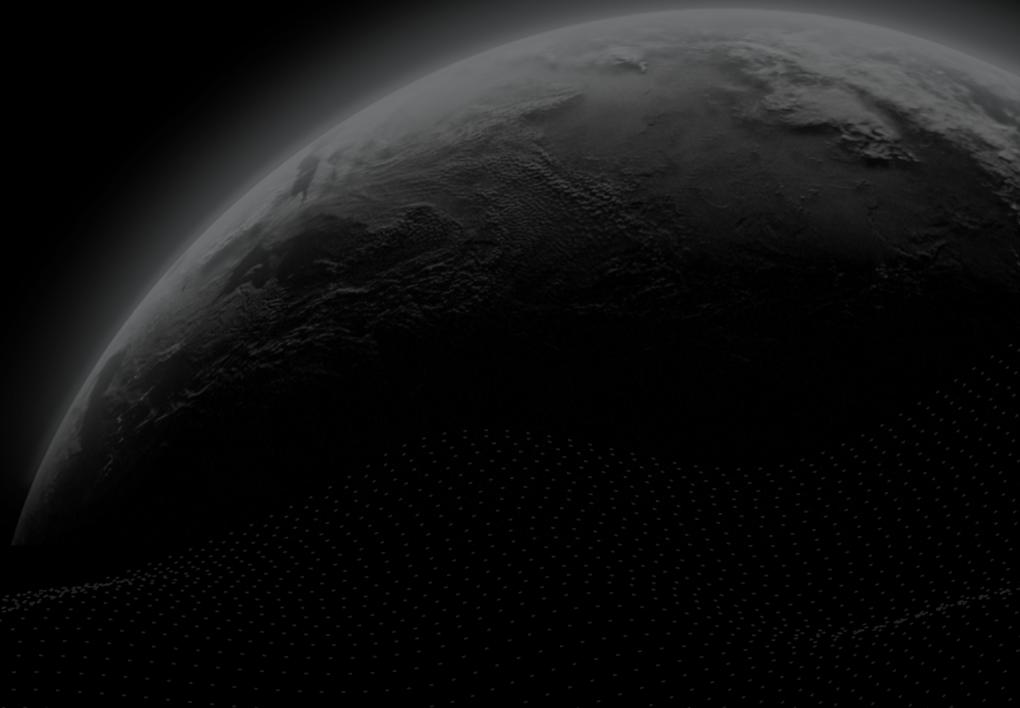




Preliminary Comments

House of Mandela Token

CertiK Verified on Oct 28th, 2022





CertiK Verified on Oct 28th, 2022

House of Mandela Token

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 10/28/2022

KEY COMPONENTS
N/A

CODEBASE
<https://github.com/HOM-dev/token/>
[...View All](#)

COMMITTS
base: [799db814bc2b766573685dbddccc4eb8fe10179b](#)
update: [5643aabe64f46025aa21dd02ef4334ede268fbbd](#)
[...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 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.

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 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.

5 Informational

3 Resolved, 1 Acknowledged, 1 Unresolved

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 | HOUSE OF MANDELA TOKEN

I Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

I Findings

HOM-01 : Centralization Related Risks

HOM-02 : Multiple Functions use `allowed`

HOM-03 : Missing Checks for `newReleaseAddress`

HOM-04 : Unnecessary Complicated Expression

HOM-05 : Typo

HOM-06 : Supply Can Differ From Available Tokens

HOM-07 : Too Many Digits

HOM-08 : Time is Incorrect

I Optimizations

HOM-09 : State Variable Should Be Declared Constant

I Formal Verification

Considered Functions And Scope

Verification Results

I Appendix

I Disclaimer

CODEBASE | HOUSE OF MANDELA TOKEN

Repository

<https://github.com/HOM-dev/token/>

Commit

base: [799db814bc2b766573685dbddccc4eb8fe10179b](#)

update: [5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

AUDIT SCOPE | HOUSE OF MANDELA TOKEN

1 file audited ● 1 file with Unresolved findings

ID	File	SHA256 Checksum
● HOM	 token.sol	13591dbc9d3e22f5341bbd9b2f6370c9a3ed2b8dc73b6c8baa227d6183fdbbc7

APPROACH & METHODS | HOUSE OF MANDELA TOKEN

This report has been prepared for House of Mandela to discover issues and vulnerabilities in the source code of the House of Mandela Token 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 | HOUSE OF MANDELA TOKEN



