Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	WP Smart Contracts
Website:	wpsmartcontracts.com
Platform:	ETH, BSC
Language:	Solidity
Date:	April 29th, 2022

Table of contents

Email: audit@EtherAuthority.io

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.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the WP Smart Contracts team to perform the Security audit of the Matcha(ERC721) and Almond(staking) smart contracts code. 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 April 29th, 2022.

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

- The WP Smart Contracts ecosystem is expanding with more than a dozen Ethereum Virtual Machine (EVM) compatible networks.
- The WP Smart Contracts provides the smart contract solutions to the wordpress users. They develop various WP plugins which lets WP websites use the smart contract deployment quickly. We audited their Matcha(ERC721) and Almond(staking) smart contracts.

Audit scope

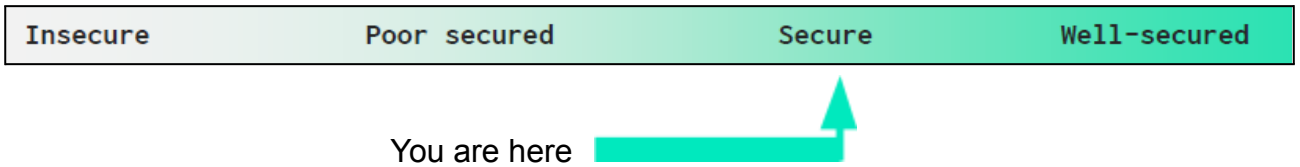
Name	Code Review and Security Analysis Report for WP Smart Contracts Protocol Smart Contracts
Platform	**Multiple blockchain platforms / Solidity**
File 1	StakesAlmond.sol
File 1 MD5 Hash	77BF055AB6BE4B98AB710BE95CB2D218
File 2	ERC721Matcha.sol
File 2 MD5 Hash	EB0440D80D8FF6CCE9AB229C5602A7B4
Audit Date	April 29th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Almond- Stakes.sol • StakesAlmond has functions like: start, end, set, ledger_length, etc.	**YES, This is valid.**
ERC721Matcha.sol • ERC721Matcha has functions like: highestBid, auctionFinalize, canFinalize, canWithdraw, etc.	**YES, This is valid.**

Audit Summary

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



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on 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. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 0 low and some very low level issues.

Investors Advice: 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	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	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 not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the WP Smart Contracts Protocol 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 WP Smart Contracts Protocol.

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

Some code parts are not well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a WP Smart Contracts Protocol smart contract code in the form of a BSCScan web link and Etherscan web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. But 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.

Another source of information was its official website https://wpsmartcontracts.com which provided rich information about the project architecture and tokenomics.

Use of Dependencies

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

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

StakesAlmond.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	isOwner	modifier	Passed	No Issue
3	changeOwner	write	access by isOwner	No Issue
4	getOwner	read	Passed	No Issue
5	nonReentrant	modifier	Passed	No Issue
6	start	external	Passed	No Issue
7	end	write	Passed	No Issue
8	set	write	access by isOwner	No Issue
9	get_gains	read	Passed	No Issue
10	get_gains2	read	Passed	No Issue
11	ledger_length	read	Passed	No Issue

ERC721Matcha.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	exists	read	Passed	No Issue
3	tokensOfOwner	read	Passed	No Issue
4	setTokenURI	write	Passed	No Issue
5	autoMint	write	access only Minter	No Issue
6	transfer	write	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	canSell	read	Passed	No Issue
9	sell	write	Passed	No Issue
10	getPrice	read	Passed	No Issue
11	canBuy	read	Passed	No Issue
12	buy	write	Passed	No Issue
13	canAuction	read	Passed	No Issue
14	createAuction	write	Passed	No Issue
15	canBid	read	Passed	No Issue
16	bid	write	Passed	No Issue
17	canWithdraw	read	Passed	No Issue
18	withdraw	write	Passed	No Issue
19	canFinalize	read	Passed	No Issue
20	auctionFinalize	write	Passed	No Issue
21	highestBidder	read	Passed	No Issue
22	highestBid	read	Passed	No Issue
23	callOptionalReturn	write	Passed	No Issue
24	updateAdmin	write	Passed	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

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

(1) data - **ERC721Matcha.sol**
data.

