

etherauthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: **MATH Token**
Website: mathwallet.org
Platform: **Base Chain Network**
Language: **Solidity**
Date: **June 18th, 2024**

Table of contents

| | |
|---------------------------------------|----|
| Introduction | 4 |
| Project Background | 4 |
| Audit Scope | 5 |
| Code Audit History | 6 |
| Severity Definitions | 6 |
| Claimed Smart Contract Features | 7 |
| Audit Summary | 8 |
| Technical Quick Stats | 9 |
| Business Risk Analysis | 10 |
| Code Quality | 11 |
| Documentation | 11 |
| Use of Dependencies | 11 |
| AS-IS overview | 12 |
| Audit Findings | 13 |
| Conclusion | 15 |
| Our Methodology | 16 |
| Disclaimers | 18 |
| Appendix | |
| • Code Flow Diagram | 19 |
| • Slither Results Log | 20 |
| • Solidity static analysis | 22 |
| • Solhint Linter | 23 |

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the MATH Token smart contract from mathwallet.org was audited extensively. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 18th, 2024.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Website Details



Math Wallet is a multichain wallet designed for Web3, supporting over 195 public chains including Ethereum, Bitcoin, Solana, and Cosmos. It offers various interfaces such as a mobile app, browser extension, and web wallet. Features include a DApp store, NFT wallet, staking tools, and multi-chain gas tracker. It supports mainstream blockchains, EVM-compatible chains, Substrate-based chains, and CosmosSDK chains. Math Wallet integrates with hardware wallets like Ledger and supports multiple browsers.

Code Details

- This Solidity code defines an ERC20-compatible token named "MATH Token" with the symbol "MATH" and 18 decimals. It includes various functionalities:
 - **SafeMath Library:** Provides safe arithmetic operations to prevent overflow and underflow.
 - **BasicToken and StandardToken Contracts:** Implement basic ERC20 functions like balance management, transfer, and approval.
 - **Ownable Contract:** Defines ownership and transfer of ownership.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

- **MintableToken Contract:** Allows minting of new tokens, with a cap on total supply.
- **PausableToken Contract:** Allows pausing and unpausing of token transfers.
- **MATHToken Contract:** Combines all functionalities and sets the total supply to 200 million tokens.

Audit scope

| | |
|---------------------|---|
| Name | Code Review and Security Analysis Report for MATH Token Smart Contract |
| Platform | Base Chain Network |
| Language | Solidity |
| File | MATHToken.sol |
| Smart Contract Code | 0x9e81f6495ba29a6b4d48bddd042c0598fa8abc9f |
| Audit Date | June 18th,2024 |
| Audit Result | Passed |

Code Audit History



1
Total Findings

0
Critical






0
High

0
Medium

0
Low

1
Informational

Severity Definitions

| | | | |
|---|---|---|---|
| 0 |  | Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| 0 |  | High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. Public access is crucial. |
| 0 |  | Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens loss |
| 0 |  | Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets, that can't have a significant impact on execution |
| 1 |  | Lowest / Informational / Best Practice | Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored. |

Claimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
|---|--|
| <p>Tokenomics:</p> <ul style="list-style-type: none">• Name: MATH Token• Symbol: MATH• Decimals: 18• Total Supply: 0.2 billion• Mint Total: 5 million | <p>YES, This is valid.</p> |
| <p>Owner Specifications:</p> <ul style="list-style-type: none">• Allows pausing and unpausing of token transfers.• Allows minting of new tokens.• Allows the current owner to transfer control of the contract to a new owner. | <p>YES, This is valid.</p> <p>We advise renouncing ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.</p> |

Audit Summary

According to the standard audit assessment, the Customer's solidity-based smart contracts are "**Secured**". Also, these contracts contain owner control, which does not make them fully decentralized.



You are here 

We used various tools like Slither, Solhint, and Remix IDE. At the same time, this finding is based on a critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit Overview section. The general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 0 low, and 1 very low-level issue.