8

Total Findings

0

Critical

1

Major

1

Medium

1

Minor

5

Informational

0

Discussion

This report has been prepared to discover issues and vulnerabilities for House of Mandela Token. Through this audit, we have uncovered 8 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:

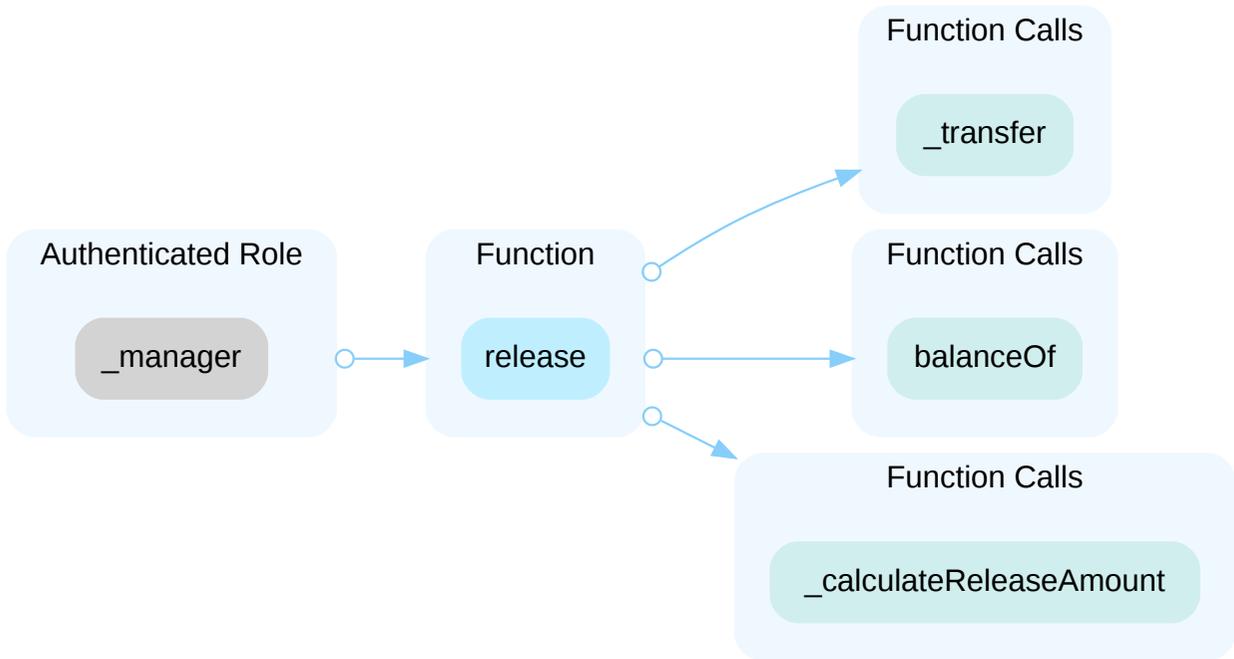
ID	Title	Category	Severity	Status
HOM-01	Centralization Related Risks	Centralization / Privilege	Major	● Acknowledged
HOM-02	Multiple Functions Use <code>_allowed</code>	Coding Style, Logical Issue	Medium	● Resolved
HOM-03	Missing Checks For <code>newReleaseAddress</code>	Volatile Code	Minor	● Resolved
HOM-04	Unnecessary Complicated Expression	Coding Style	Informational	● Resolved
HOM-05	Typo	Inconsistency	Informational	● Resolved
HOM-06	Supply Can Differ From Available Tokens	Inconsistency	Informational	● Unresolved
HOM-07	Too Many Digits	Coding Style	Informational	● Acknowledged
HOM-08	Time Is Incorrect	Logical Issue	Informational	● Resolved