Resolution: data.

Very Low / Informational / Best practices:

Centralization

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

- set: StakesAlmond owner can set values like: ratio1 ,ratio2, lower amount, maturity rate, interest rate, penalization, etc.

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 files. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts, but those issues are not critical ones. **So, the smart contracts are ready for the mainnet deployment.**

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 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.

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

Security state of the reviewed 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 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 whitebox penetration testing. We look at the project's web site 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.

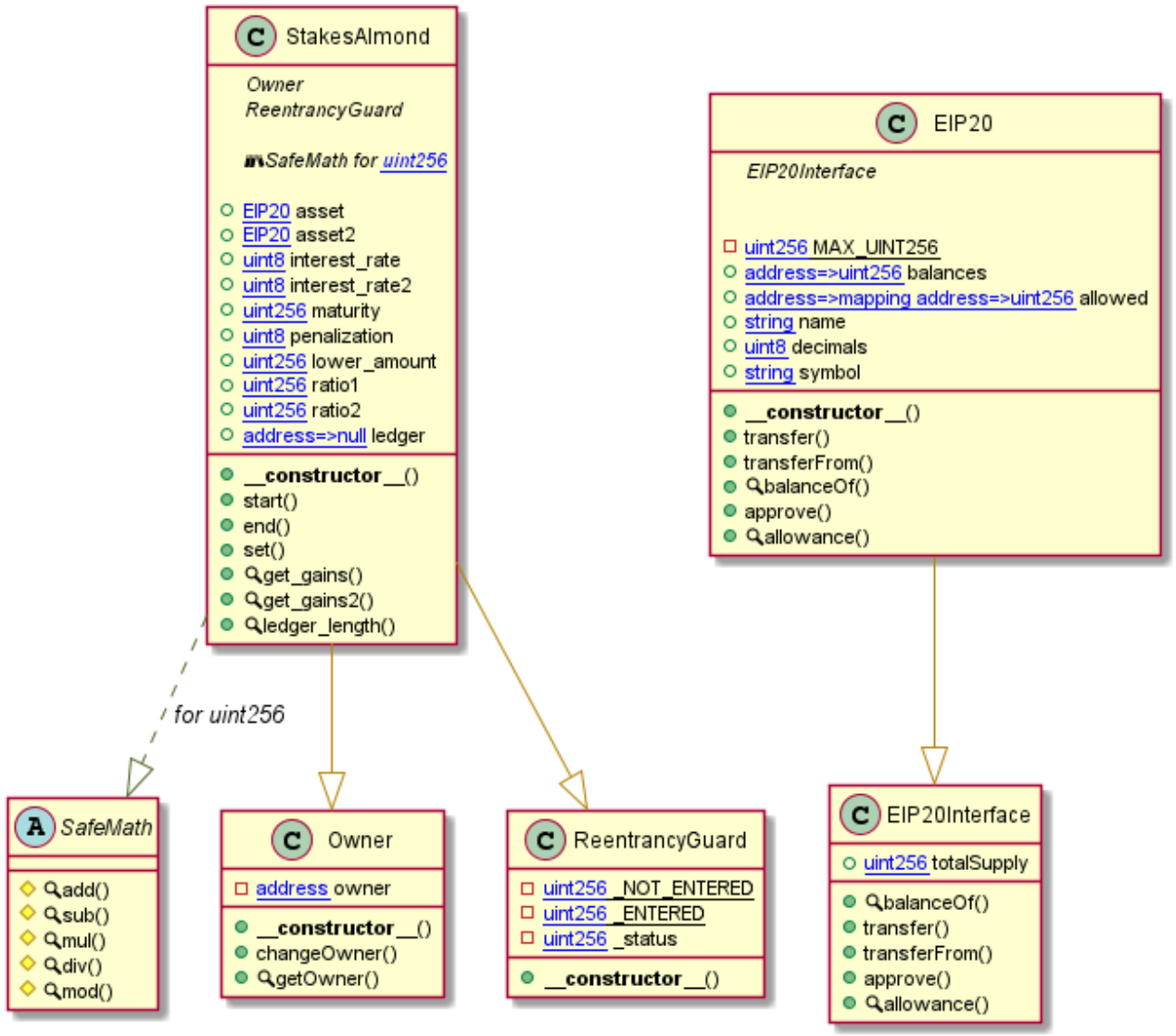
Technical Disclaimer

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.