Investor Advice: A technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner-controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

| Main Category | Subcategory | Result |
|----------------------|---|-----------|
| Contract Programming | The solidity version is not specified | Passed |
| | The solidity version is too old | Passed |
| | Integer overflow/underflow | Passed |
| | Function input parameters lack check | Passed |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Passed |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | N/A |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Features claimed | Passed |
| | Other programming issues | Moderated |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Unused code | Passed |
| Gas Optimization | "Out of Gas" Issue | Passed |
| | High consumption 'for/while' loop | Passed |
| | High consumption 'storage' storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage is not set | Passed |
| | "Short Address" Attack | Passed |
| | "Double Spend" Attack | Passed |

Overall Audit Result: **PASSED**

Business Risk Analysis

| Category | Result |
|---------------------------|--------------|
| ● Buy Tax | 0% |
| ● Sell Tax | 0% |
| ● Cannot Buy | No |
| ● Cannot Sell | No |
| ● Max Tax | 0% |
| ● Modify Tax | No |
| ● Fee Check | Not Detected |
| ● Is Honeypot | Not Detected |
| ● Trading Cooldown | Not Detected |
| ● Can Pause Trade? | Yes |
| ● Pause Transfer? | Yes |
| ● Max Tax? | No |
| ● Is it Anti-whale? | Not Detected |
| ● Is Anti-bot? | Not Detected |
| ● Is it a Blacklist? | No |
| ● Blacklist Check | No |
| ● Can Mint? | Yes |
| ● Is it a Proxy Contract? | No |
| ● Is it used Open Source? | No |
| ● External Call Risk? | No |
| ● Balance Modifiable? | No |
| ● Can Take Ownership? | Yes |
| ● Ownership Renounce? | No |
| ● Hidden Owner? | Not Detected |
| ● Self Destruction? | Not Detected |
| ● Auditor Confidence | High |

Overall Audit Result: PASSED

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract. Smart contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in MATH Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties/methods can be reused many times by other contracts in the MATH Token.

The EtherAuthority team has not provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a MATH Token smart contract code in the form of a [basescan](#) web link.

As mentioned above, code parts are well commented on. And the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries used in this smart contract infrastructure are based on well-known industry standard open-source projects.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

MATHToken.sol

Functions

| Sl. | Functions | Type | Observation | Conclusion |
|-----|-------------------|----------|---------------------|------------|
| 1 | constructor | write | Passed | No Issue |
| 2 | transfer | write | Passed | No Issue |
| 3 | transferFrom | write | Passed | No Issue |
| 4 | approve | write | Passed | No Issue |
| 5 | increaseApproval | write | Passed | No Issue |
| 6 | decreaseApproval | write | Passed | No Issue |
| 7 | canMint | modifier | Passed | No Issue |
| 8 | hasMintPermission | modifier | Passed | No Issue |
| 9 | mint | write | has Mint Permission | No Issue |
| 10 | whenNotPaused | modifier | Passed | No Issue |
| 11 | whenPaused | modifier | Passed | No Issue |
| 12 | pause | write | access only Owner | No Issue |
| 13 | unpause | write | access only Owner | No Issue |
| 14 | onlyOwner | modifier | Passed | No Issue |
| 15 | transferOwnership | write | access only Owner | No Issue |

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium-severity vulnerabilities were found.

Low

No Low-severity vulnerabilities were found.

Very Low / Informational / Best practices:

[I-01] Make variables constant:

```
// public variables
string public name = "MATH Token";
string public symbol = "MATH";
uint8 public decimals = 18;
```

Description:

These variable values will remain unchanged.

Recommendation: We suggest making them constant. It is best practice and it also saves some gas. Just add a constant keyword.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet's private key would be compromised, then it would create trouble.



The following are Owner functions:

Pausable.sol

- pause: Allows pausing of token transfers only by the owner.
- unpause: Allows unpausing of token transfers only by the owner.

MintableToken.sol

- mint: Allows minting of new tokens only by the owner.

Ownable.sol

- transferOwnership: Allows the current owner to transfer control of the contract to a newOwner.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a [basescan](#) web link. And we have used all possible tests based on given objects as files. We observed 1 Informational issue in the smart contracts. but those are not critical. So, **it's good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover the maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed smart contract, based on standard audit procedure scope, is **“Secured”**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of the systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and white box penetration testing. We look at the project's website to get a high-level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

