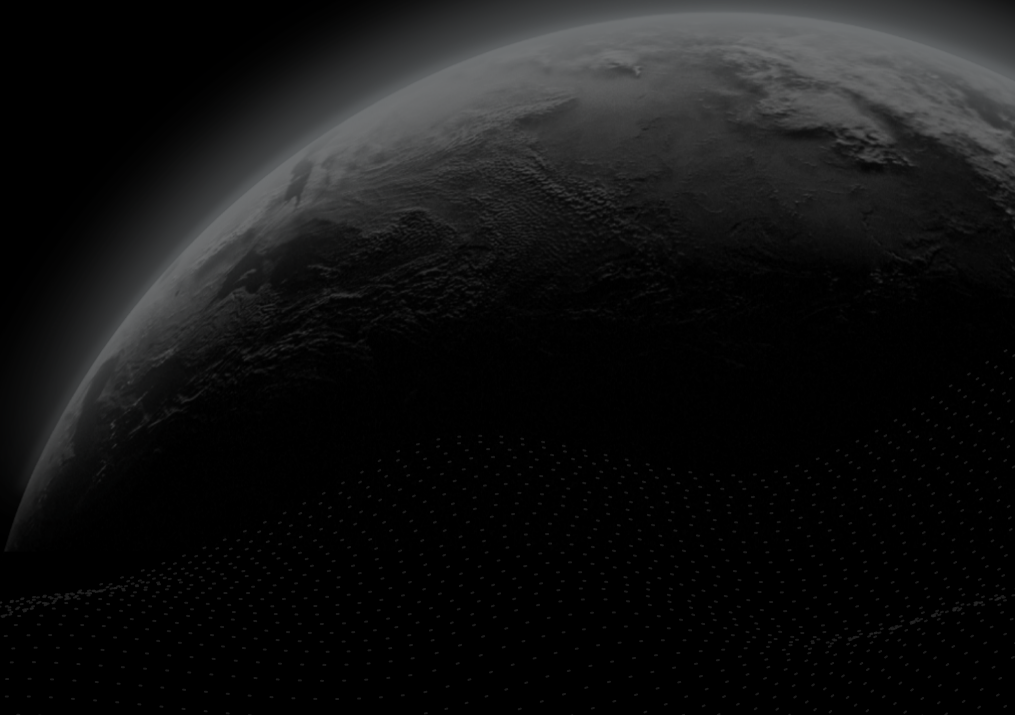




Preliminary Comments

WaffleStay

CertiK Verified on Nov 21st, 2022





Certik Verified on Nov 21st, 2022

WaffleStay

These preliminary comments were prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES

ERC-20

ECOSYSTEM

Ethereum

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 11/21/2022

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/wafflestay/WAFL>[...View All](#)

COMMITTS

[2ed0393aa783ca175d9ca01e9c9a924b038ebd04](#)[cbfa735b73c9e98c6f1a6918c5d9e9f9350d781c](#)[e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b](#)[...View All](#)

Vulnerability Summary



5

Total Findings

3

Resolved

0

Mitigated

0

Partially Resolved

2

Acknowledged

0

Declined

0

Unresolved

1

Critical

1 Resolved



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.

2

Major

2 Acknowledged



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.

0

Medium

0 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 Resolved



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.

1

Informational

1 Resolved



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.

0

Discussion

The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | WAFFLESTAY

I **Summary**

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

I **Findings**

GLOBAL-01 : Centralization Risks in WaffleStay.sol

WSE-01 : Missing Check For `verifyMultiSig()` in `freezeAccount()` and `unFreezeAccount()`

WSE-02 : Initial Token Distribution

WSE-03 : Potential Initialization By Frontrunner

WSE-04 : Missing Check For `v` And `s`

I **Optimizations**

WSE-05 : Functions Can Be Merged

I **Formal Verification**

Considered Functions And Scope

Verification Results

I **Appendix**

I **Disclaimer**

CODEBASE | WAFFLESTAY

Repository

<https://github.com/wafflestay/WAFL>

Commit

[2ed0393aa783ca175d9ca01e9c9a924b038ebd04](#)