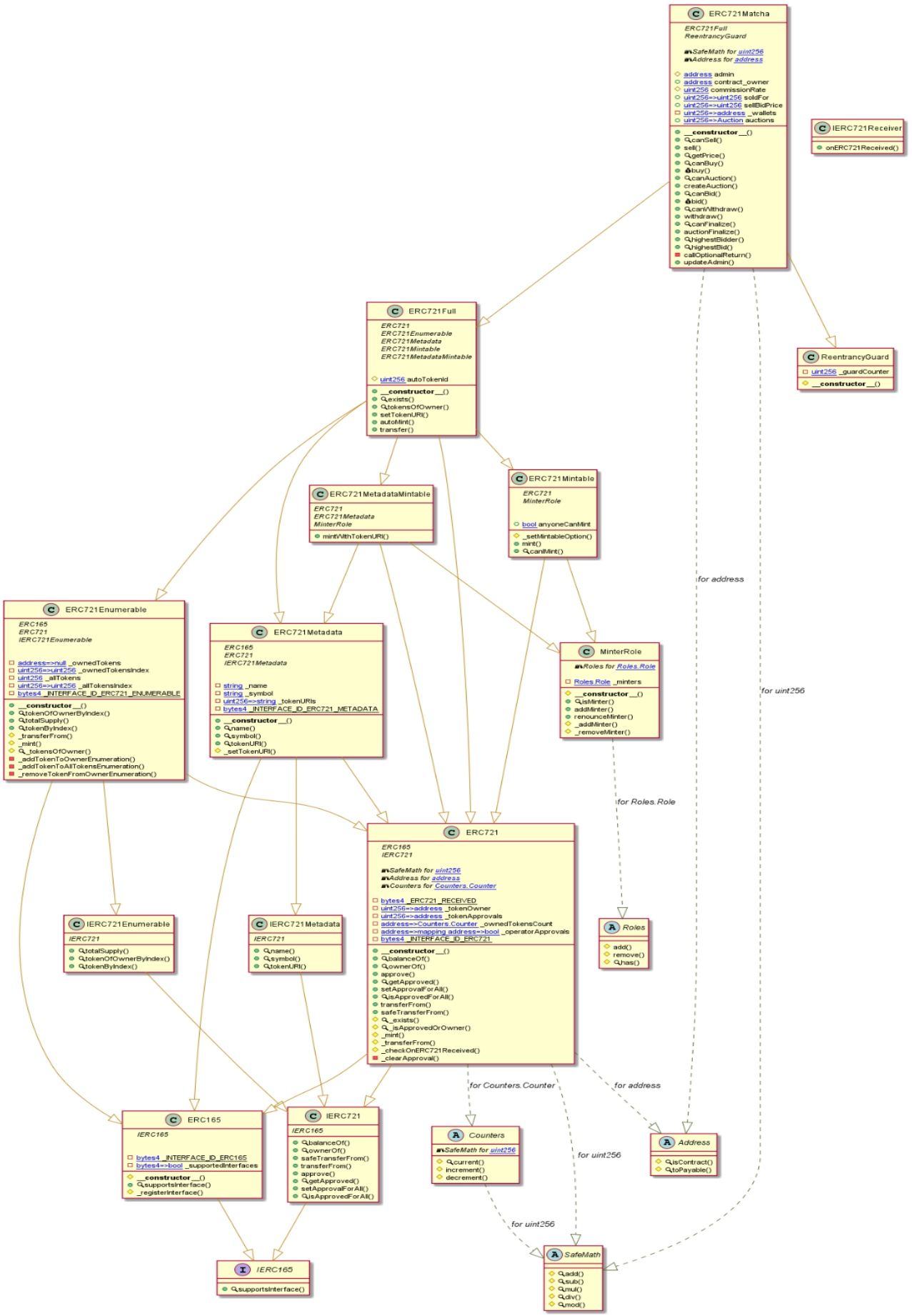
Appendix

Code Flow Diagram - WP Smart Contracts Protocol

StakesAlmond Diagram



ERC721Matcha Diagram



Slither Results Log

Slither log >> StakesAlmond.sol

```
INFO:Detectors:
StakesAlmond.set(EIP20,EIP20,uint256,uint256,uint8,uint8,uint8,uint256,uint256) (StakesAlmond.sol#562-573) should emit an event
 for:
        - ratio1 = _ratio1 (StakesAlmond.sol#566)
        - ratio2 = _ratio2 (StakesAlmond.sol#567)
        - lower_amount = _lower (StakesAlmond.sol#568)
        - interest_rate = _rate (StakesAlmond.sol#570)
        - interest_rate2 = _rate2 (StakesAlmond.sol#571)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Owner.constructor(address)._owner (StakesAlmond.sol#237) lacks a zero-check on :
                - owner = _owner (StakesAlmond.sol#238)
Owner.changeOwner(address).newOwner (StakesAlmond.sol#246) lacks a zero-check on :
                - owner = newOwner (StakesAlmond.sol#248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StakesAlmond.start(uint256) (StakesAlmond.sol#505-510):
        External calls:
        - asset.transferFrom(msg.sender,address(this),_value) (StakesAlmond.sol#507)
        State variables written after the call(s):
        - ledger[msg.sender].push(Record(block.timestamp,_value,0,0,0,0,false)) (StakesAlmond.sol#508)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StakesAlmond.end(uint256) (StakesAlmond.sol#512-560):
        External calls:
        - asset.transfer(msg.sender,ledger[msg.sender][i].amount.sub(_penalization)) (StakesAlmond.sol#521)
        - asset.transfer(getOwner(),_penalization) (StakesAlmond.sol#522)
        Event emitted after the call(s):
