UnshETH]: This is intentional logic. We don't want the time owner or admin the ability to extend the `shanghaitime` indefinitely, which would indefinitely postpone when users can withdraw their funds.

LSV-03 | THE PURPOSE OF THE FUNCTION `setVdAmm`

Category	Severity	Location	Status
Logical Issue	● Informational	src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 508~519	● Resolved

Description

There is a function `setVdAmm` which only changes the `swapperAddress` and `ammEnabled`. However, we can not find any usage of the `swapperAddress` and `ammEnabled`. We would like the team to elaborate more about the usages of the `swapperAddress` and `ammEnabled`.

Recommendation

We would like the team to elaborate more about the usages of the `swapperAddress` and `ammEnabled`.

Alleviation

[UnshETH]: We intend for this to enable giving token approvals to an "AMM" contract which will be used to facilitate swaps between the assets in the LSDVault. The activation of the AMM is locked under timelock + onlyOwner (multisig). We put the function in there to facilitate enabling this functionality when the AMM logic is ready to launch without requiring a full migration of user funds.

UET-02 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	src/sgReciever.sol (e74db6733f4363dabdbffd21acc06121088d2038): 64, 70, 81; src/sgSender.sol (e74db6733f4363dabdbffd21acc06121088d2038): 76~81, 89, 94; src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 564	● Acknowledged

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

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

UET-03 | USAGE OF `transfer()` / `send()` FOR SENDING ETHER

Category	Severity	Location	Status
Volatile Code	● Informational	src/sgReciever.sol (e74db6733f4363dabdbffd21acc06121088d2038): 83; src/LSDVault.sol (92efb3b84ccf78052d5f9af3822c5af0de892939): 567	● Resolved

Description

Using Solidity's `transfer()` and `send()` functions for transferring Ether is not recommended, since some contracts may not be able to receive the funds. These functions forward only a fixed amount of gas (2300 specifically) and the receiving contracts may run out of gas before finishing the transfer. Additionally, gas costs may increase in the future. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.

Recommendation

We recommend using the `Address.sendValue()` function from OpenZeppelin.

Since `Address.sendValue()` may allow reentrancy, we also recommend guarding against reentrancy attacks by utilizing the Checks-Effects-Interactions Pattern or applying OpenZeppelin ReentrancyGuard.

Alleviation

[Certik]: The team heeded the advice and resolved the finding in the commit [ddf21ec6e8a664ce4066992585f0e484bb976df4](https://github.com/Uniswap/v2-core/commit/ddf21ec6e8a664ce4066992585f0e484bb976df4).

APPENDIX | UNSHETH #2

Finding Categories

Categories	Description
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 | 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.



