Code Security Assessment

Stripto Marketplace-Addendum

Jan 28th, 2022

Table of Contents

Summary

Overview

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

Findings

- SSM-01 : Third Party Dependencies
- SSM-02 : Centralization Related Risks
- SSM-03 : Missing emit events
- SSM-04 : Variables that could be declared as immutable
- SSM-05 : Initial Token Distribution
- SSM-06 : Return value not handled
- SSM-07 : Unchecked value from low-level call
- SSM-08 : Redundant Operation
- SSM-09 : Limited Effect to Delay the Transfer
- SSM-10 : Potential Logic Flaw for Fee Calculation
- SSM-11 : Fee Collectors
- SSM-12 : Redundant Check
- SSM-13 : Unchecked Value of ERC-20 `transfer()` Call

Appendix

Disclaimer

About

Summary

This report has been prepared for Stripto Marketplace-Addendum to discover issues and vulnerabilities in the source code of the Stripto Marketplace-Addendum 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	Stripto Marketplace-Addendum
Platform	bsc
Language	Solidity
Codebase	https://github.com/striptoken/strip_token_contract/blob/604435d12c269da56746ec48d7cbb7d032e9df95/Stripto_smar
Commit	604435d12c269da56746ec48d7cbb7d032e9df95 ed491e9fff4e307485020ede6785e8d1f80c3b36

Audit Summary

Delivery Date	Jan 28, 2022
Audit Methodology	Static Analysis, Manual Review

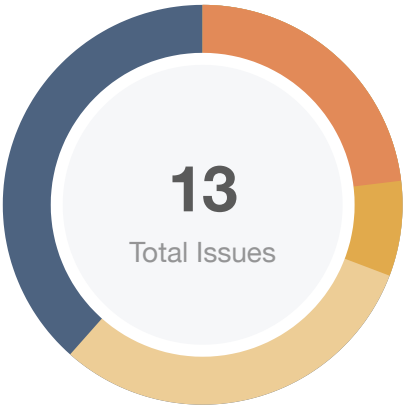
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated
Critical	0	0	0	0	0	0
Major	3	0	0	2	0	0
Medium	1	0	0	1	0	0
Minor	4	0	0	2	0	0
Informational	5	0	0	3	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SSM	Stripto_smart_contract_revised.sol	b99dd906b204ad2915b99ff4a8f641ffdffa7311aa729886f0444cbe6d9411ec

Findings



Critical	**0** (0.00%)
Major	**3** (23.08%)
Medium	**1** (7.69%)
Minor	**4** (30.77%)
Informational	**5** (38.46%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
SSM-01	Third Party Dependencies	Volatile Code	Minor	Acknowledged
SSM-02	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
SSM-03	Missing emit events	Coding Style	Informational	Resolved
SSM-04	Variables that could be declared as immutable	Gas Optimization	Informational	Resolved
SSM-05	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
SSM-06	Return value not handled	Volatile Code	Informational	Acknowledged
SSM-07	Unchecked value from low-level call	Volatile Code	Minor	Resolved
SSM-08	Redundant Operation	Gas Optimization	Informational	Acknowledged
SSM-09	Limited Effect to Delay the Transfer	Logical Issue	Minor	Resolved
SSM-10	Potential Logic Flaw for Fee Calculation	Logical Issue	Major	Resolved
SSM-11	Fee Collectors	**Centralization / Privilege**	**Medium**	Acknowledged
SSM-12	Redundant Check	Logical Issue	Informational	Acknowledged
SSM-13	Unchecked Value of ERC-20 `transfer()` Call	Logical Issue	Minor	Acknowledged

SSM-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	Stripto_smart_contract_revised.sol: 373	Acknowledged

Description

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

Recommendation

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

Alleviation

No Alleviation.

SSM-02 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	Stripto_smart_contract_revised.sol: 247~250, 252~256, 414~421, 424~428, 430~433, 436~438, 440~444, 446~450, 453~457, 464~474, 476~481, 483~487, 497~502, 504~509, 511~514, 671~677, 680~683	Acknowledged

Description

In the contract, `Ownable`, the role, `_owner`, has authority over the following functions.

- `renounceOwnership()`
- `transferOwnership(address newOwner)`

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

In the contract, `Stripto`, the role, `_owner`, has authority over the following functions.

- `enableTrading()`
- `removeLimits()`
- `removeBoughtEarly(address account)`
- `disableTransferDelay()`
- `updateMaxBuyAmount(uint256 newNum)`
- `updateMaxSellAmount(uint256 newNum)`
- `updateSwapTokensAtAmount(uint256 newAmount)`
- `airdropToWallets(address[] memory airdropWallets, uint256[] memory amounts)`
- `excludeFromMaxTransaction(address updAds, bool isEx)`
- `setAutomatedMarketMakerPair(address pair, bool value)`
- `updateBuyFees(uint256 _operationsFee, uint256 _liquidityFee)`
- `updateSellFees(uint256 _operationsFee, uint256 _liquidityFee)`
- `excludeFromFees(address account, bool excluded)`
- `transferForeignToken(address _token, address _to)`
- `withdrawStuckETH()`
- `setOperationsAddress()` in the commit ed491e9fff4e307485020ede6785e8d1f80c3b36

Any compromise to the `_owner` account may allow the 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., multisignature 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.

Alleviation

No Alleviation.