HOM-01 | CENTRALIZATION RELATED RISKS

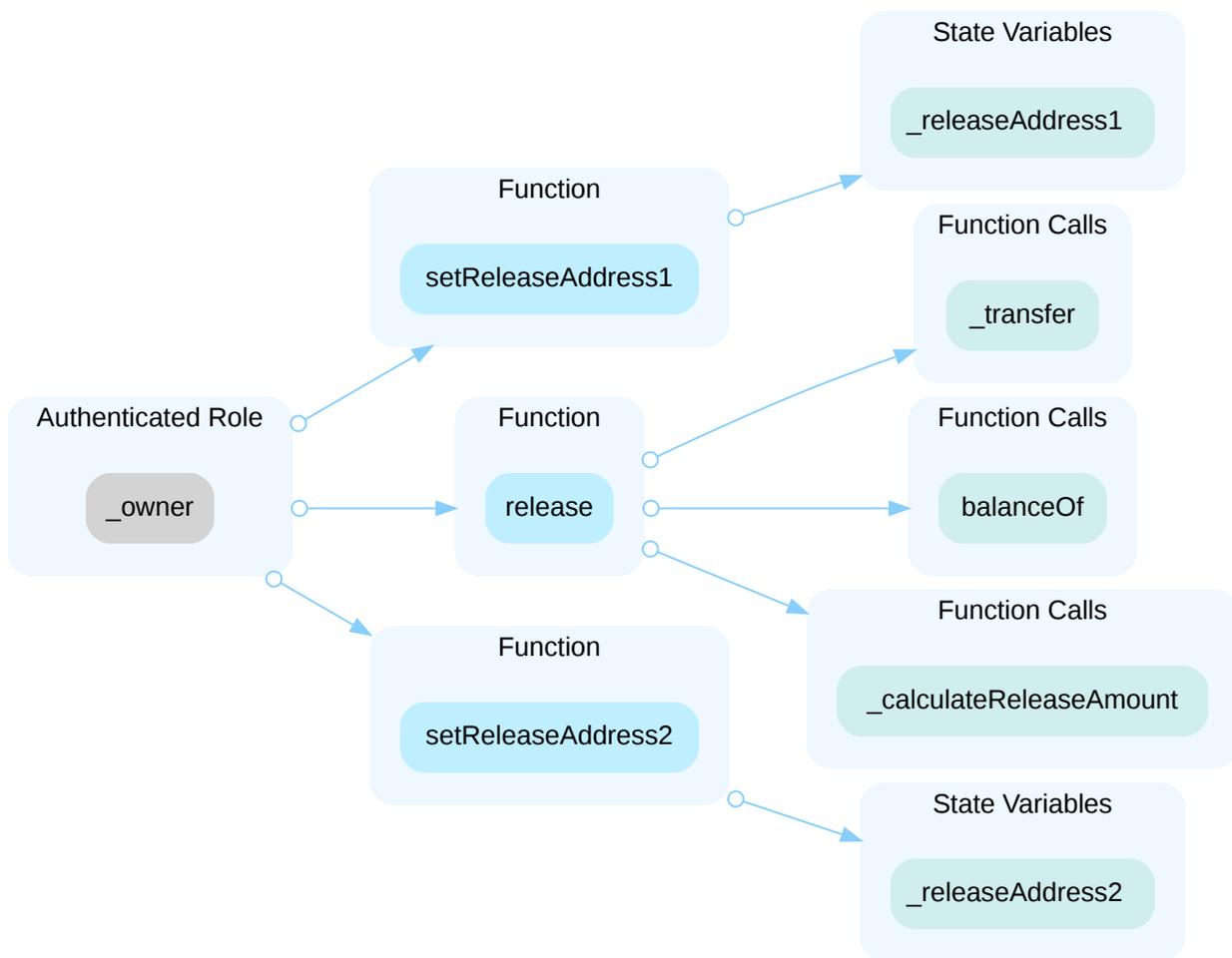
Category	Severity	Location	Status
Centralization / Privilege	● Major	token.sol (base): 164, 174, 240, 250, 275, 285, 570, 583, 600	● Acknowledged