        - StakeEnd(msg.sender,ledger[msg.sender][i].amount,_penalization,0,i) (StakesAlmond.sol#526)
Reentrancy in StakesAlmond.end(uint256) (StakesAlmond.sol#512-560):
        External calls:
        - asset.transferFrom(getOwner(),msg.sender,_interest) (StakesAlmond.sol#536)
        - asset2.transferFrom(getOwner(),msg.sender,_interest2) (StakesAlmond.sol#546)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakesAlmond.end(uint256) (StakesAlmond.sol#512-560) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp.sub(ledger[msg.sender][i].from) < maturity (StakesAlmond.sol#518)
        - _interest > 0 && asset.allowance(getOwner(),address(this)) >= _interest && asset.balanceOf(getOwner()) >= _interest (
StakesAlmond.sol#535)
        - _interest2 > 0 && asset2.allowance(getOwner(),address(this)) >= _interest2 && asset2.balanceOf(getOwner()) >= _intere
st2 (StakesAlmond.sol#545)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
StakesAlmond.end(uint256) (StakesAlmond.sol#512-560) compares to a boolean constant:
        -require(bool,string)(ledger[msg.sender][i].ended == false,Invalid stake) (StakesAlmond.sol#515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
SafeMath.add(uint256,uint256) (StakesAlmond.sol#86-91) is never used and should be removed
SafeMath.mod(uint256,uint256) (StakesAlmond.sol#191-193) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (StakesAlmond.sol#206-209) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (StakesAlmond.sol#60) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter EIP20.transfer(address,uint256)._to (StakesAlmond.sol#402) is not in mixedCase
Parameter EIP20.transfer(address,uint256)._value (StakesAlmond.sol#402) is not in mixedCase
```