SSM-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Stripto_smart_contract_revised.sol: 424~428, 436~438, 453~457, 464~474, 476~481, 483~487, 497~502, 504~509, 680~683	Resolved

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

The development team resolved this issue in commit ed491e9fff4e307485020ede6785e8d1f80c3b36.

SSM-04 | Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	Stripto_smart_contract_revised.sol: 302	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 development team resolved this issue in commit ed491e9fff4e307485020ede6785e8d1f80c3b36.

SSM-05 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Stripto_smart_contract_revised.sol: 405	Acknowledged

Description

All of the `$STRIP` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `$STRIP` 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

No Alleviation.

SSM-06 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	Stripto_smart_contract_revised.sol: 631~638	Acknowledged

Description

The return value of function `addLiquidityETH` is not properly handled.

```
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            address(0xdead),
            block.timestamp
        );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

No Alleviation.

SSM-07 | Unchecked Value From Low-level Call

Category	Severity	Location	Status
Volatile Code	Minor	Stripto_smart_contract_revised.sol: 668, 682	Resolved

Description

Ignores return value of low-level calls.

```
668  (success,) = address(operationsAddress).call{value: address(this).balance}("");
```

```
682  (success,) = address(msg.sender).call{value: address(this).balance}("");
```

Recommendation

If you choose to use the low-level call methods, make sure to handle the possibility that the call will fail, by checking the return value.

Alleviation

The development team resolved this issue in commit ed491e9fff4e307485020ede6785e8d1f80c3b36.

SSM-08 | Redundant Operation

Category	Severity	Location	Status
Gas Optimization	Informational	Stripto_smart_contract_revised.sol: 406	Acknowledged

Description

Since the `newOwner` is `msg.sender`, there is no need to transfer ownership because the `_owner` is initialized as `msg.sender` in the contract `Ownable` when it is deployed.

```
232     constructor () {
233         address msgSender = _msgSender();
234         _owner = msgSender;
235         emit OwnershipTransferred(address(0), msgSender);
236     }
```

```
369 constructor() ERC20("STRIPTO", "$STRIP") {
370 ...
371 address newOwner = msg.sender;
372 ...
373 ...
374 transferOwnership(newOwner);
```

Recommendation

Consider removing the linked code for gas efficiencies and code readability.

Alleviation

No Alleviation.

SSM-09 | Limited Effect To Delay The Transfer

Category	Severity	Location	Status
Logical Issue	Minor	Stripto_smart_contract_revised.sol: 541	Resolved

Description

It is noted that the `if` condition is limited to the specified `uniswapV2Pair` pair. There are probably multiple pairs in the DEX or in different DEXs. Is that designed as expected?

Recommendation

Consider using a white list.

Alleviation

The development team heeded our advice and resolved this issue in commit ed491e9fff4e307485020ede6785e8d1f80c3b36.

SSM-10 | Potential Logic Flaw For Fee Calculation

Category	Severity	Location	Status
Logical Issue	Major	Stripto_smart_contract_revised.sol: 579~595	Resolved

Description

According the logic of this contract, the early buyer should be punished to undertake high fees and these fees will be accumulated to `tokensForLiquidity` and `tokensForOperations`:

```
579  if(boughtEarly[from] && automatedMarketMakerPairs[to] && block.timestamp <
earlyBuyPenaltyEnd){
580          fees = amount * 75 / 100;
581          tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
582          tokensForOperations += fees * sellOperationsFee / sellTotalFees;
583  }
```

However, the above fees will be overwritten and the `tokensForLiquidity` and `tokensForOperations` will be accumulated again by the following code:

```
585  if (automatedMarketMakerPairs[to] && sellTotalFees > 0){
586          fees = amount * sellTotalFees /100;
587          tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
588          tokensForOperations += fees * sellOperationsFee / sellTotalFees;
589  }
```

Recommendation

Consider refactoring the `if` condition to `if-else` condition:

```
if(boughtEarly[from] && automatedMarketMakerPairs[to] && block.timestamp <
earlyBuyPenaltyEnd){
    fees = amount * 75 / 100;
    tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
    tokensForOperations += fees * sellOperationsFee / sellTotalFees;
}
// on sell
else if (automatedMarketMakerPairs[to] && sellTotalFees > 0){
    fees = amount * sellTotalFees /100;
    tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
    tokensForOperations += fees * sellOperationsFee / sellTotalFees;
}
```

Alleviation

The development team resolved this issue in commit ed491e9fff4e307485020ede6785e8d1f80c3b36.

SSM-11 | Fee Collectors

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	Stripto_smart_contract_revised.sol: 302	Acknowledged

Description

There is a fee collector, i.e. `operationsAddress`, over time, this account would gain more and more fees.

Recommendation

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

No Alleviation.

SSM-12 | Redundant Check

Category	Severity	Location	Status
Logical Issue	Informational	Stripto_smart_contract_revised.sol: 524	Acknowledged

Description

In the function `_transfer`, the aforementioned statement `to != address(0)` in `if` condition has been checked in line 519:

```
519  require(to != address(0), "ERC20: transfer to the zero address");
```

Recommendation

Consider removing the redundant check.

Alleviation

No Alleviation.

SSM-13 | Unchecked Value Of ERC-20 `transfer()` Call

Category	Severity	Location	Status
Logical Issue	Minor	Stripto_smart_contract_revised.sol: 675	Acknowledged

Description

The linked `transfer()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation

No Alleviation.

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.