[cbfa735b73c9e98c6f1a6918c5d9e9f9350d781c](#)

[e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b](#)

AUDIT SCOPE | WAFFLESTAY

1 file audited ● 1 file with Acknowledged findings

ID	File	SHA256 Checksum
----	------	-----------------

● WSE



WAFL/WaffleStayERC20.sol

484d055134a43f048194db6673fb811b53b93c4de81492d336c8f6a55094eb65

APPROACH & METHODS | WAFFLESTAY

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

FINDINGS | WAFFLESTAY



5

Total Findings

1

Critical

2

Major

0

Medium

1

Minor

1

Informational

0

Discussion

This report has been prepared to discover issues and vulnerabilities for WaffleStay. Through this audit, we have uncovered 5 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:

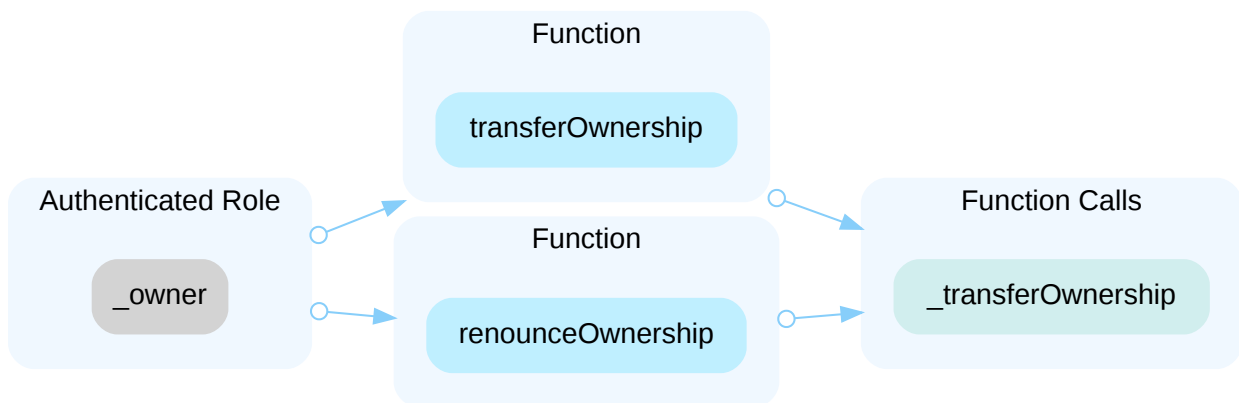
ID	Title	Category	Severity	Status
<u>GLOBAL-01</u>	Centralization Risks In WaffleStay.Sol	Centralization / Privilege	Major	● Acknowledged
<u>WSE-01</u>	Missing Check For <code>verifyMultiSig()</code> In <code>freezeAccount()</code> And <code>unFreezeAccount()</code>	Logical Issue	Critical	● Resolved
<u>WSE-02</u>	Initial Token Distribution	Centralization / Privilege	Major	● Acknowledged
<u>WSE-03</u>	Potential Initialization By Frontrunner	Volatile Code	Minor	● Resolved
<u>WSE-04</u>	Missing Check For <code>v</code> And <code>s</code>	Logical Issue	Informational	● Resolved