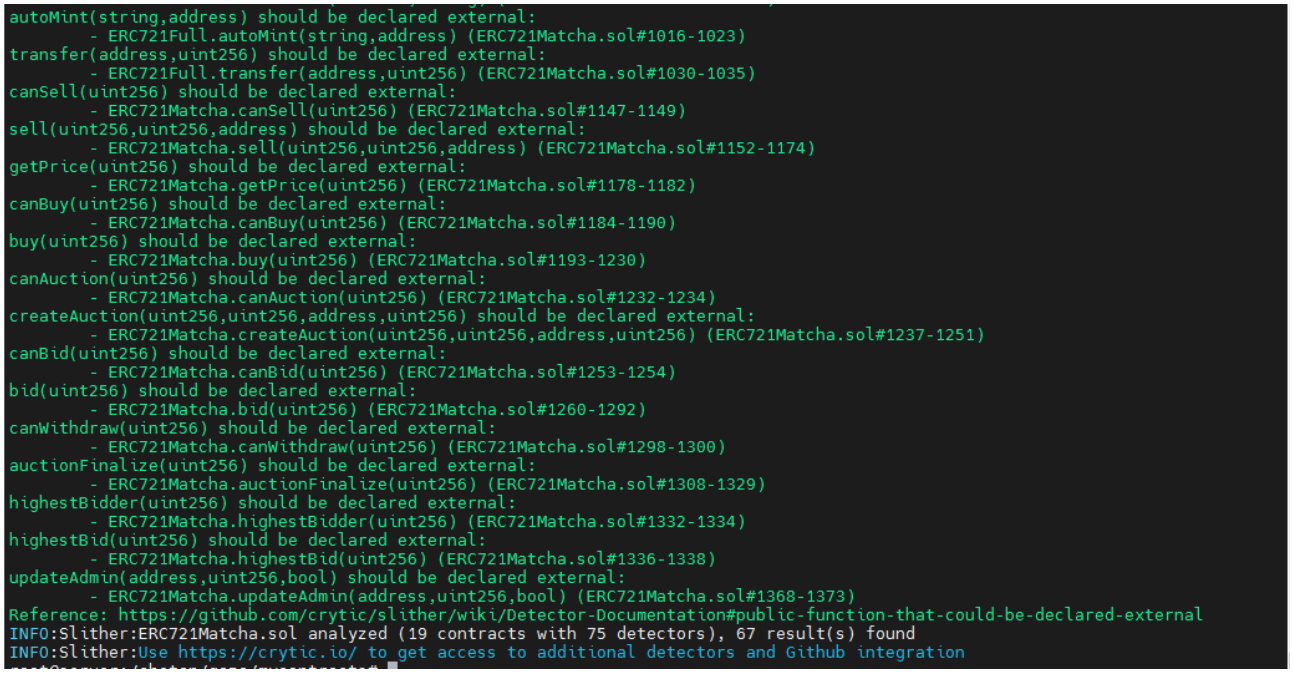
```
Parameter StakesAlmond.get_gains(address,uint256)._rec_number (StakesAlmond.sol#576) is not in mixedCase
Function StakesAlmond.get_gains2(address,uint256) (StakesAlmond.sol#585-598) is not in mixedCase
Parameter StakesAlmond.get_gains2(address,uint256)._address (StakesAlmond.sol#585) is not in mixedCase
Parameter StakesAlmond.get_gains2(address,uint256)._rec_number (StakesAlmond.sol#585) is not in mixedCase
Function StakesAlmond.ledger_length(address) (StakesAlmond.sol#600-603) is not in mixedCase
Parameter StakesAlmond.ledger_length(address)._address (StakesAlmond.sol#600) is not in mixedCase
Variable StakesAlmond.interest_rate (StakesAlmond.sol#472) is not in mixedCase
Variable StakesAlmond.interest_rate2 (StakesAlmond.sol#473) is not in mixedCase
Variable StakesAlmond.lower_amount (StakesAlmond.sol#476) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
changeOwner(address) should be declared external:
        - Owner.changeOwner(address) (StakesAlmond.sol#246-249)
balanceOf(address) should be declared external:
        - EIP20.balanceOf(address) (StakesAlmond.sol#422-424)
        - EIP20Interface.balanceOf(address) (StakesAlmond.sol#339)
transfer(address,uint256) should be declared external:
        - EIP20.transfer(address,uint256) (StakesAlmond.sol#402-408)
        - EIP20Interface.transfer(address,uint256) (StakesAlmond.sol#345)
transferFrom(address,address,uint256) should be declared external:
        - EIP20.transferFrom(address,address,uint256) (StakesAlmond.sol#410-420)
        - EIP20Interface.transferFrom(address,address,uint256) (StakesAlmond.sol#352)
approve(address,uint256) should be declared external:
        - EIP20.approve(address,uint256) (StakesAlmond.sol#426-430)
        - EIP20Interface.approve(address,uint256) (StakesAlmond.sol#358)
allowance(address,address) should be declared external:
        - EIP20.allowance(address,address) (StakesAlmond.sol#432-434)
        - EIP20Interface.allowance(address,address) (StakesAlmond.sol#363)
set(EIP20,EIP20,uint256,uint256,uint8,uint8,uint8,uint256,uint256) should be declared external:
        - StakesAlmond.set(EIP20,EIP20,uint256,uint256,uint8,uint8,uint8,uint256,uint256) (StakesAlmond.sol#562-573)
ledger_length(address) should be declared external:
        - StakesAlmond.ledger_length(address) (StakesAlmond.sol#600-603)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StakesAlmond.sol analyzed (6 contracts with 75 detectors), 64 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ERC721Matcha.sol