Description

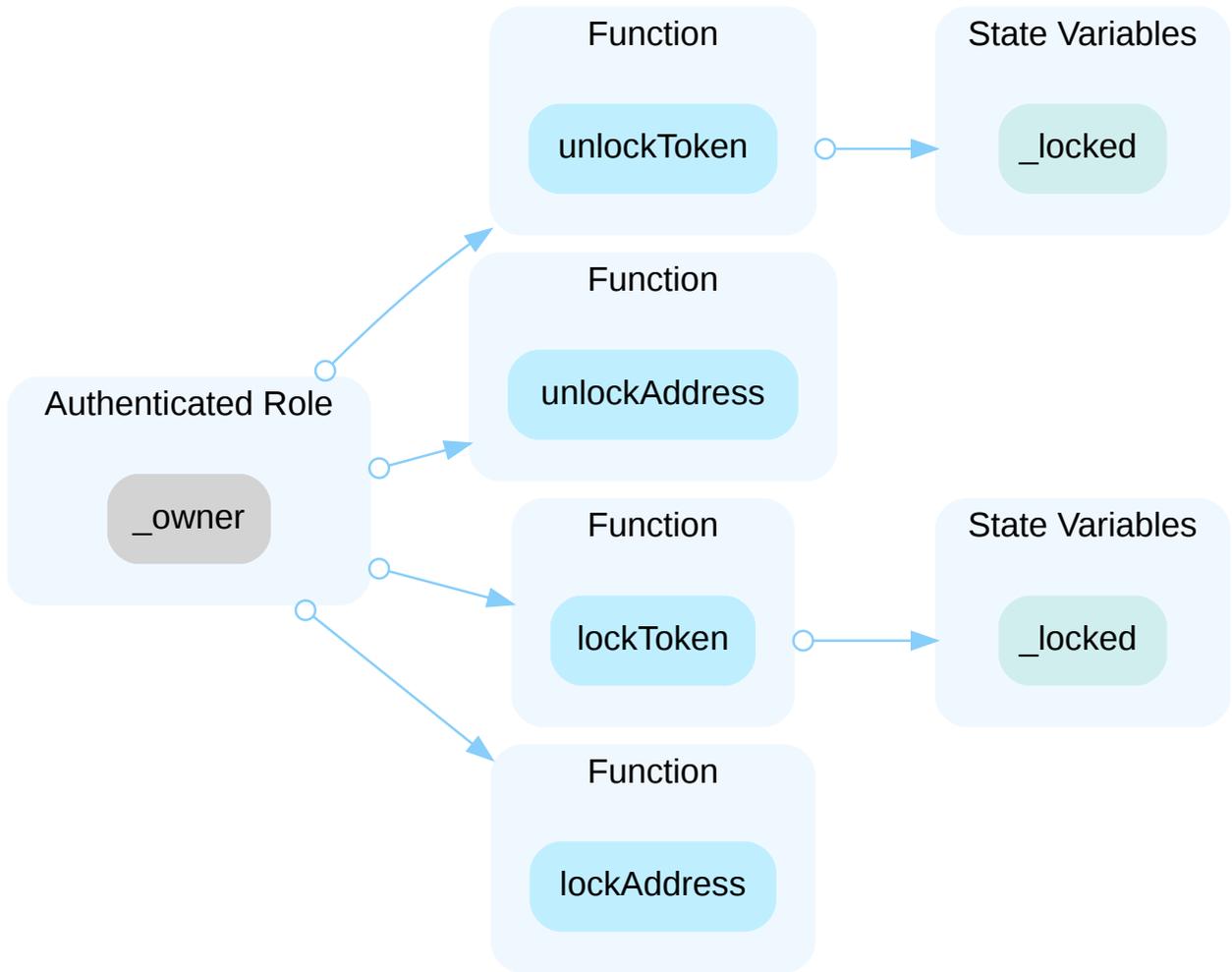
In the contract ERC20 the role `_manager` has authority over the functions shown in the diagram below. Any compromise to the `_manager` account may allow the hacker to take advantage of this authority and call `release()`.