GLOBAL-01 | CENTRALIZATION RISKS IN WAFFLESTAY.SOL

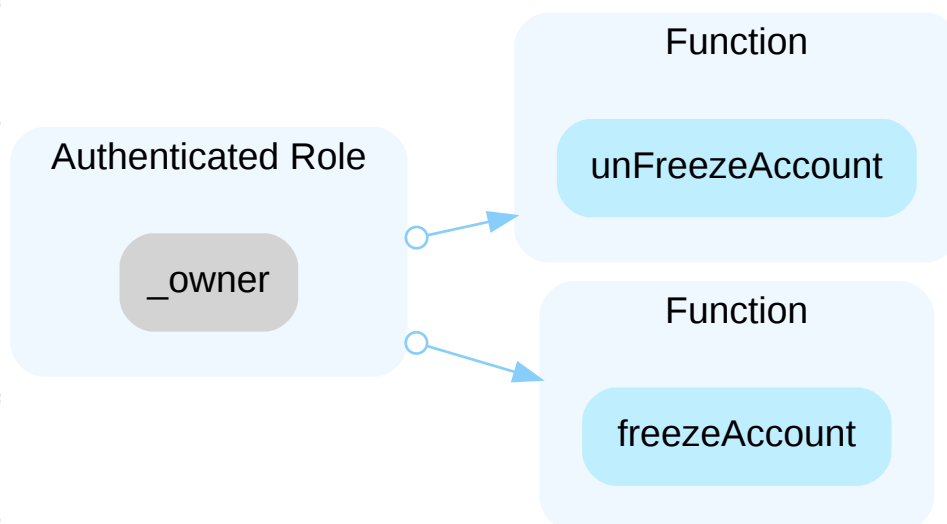
Category	Severity	Location	Status
Centralization / Privilege	Major		Acknowledged

Description

In the contract `ownable` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `WaffleStayERC20` the role `_owner` has authority over the functions shown in the diagram below.



Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and update the sensitive settings and execute sensitive functions of the project

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 ($\frac{2}{3}$, $\frac{3}{5}$) 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 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.

WSE-01 MISSING CHECK FOR `verifyMultiSig()` IN `freezeAccount()` AND `unFreezeAccount()`

Category	Severity	Location	Status
Logical Issue	Critical	WAFL/WaffleStayERC20.sol (cbfa735): 556, 570	Resolved

Description

In the functions `freezeAccount()` and `unFreezeAccount()`, the `verifyMultiSig()` is invoked to verify the signature of the operator. However everyone can input an empty `signatures` with a length of 0, and then bypass the checks in the `for` loop in the `verifyMultiSig()`

Recommendation

We advise the team to consider adding a check to guarantee the length of `signatures` is not 0 in the functions `freezeAccount()` and `unFreezeAccount()` and preferably check if a valid signature is provided in these two functions input according to design.

Meanwhile, we advise adding a check to the return value of `verifyMultiSig()` as it will return a boolean value.

Alleviation

[Certik]: The team heeded the advice and resolved the finding in `waffleStay.sol` in the commit [e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b](#)

WSE-02 INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	● Major	WAFL/WaffleStayERC20.sol (2ed0393): 319	● Acknowledged

Description

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

WSE-03 | POTENTIAL INITIALIZATION BY FRONTRUNNER

Category	Severity	Location	Status
Volatile Code	Minor	WAFL/WaffleStayERC20.sol (cbfa735): 535	Resolved

Description

In the `WaffleStayERC20` contract, the function `initSigners()` can be called by anyone to initialize the contract with the signers. Although the project deployer can discard incorrectly initialized contracts, it might still bring errors if the deployment is not properly processed. One of the possible scenarios is described below:

1. The deployer writes a script to deploy and initialize the contract.
2. The attacker noticed the deployment and initialized the contract before the `initSigners()` called by the deployer is committed (this can be achieved by front-running).
3. The deployment script mistakenly ignores the error of initializing the contract and continues executing other transactions in the script.

In this way, the attacker can inject suspicious addresses into the `signers` of the contract.

Recommendation

We recommend checking the status of initialization in the deployment process adding proper access control to the aforementioned functions.