```
INFO:Detectors:
ERC721MetadataMintable.mintWithTokenURI(address,uint256,string).tokenURI (ERC721Matcha.sol#983) shadows:
        - ERC721Metadata.tokenURI(uint256) (ERC721Matcha.sol#953-956) (function)
        - IERC721Metadata.tokenURI(uint256) (ERC721Matcha.sol#906) (function)
ERC721Full.autoMint(string,address).tokenURI (ERC721Matcha.sol#1016) shadows:
        - ERC721Metadata.tokenURI(uint256) (ERC721Matcha.sol#953-956) (function)
        - IERC721Metadata.tokenURI(uint256) (ERC721Matcha.sol#906) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ERC721Matcha.updateAdmin(address,uint256,bool) (ERC721Matcha.sol#1368-1373) should emit an event for:
        - commissionRate = _commissionRate (ERC721Matcha.sol#1371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
ERC721Matcha.updateAdmin(address,uint256,bool)._admin (ERC721Matcha.sol#1368) lacks a zero-check on :
                - admin = _admin (ERC721Matcha.sol#1370)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (ERC721Matcha.sol#254-270) uses assembly
        - INLINE ASM (ERC721Matcha.sol#266-268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.isContract(address) (ERC721Matcha.sol#254-270) is never used and should be removed
Address.toPayable(address) (ERC721Matcha.sol#276-282) is never used and should be removed
ERC165._registerInterface(bytes4) (ERC721Matcha.sol#361-364) is never used and should be removed
ERC721._checkOnERC721Received(address,address,uint256,bytes) (ERC721Matcha.sol#618-636) is never used and should be removed
ERC721._isApprovedOrOwner(address,uint256) (ERC721Matcha.sol#548-561) is never used and should be removed
ERC721Enumerable._addTokenToAllTokensEnumeration(uint256) (ERC721Matcha.sol#882-885) is never used and should be removed
ERC721Enumerable._addTokenToOwnerEnumeration(address,uint256) (ERC721Matcha.sol#873-876) is never used and should be removed
ERC721Matcha.callOptionalReturn(IERC721,bytes) (ERC721Matcha.sol#1346-1365) is never used and should be removed
ERC721Mintable._setMintableOption(bool) (ERC721Matcha.sol#741-743) is never used and should be removed
SafeMath.add(uint256,uint256) (ERC721Matcha.sol#156-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (ERC721Matcha.sol#233-236) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (ERC721Matcha.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in ERC721Matcha.callOptionalReturn(IERC721,bytes) (ERC721Matcha.sol#1346-1365):
        - (success,returndata) = address(token).call(data) (ERC721Matcha.sol#1358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ERC721Matcha.createAuction(uint256,uint256,address,uint256)._closingTime (ERC721Matcha.sol#1237) is not in mixedCase
Parameter ERC721Matcha.createAuction(uint256,uint256,address,uint256)._beneficiary (ERC721Matcha.sol#1237) is not in mixedCase
Parameter ERC721Matcha.createAuction(uint256,uint256,address,uint256)._reservePrice (ERC721Matcha.sol#1237) is not in mixedCase