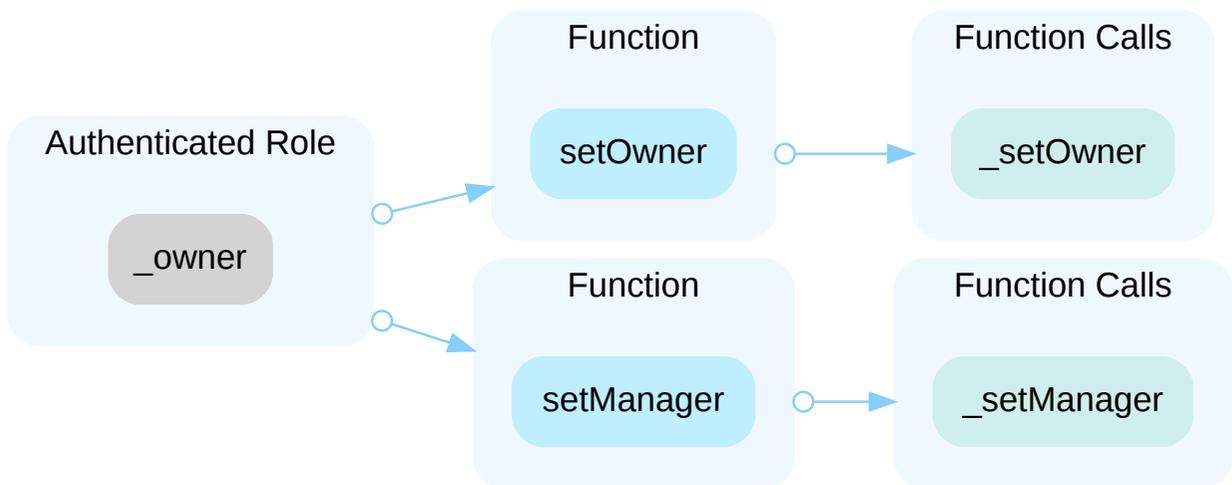
In the contract ERC20 the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the addresses tokens are released to and call `release()`.



In the contract `Lockable` 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 lock or unlock the token and lock or unlock addresses.



In the contract `Managed` 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 change the `_owner` or `_manager`.



If a malicious user get a hold of these private keys, they could make themselves the owner and transfer the accounts to their name. This would allow the malicious user to receive all the funds from each `release()`. They could also advantageously lock the protocol to create a favorable trade when unlocked.

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.

Alleviation

[House of Mandela] : We affirm that, for the avoidance of a single point of key management failure, the owner role will be assigned to multi-signature wallets.

We would also like to clarify that the economics of the token will give certain commercial rewards/discounts to the token users as agreed with this project's partners (some of which are reputable publicly listed corporations), therefore the token issuer must retain the ownership rights and certain privileged roles in order to have sufficient control to be able to fulfill its obligations (which are for the benefit of the token users).

HOM-02 | MULTIPLE FUNCTIONS USE `_allowed`

Category	Severity	Location	Status
Coding Style, Logical Issue	● Medium	token.sol (base): 467	● Resolved

Description

Transferring tokens and burning tokens are separate unique events. The mapping `_allowed` tracks allowances for an external user to perform these functions for another user. In the worst case, if a malicious external contract received permission to transfer tokens then it would be able to burn the user's balance.

Recommendation

It is recommended to track approval of transfers and burning of external tokens separately.

Alleviation

[CertiK]: The client made the recommended changes in the following commit:
[5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

HOM-03 | MISSING CHECKS FOR `newReleaseAddress`

Category	Severity	Location	Status
Volatile Code	Minor	token.sol (base): 570, 583	Resolved

Description

The function `setReleaseAddress1()` ensures with a require statement that the `newReleaseAddress` is not equal to `_releaseAddress2` but does not ensure anything about it not being equal to `_releaseAddress1`.

The function `setReleaseAddress2()` ensures with a require statement that the `newReleaseAddress` is not equal to `_releaseAddress1` but does not ensure anything about it not being equal to `_releaseAddress2`.

Recommendation

We recommend adding a require statement for the `setReleaseAddress1()` function that would ensure the `newReleaseAddress` is not equal to `_releaseAddress1` and a require statement for the `setReleaseAddress2()` function that would ensure the `newReleaseAddress` is not equal to `_releaseAddress2`.

Alleviation

[certik]: The client made the recommended changes in the following commit:

[5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

HOM-04 | UNNECESSARY COMPLICATED EXPRESSION

Category	Severity	Location	Status
Coding Style	● Informational	token.sol (base): 637	● Resolved

Description

A complicated expression `(_releaseAmount * 500000) / 1000000`, which is equivalent to `_releaseAmount / 2`, is used in the code which reduces the code readability.

Recommendation

We recommend replacing the current expression by `releaseAmount / 2`.

Alleviation

[Certik]: The client made the recommended changes in the following commit:
[5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

HOM-05 | TYPO

Category	Severity	Location	Status
Inconsistency	● Informational	token.sol (base): 130, 206, 586	● Resolved

Description

The error message states `_releaseAddress2` instead of `_releaseAddress1`.

The emit and event `OwnershipTransferred` is spelled incorrectly.