Alleviation

[Certik]: The team heeded the advice and resolved the finding in `WaffleStay.sol` in the commit [e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b](https://github.com/WaffleStay/WaffleStay/commit/e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b)

WSE-04 | MISSING CHECK FOR `v` AND `s`

Category	Severity	Location	Status
Logical Issue	● Informational	WAFL/WaffleStayERC20.sol (cbfa735): 617	● Resolved

Description

The following description is adapted from OpenZeppelin's ECDSA file:

EIP-2 still allows signature malleability for `ecrecover()`. Appendix F in the Ethereum Yellow paper, defines the valid range for `s` in (311): $0 < s < \text{secp256k1n} \div 2 + 1$ and for the recovery identifier (312): $v \in \{0,1\}$. This should not be confused with the input for `ecrecover()` where $v \in \{27,28\}$. (See doc) However, these values can be obtained by taking `27 + "recovery identifier"`, so that they will also yield a unique signature and are often the `v` values returned from signatures. (For example `web3.eth.accounts.sign()`)

If your library generates malleable signatures, such as `s`-values in the upper range, calculate a new `s`-value with `0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141 - s1` and flip `v` from `27` to `28` or vice versa. If your library also generates signatures with `0/1` for `v` instead `27/28`, add `27` to `v` so that `ecrecover()` accepts these signatures as well.

Recommendation

We recommend adding the following checks or to consider the example in `ECDSA.sol` from the OpenZeppelin library.

```
require(uint256(s) <=
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0, "ECDSA: invalid
signature 's' value");
require(uint8(v) == 27 || uint8(v) == 28, "ECDSA: invalid signature 'v' value");
```

Alleviation

[certik]: The team heeded the advice and resolved the finding in `waffleStay.sol` in the commit `e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b`



OPTIMIZATIONS | WAFFLESTAY

ID	Title	Category	Severity	Status
<u>WSE-05</u>	Functions Can Be Merged	Gas Optimization	Optimization	● Resolved

WSE-05 | FUNCTIONS CAN BE MERGED

Category	Severity	Location	Status
Gas Optimization	● Optimization	WAFL/WaffleStayERC20.sol (cbfa735): 547, 561	● Resolved

Description

The `freezeAccount()` and `unFreezeAccount()` functions can be merged into a single function because the only difference between two functions is the emit event.

Recommendation

We advise the team to merge `freezeAccount()` and `unFreezeAccount()` and remove the redundant one.

Alleviation

[Certik]: The team heeded the advice and resolved the finding in `waffleStay.sol` in the commit [e8e0a3cc777e7c6a642c27b5a43941ed0a1c2f9b](#)

FORMAL VERIFICATION | WAFFLESTAY

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied automated formal verification (symbolic model checking) to prove that well-known functions in the smart contracts adhere to their expected behavior.

Considered Functions And Scope

Verification of ERC-20 compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply`, which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title	
erc20-transfer-revert-zero	Function <code>transfer</code>	Prevents Transfers to the Zero Address
erc20-transfer-correct-amount	Function <code>transfer</code>	Transfers the Correct Amount in Non-self Transfers
erc20-transfer-correct-amount-self	Function <code>transfer</code>	Transfers the Correct Amount in Self Transfers
erc20-transfer-change-state	Function <code>transfer</code>	Has No Unexpected State Changes
erc20-transfer-exceed-balance	Function <code>transfer</code>	Fails if Requested Amount Exceeds Available Balance
erc20-transfer-recipient-overflow	Function <code>transfer</code>	Prevents Overflows in the Recipient's Balance
erc20-transfer-false	If Function <code>transfer</code> Returns <code>false</code>	, the Contract State Has Not Been Changed
erc20-transfer-never-return-false	Function <code>transfer</code>	Never Returns <code>false</code>
erc20-transferfrom-revert-from-zero	Function <code>transferFrom</code>	Fails for Transfers From the Zero Address
erc20-transferfrom-revert-to-zero	Function <code>transferFrom</code>	Fails for Transfers To the Zero Address