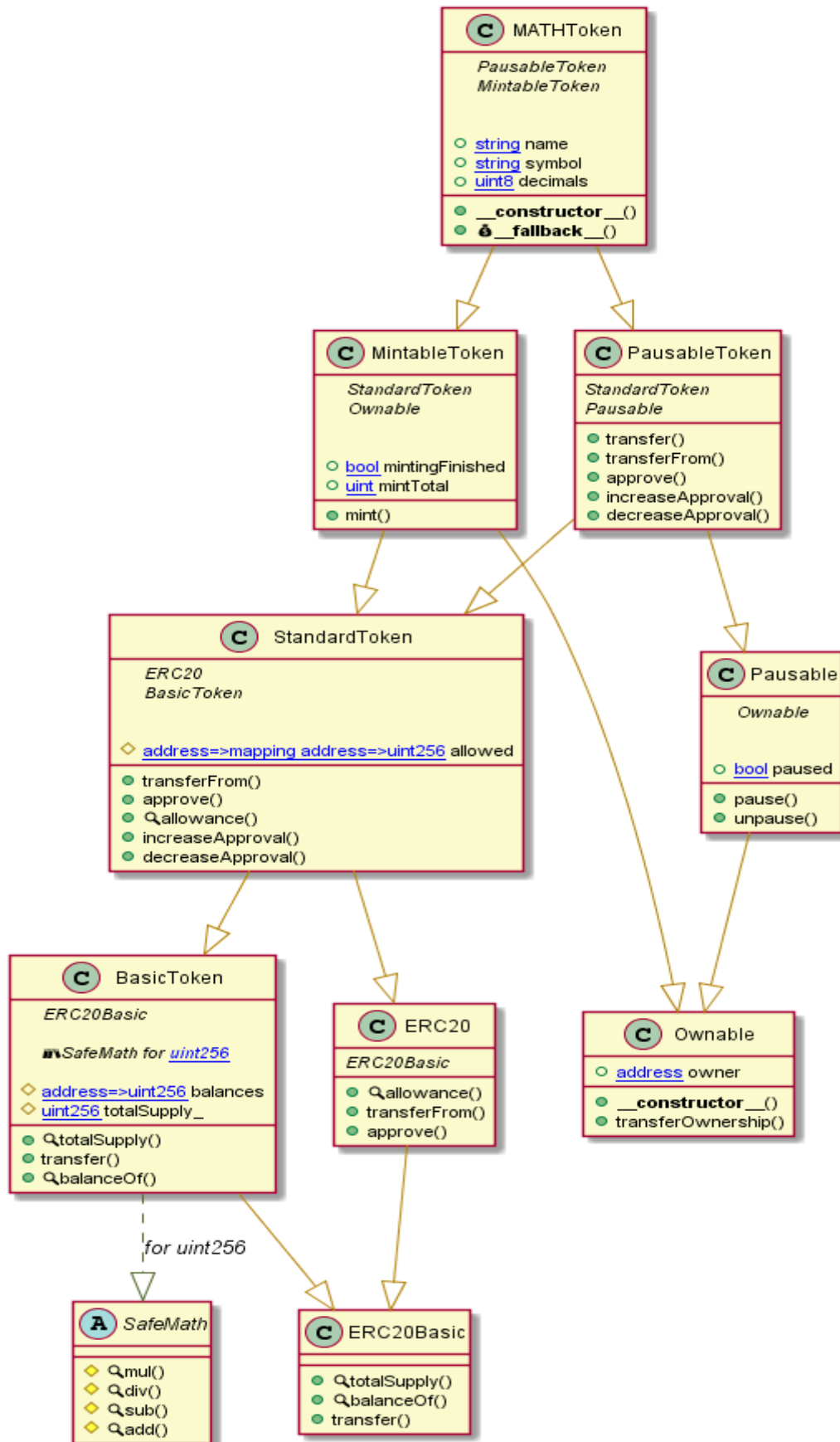
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - MATH Token