Recommendation

We recommend updating the comment that mentions `_releaseAddress1` and updating `OwnershipTransferred` to `OwnershipTransferred`.

Alleviation

[certik]: The client made the recommended changes in the following commit:

[5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

HOM-06 | SUPPLY CAN DIFFER FROM AVAILABLE TOKENS

Category	Severity	Location	Status
Inconsistency	● Informational	token.sol (base): 430	● Pending

Description

The contract locks up some tokens and are locked to X release date. Although, It eventually releases all of them, given the `_owner` or the `_manager` keeps calling the `release()` function, the users still cannot interact with the tokens that are in the contract's balance. The actual number of tokens in the market would be less than what the `totalSupply()` returns.

Recommendation

For transparency sake, it would be beneficial to have another view function to get the total supply at a given time.

Alleviation

The current fix did not answer the problem. We are saying that another view function for the "currentTotalSupply"

HOM-07 | TOO MANY DIGITS

Category	Severity	Location	Status
Coding Style	<input checked="" type="radio"/> Informational	token.sol (base): 345, 346, 370, 371	<input checked="" type="radio"/> Acknowledged

Description

Literals with many digits are difficult to read and review.

Recommendation

We advise the client to use the scientific notation to improve readability.

Alleviation

[certik] : The client acknowledged the finding, but opted not to make any changes.

HOM-08 | TIME IS INCORRECT

Category	Severity	Location	Status
Logical Issue	● Informational	token.sol (base): 345, 346	● Resolved

Description

The time is off by a small amount. We understand this is to make math potentially easier to read.

A week is 604,800 Seconds and 4 years is 126,227,704 seconds.

Recommendation

We recommend rechecking this and confirming this is intended.

Alleviation

[House_of Mandela1a] : We confirm this is intended. The variable names are an approximation of the set time periods, so the public can more easily comprehend them.

OPTIMIZATIONS | HOUSE OF MANDELA TOKEN

ID	Title	Category	Severity	Status
HOM-09	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved

HOM-09 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	token.sol (base): 345, 346	● Resolved

Description

These state variables never change should be declared as `constant` to save gas.

- `_week`
- `_4years`

Recommendation

We recommend adding the `constant` attribute to these state variables.

Alleviation

[Certik] : The client made the recommended changes in the following commit:

[5643aabe64f46025aa21dd02ef4334ede268fbbd](#)

FORMAL VERIFICATION | HOUSE OF MANDELA TOKEN

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-succeed-self	Function <code>transfer</code> Succeeds on Admissible Self Transfers
erc20-transfer-succeed-normal	Function <code>transfer</code> Succeeds on Admissible 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>

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

Property Name	Title
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
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

In the remainder of this section, we list all contracts where model checking of at least one property was not successful. There are several reasons why this could happen:

- Model checking reports a counterexample that violates the property. Depending on the counterexample, this occurs if
 - The specification of the property is too generic and does not accurately capture the intended behavior of the smart contract. In that case, the counterexample does not indicate a problem in the underlying smart contract. We report such instances as being "inapplicable".
 - The property is applicable to the smart contract. In that case, the counterexample showcases a problem in the smart contract and a correspond finding is reported separately in the Findings section of this report. In the following tables, we report such instances as "invalid". The distinction between spurious and actual counterexamples is done manually by the auditors.
- The model checking result is inconclusive. Such a result does not indicate a problem in the underlying smart contract. An inconclusive result may occur if
 - The model checking engine fails to construct a proof. This can happen if the logical deductions necessary are beyond the capabilities of the automated reasoning tool. It is a technical limitation of all proof engines and cannot be avoided in general.
 - The model checking engine runs out of time or memory and did not produce a result. This can happen if automatic abstraction techniques are ineffective or of the state space is too big.

Contract ERC20 (Source File token.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	● True	
erc20-transfer-correct-amount	● True	
erc20-transfer-succeed-self	● False	
erc20-transfer-succeed-normal	● False	
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-succeed-normal	● False	
erc20-transferfrom-correct-amount-self	● True	
erc20-transferfrom-correct-amount	● True	
erc20-transferfrom-succeed-self	● False	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-fail-exceed-allowance	● True	
erc20-transferfrom-change-state	● True	
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-false	● 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 | HOUSE OF MANDELA TOKEN

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.
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.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.