Property Name	Title	
erc20-transferfrom-correct-amount	Function <code>transferFrom</code>	Transfers the Correct Amount in Non-self Transfers
erc20-transferfrom-correct-amount-self	Function <code>transferFrom</code>	Performs Self Transfers Correctly
erc20-transferfrom-correct-allowance	Function <code>transferFrom</code>	Updated the Allowance Correctly
erc20-transferfrom-fail-exceed-balance	Function <code>transferFrom</code>	Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-change-state	Function <code>transferFrom</code>	Has No Unexpected State Changes
erc20-transferfrom-fail-exceed-allowance	Function <code>transferFrom</code>	Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-false	If Function <code>transferFrom</code> Returns <code>false</code>	, the Contract's State Has Not Been Changed
erc20-transferfrom-fail-recipient-overflow	Function <code>transferFrom</code>	Prevents Overflows in the Recipient's Balance
erc20-transferfrom-never-return-false	Function <code>transferFrom</code>	Never Returns <code>false</code>
erc20-totalsupply-succeed-always	Function <code>totalSupply</code>	Always Succeeds
erc20-totalsupply-correct-value	Function <code>totalSupply</code>	Returns the Value of the Corresponding State Variable
erc20-totalsupply-change-state	Function <code>totalSupply</code>	Does Not Change the Contract's State
erc20-balanceof-succeed-always	Function <code>balanceOf</code>	Always Succeeds
erc20-balanceof-correct-value	Function <code>balanceOf</code>	Returns the Correct Value
erc20-allowance-succeed-always	Function <code>allowance</code>	Always Succeeds
erc20-balanceof-change-state	Function <code>balanceOf</code>	Does Not Change the Contract's State
erc20-allowance-correct-value	Function <code>allowance</code>	Returns Correct Value
erc20-allowance-change-state	Function <code>allowance</code>	Does Not Change the Contract's State
erc20-approve-revert-zero	Function <code>approve</code>	Prevents Giving Approvals For the Zero Address
erc20-approve-succeed-normal	Function <code>approve</code>	Succeeds for Admissible Inputs
erc20-approve-correct-amount	Function <code>approve</code>	Updates the Approval Mapping Correctly

Property Name	Title
erc20-approve-change-state	Function <code>approve</code> Has No Unexpected State Changes
erc20-approve-false	If Function <code>approve</code> Returns <code>false</code> , the Contract's State Has Not Been Changed
erc20-approve-never-return-false	Function <code>approve</code> Never Returns <code>false</code>

Verification Results

For the following contracts, model checking established that each of the 38 properties that were in scope of this audit (see scope) are valid:

Contract WaffleStayERC20 (Source File WAFL/WaffleStayERC20.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	● True	
erc20-transfer-correct-amount	● True	
erc20-transfer-correct-amount-self	● True	
erc20-transfer-change-state	● True	
erc20-transfer-exceed-balance	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-false	● True	
erc20-transfer-never-return-false	● True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	● True	
erc20-transferfrom-revert-to-zero	● True	
erc20-transferfrom-correct-amount	● True	
erc20-transferfrom-correct-amount-self	● True	
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-change-state	● True	
erc20-transferfrom-fail-exceed-allowance	● True	
erc20-transferfrom-false	● True	
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-never-return-false	● True	




Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	● True	
erc20-totalsupply-correct-value	● True	
erc20-totalsupply-change-state	● True	







Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	● True	
erc20-balanceof-correct-value	● True	
erc20-balanceof-change-state	● True	

Detailed results for function allowance

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	 True	
erc20-allowance-correct-value	 True	
erc20-allowance-change-state	 True	

Detailed results for function approve

Property Name	Final Result	Remarks
erc20-approve-revert-zero	 True	
erc20-approve-succeed-normal	 True	
erc20-approve-correct-amount	 True	
erc20-approve-change-state	 True	
erc20-approve-false	 True	
erc20-approve-never-return-false	 True	

APPENDIX | WAFFLESTAY

Finding Categories

Categories	Description
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.

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.