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

MATHToken.sol

```
INFO:Detectors:
SafeMath.div(uint256,uint256) (MATHToken.sol#30-35) is never used and should be removed
SafeMath.mul(uint256,uint256) (MATHToken.sol#18-25) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.4.23 (MATHToken.sol#6) allows old versions
solc-0.4.26 is not recommended for deployment
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter BasicToken.transfer(address,uint256)._to (MATHToken.sol#90) is not in mixedCase
Parameter BasicToken.transfer(address,uint256)._value (MATHToken.sol#90) is not in mixedCase
Parameter BasicToken.balanceOf(address)._owner (MATHToken.sol#105) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._from (MATHToken.sol#147) is
not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._to (MATHToken.sol#148) is
not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._value (MATHToken.sol#149)
is not in mixedCase
Parameter StandardToken.approve(address,uint256)._spender (MATHToken.sol#175) is not in
mixedCase
Parameter StandardToken.approve(address,uint256)._value (MATHToken.sol#175) is not in
mixedCase
Parameter StandardToken.allowance(address,address)._owner (MATHToken.sol#188) is not in
mixedCase
Parameter StandardToken.allowance(address,address)._spender (MATHToken.sol#189) is not in
mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._spender (MATHToken.sol#209) is
not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._addedValue
(MATHToken.sol#210) is not in mixedCase
```

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Parameter StandardToken.decreaseApproval(address,uint256)._spender (MATHToken.sol#232) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._subtractedValue (MATHToken.sol#233) is not in mixedCase
Parameter MintableToken.mint(address,uint256)._to (MATHToken.sol#322) is not in mixedCase
Parameter MintableToken.mint(address,uint256)._amount (MATHToken.sol#323) is not in mixedCase
Parameter PausableToken.transfer(address,uint256)._to (MATHToken.sol#393) is not in mixedCase
Parameter PausableToken.transfer(address,uint256)._value (MATHToken.sol#394) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._from (MATHToken.sol#404) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._to (MATHToken.sol#405) is not in mixedCase
Parameter PausableToken.transferFrom(address,address,uint256)._value (MATHToken.sol#406) is not in mixedCase
Parameter PausableToken.approve(address,uint256)._spender (MATHToken.sol#416) is not in mixedCase
Parameter PausableToken.approve(address,uint256)._value (MATHToken.sol#417) is not in mixedCase
Parameter PausableToken.increaseApproval(address,uint256)._spender (MATHToken.sol#427) is not in mixedCase
Parameter PausableToken.increaseApproval(address,uint256)._addedValue (MATHToken.sol#428) is not in mixedCase
Parameter PausableToken.decreaseApproval(address,uint256)._spender (MATHToken.sol#438) is not in mixedCase
Parameter PausableToken.decreaseApproval(address,uint256)._subtractedValue (MATHToken.sol#439) is not in mixedCase
Reference:
<https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>
INFO:Detectors:
MATHToken.constructor() (MATHToken.sol#455-457) uses literals with too many digits:
- totalSupply_ = 200000000 * (10 ** uint256(decimals)) (MATHToken.sol#456)
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits>
INFO:Detectors:
MATHToken.decimals (MATHToken.sol#453) should be constant
MATHToken.name (MATHToken.sol#451) should be constant
MATHToken.symbol (MATHToken.sol#452) should be constant
MintableToken.mintingFinished (MATHToken.sol#302) should be constant
Reference:
<https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant>
INFO:Slither:MATHToken.sol analyzed (10 contracts with 93 detectors), 37 result(s) found

Solidity Static Analysis

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

MATHToken.sol

Gas costs:

Gas requirement of function MATHToken.transferFrom is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 146:2:

Gas costs:

Gas requirement of function MATHToken.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 321:2:

Constant/View/Pure functions:

MATHToken() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

Pos: 455:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

Pos: 364:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. $10 / 100 = 0$ instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 34:11:

Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

MATHToken.sol

```
Compiler version ^0.4.23 does not satisfy the ^0.5.8 semver requirement
Pos: 1:5
Explicitly mark visibility of state
Pos: 3:73
Explicitly mark visibility of state
Pos: 3:75
Provide an error message for require
Pos: 5:90
Provide an error message for require
Pos: 5:91
Provide an error message for require
Pos: 5:153
Provide an error message for require
Pos: 5:154
Provide an error message for require
Pos: 5:155
Provide an error message for require
Pos: 5:277
Provide an error message for require
Pos: 5:286
Provide an error message for require
Pos: 5:305
Provide an error message for require
Pos: 5:310
Visibility modifier must be first in list of modifiers
Pos: 5:326
Provide an error message for require
Pos: 5:330
Provide an error message for require
Pos: 5:355
Provide an error message for require
Pos: 5:363
Visibility modifier must be first in list of modifiers
Pos: 44:370
Visibility modifier must be first in list of modifiers
Pos: 43:378
```

Provide an error message for revert

Pos: 9:459

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io