



Security Assessment: **metazerogg Staking**

April 10, 2024

- Audit Status: **Pass**
- Audit Edition: **Standard**



Project Overview

Token Summary

Parameter	Result
Address	
Name	metazerogg
Token Tracker	metazerogg (meta)
Decimals	0
Supply	
Platform	ETHEREUM
compiler	0.8.24
Contract Name	StakingContract
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
metazerogg		Yes

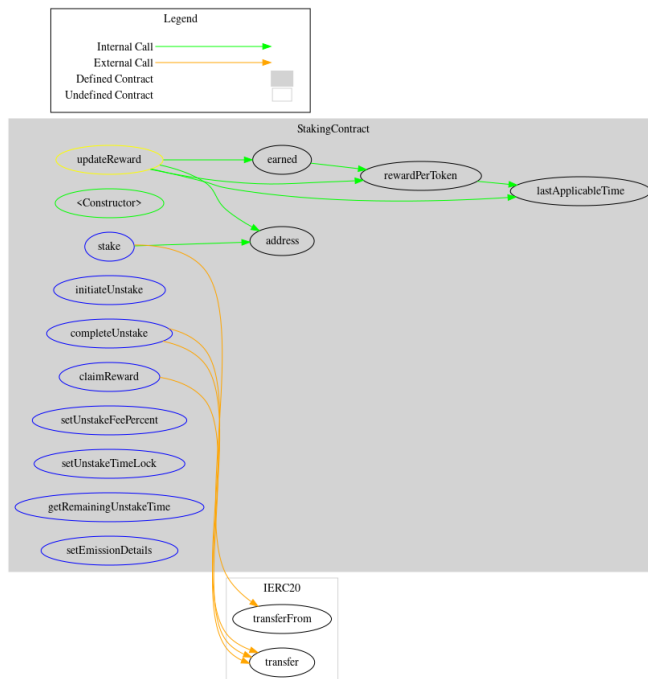
TestNet Contract was Not Assessed

Solidity Code Provided

SolidID	File Sha-1	FileName
Meta	14da0626e1b30ad52b049c42a22ac918d65f098a	StakingContract.sol
Meta		
Meta		
Meta		
Meta		
Meta	undefined	

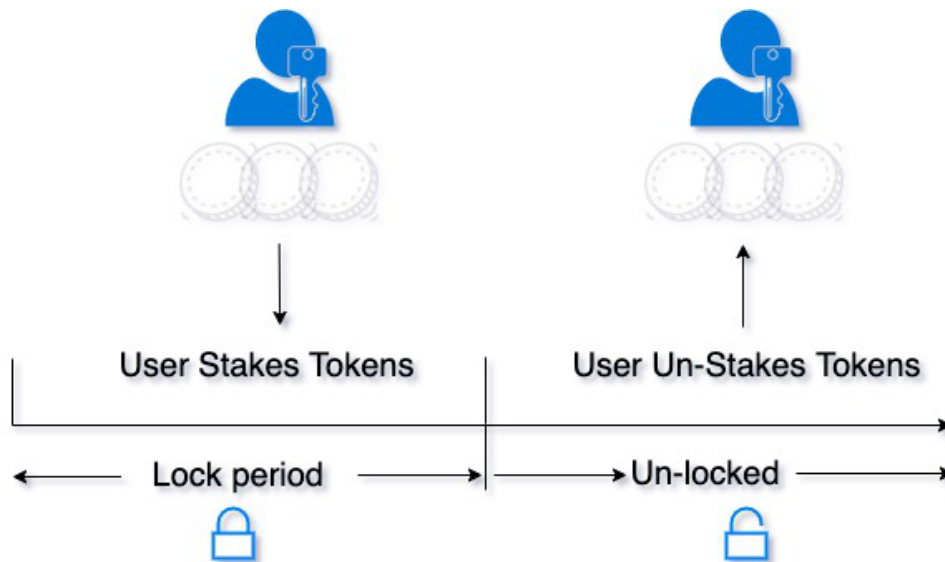
Call Graph

The contract for metazerogg has the following call graph structure.



What is a Staking Contract

A smart contract which allows users to stake and un-stake a specified ERC20 token. Staked tokens are locked for a specific length of time (set by the contract owner at the outset). Once the time period has elapsed, the user can remove their tokens again.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	StakingContract.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	StakingContract.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	StakingContract.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	StakingContract.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	StakingContract.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	StakingContract.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	StakingContract.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	StakingContract.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	StakingContract.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	StakingContract.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	StakingContract.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	StakingContract.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	StakingContract.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	StakingContract.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	StakingContract.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	StakingContract.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	StakingContract.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	StakingContract.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	StakingContract.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	StakingContract.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	StakingContract.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	StakingContract.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	StakingContract.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	StakingContract.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	StakingContract.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	StakingContract.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	StakingContract.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	StakingContract.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	StakingContract.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	StakingContract.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	StakingContract.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	StakingContract.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	StakingContract.sol	L: 0 C: 0

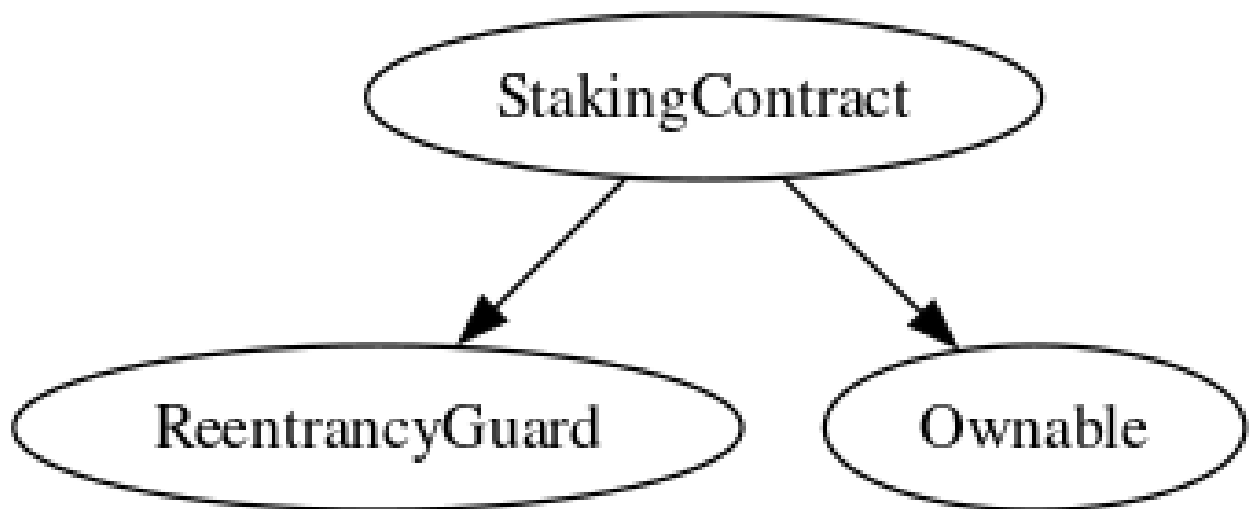
ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	StakingContract.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	StakingContract.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	StakingContract.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	StakingContract.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Inheritance

The contract for metazerogg has the following inheritance structure.

The Project has a Total Supply of





Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
setUnstakeFeePercent	uint256 _newFee	External
setUnstakeTimeLock	uint256 _newTimeLock	External
withdrawFees	uint256 amount	External
withdrawRemainingTokens		External

meta-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Low	StakingContract.sol: L: 186-192 C: 14, L: 168 C: 14, L: 190 C: 14	 Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the missing required function.



Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...  
    require(receiver != address(0), "Receiver is the zero address");  
...  
...  
    require(value X limitation, "Your not able to do this function");  
...
```

We also recommend customer to review the following function that is missing a required validation. missing required function.

meta-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	StakingContract.sol: L: 186 C: 14, L: 192 C: 14	 Detected

Description






Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes. The linked code does not create an event for the transfer.

Remediation






Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Technical Findings Summary

Classification of Risk

Severity	Description
 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.
 High	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.
 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
 Low	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.
 Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
 Critical	0	0	1
 High	0	0	0
 Medium	0	0	0
 Low	2	2	3
 Informational	0	0	0
Total	2	2	4

Social Media Checks

Social Media		URL	Result
Twitter	N/A		No
Other			N/A
Website			N/A
Telegram	N/A		No

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:



Assessment Results

Score Results

Review	Score
Overall Score	86/100
Auditor Score	85/100
Review by Section	Score
Manual Scan Score	15
SWC Scan Score	37
Advance Check Score	34

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Passed



Assessment Results

Important Notes:

- Fixed Issues:
 - Duplicate Variable: `unstakeInitTime` is not declared twice; the issue seems resolved.␣
 - Dead Code: The `claimedAfterUnstake` variable is not present, and `setEmissionDetails` is not in the code.␣
 - Solidity Version: The version is fixed at 0.8.24, mitigating the risk of automatic updates to newer, potentially unstable versions.␣
- Unresolved Issues:
 - Timestamp Dependence: `block.timestamp` is still used for emission and unstaking.␣
 - Owner Privileges: Owner can still set `unstakeFeePercent` and `unstakeTimeLock`.␣
 - Emission Rate Control: No `setEmissionDetails` function present, so emission rate control is not an issue in the provided code.␣
 - Fee Limit Check: `setUnstakeFeePercent` still ensures the fee does not exceed 2%, but allows for owner changes.␣
 - Time Lock Adjustment: Owner can still change the unstake time lock.␣
 - Potential Gas Optimizations: Some optimizations may still be possible.␣

- Centralization Risks: Heavy reliance on owner remains for contract parameters.␣
- ERC20 Compliance: Still assumes basicToken complies with ERC20's transfer and transferFrom return values.␣
- Reward Calculation Accuracy: Still relies on rewardPerTokenStored and lastUpdateTime.␣
- Unstake Logic: No mechanism to cancel an unstake request is present.␣
- Event Emissions: Events are emitted, but additional details could be beneficial.␣
- Contract Visibility: Functions have appropriate visibility.␣
- Fee Accrual: feesAccrued is present, and withdrawFees allows the owner to withdraw accrued fees.␣
- Commenting and Documentation: Some areas could benefit from additional comments.␣
- Naming Conventions: No apparent inconsistencies in the provided code.␣
- withdrawRemainingTokens:␣
- Has a validation to ensure the current time is after the emission end.␣
- However, it lacks a check to ensure that the amount to be withdrawn does not exceed the available balance minus the pending rewards.␣
- ClaimReward (external):␣
- Has a validation for the reward amount but lacks a check to

ensure that the reward does not exceed the contract's available balance for rewards.⌋

- The missing validations could lead to unexpected behavior or errors during execution. It's important to ensure that all state-changing functions have comprehensive validations to maintain contract integrity and prevent misuse or errors.⌋
- Overall, while some issues have been addressed, the contract still exhibits a level of risk associated with owner privileges and the use of block.timestamp. The absence of setEmissionDetails suggests that the emission rate and duration are now fixed, reducing the risk of owner manipulation in this area. However, centralization concerns remain, and there is still room for optimization and improved documentation.

Auditor Score =85
Audit Passed



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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

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 requirements on the input variables than a setter function.

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all the developers.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocacy for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access 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 with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.