Parameter ERC721Matcha.updateAdmin(address,uint256,bool)._admin (ERC721Matcha.sol#1368) is not in mixedCase
Parameter ERC721Matcha.updateAdmin(address,uint256,bool)._commissionRate (ERC721Matcha.sol#1368) is not in mixedCase
Parameter ERC721Matcha.updateAdmin(address,uint256,bool)._anyoneCanMint (ERC721Matcha.sol#1368) is not in mixedCase
Variable ERC721Matcha.contract_owner (ERC721Matcha.sol#1085) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ERC721Matcha (ERC721Matcha.sol#1076-1376) does not implement functions:
        - IERC721.safeTransferFrom(address,address,uint256,bytes) (ERC721Matcha.sol#97-102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ERC165._INTERFACE_ID_ERC165 (ERC721Matcha.sol#329) is never used in ERC721Matcha (ERC721Matcha.sol#1076-1376)
ERC721._INTERFACE_ID_ERC721 (ERC721Matcha.sol#406) is never used in ERC721Matcha (ERC721Matcha.sol#1076-1376)
ERC721Enumerable._INTERFACE_ID_ERC721_ENUMERABLE (ERC721Matcha.sol#805) is never used in ERC721Matcha (ERC721Matcha.sol#1076-13
76)
ERC721Metadata._INTERFACE_ID_ERC721_METADATA (ERC721Matcha.sol#926) is never used in ERC721Matcha (ERC721Matcha.sol#1076-1376)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
ERC721Matcha.contract_owner (ERC721Matcha.sol#1085) should be constant
ERC721Metadata._name (ERC721Matcha.sol#911) should be constant
ERC721Metadata._symbol (ERC721Matcha.sol#914) should be constant
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (ERC721Matcha.sol#522-528)
        - IERC721.safeTransferFrom(address,address,uint256) (ERC721Matcha.sol#68-72)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (ERC721Matcha.sol#481-486)
        - IERC721.setApprovalForAll(address,bool) (ERC721Matcha.sol#90)
onERC721Received(address,address,uint256,bytes) should be declared external:
        - IERC721Receiver.onERC721Received(address,address,uint256,bytes) (ERC721Matcha.sol#125-130)
addMinter(address) should be declared external:
        - MinterRole.addMinter(address) (ERC721Matcha.sol#710-712)
renounceMinter() should be declared external:
        - MinterRole.renounceMinter() (ERC721Matcha.sol#714-716)
mint(address,uint256) should be declared external:
        - ERC721Mintable.mint(address,uint256) (ERC721Matcha.sol#751-758)
canIMint() should be declared external:
        - ERC721Mintable.canIMint() (ERC721Matcha.sol#760-762)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (ERC721Matcha.sol#818-821)
        - IERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (ERC721Matcha.sol#776)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (ERC721Matcha.sol#837-840)
        - IERC721Enumerable.tokenByIndex(uint256) (ERC721Matcha.sol#778)
mintWithTokenURI(address,uint256,string) should be declared external:
        - ERC721MetadataMintable.mintWithTokenURI(address,uint256,string) (ERC721Matcha.sol#983-987)
exists(uint256) should be declared external:
        - ERC721Full.exists(uint256) (ERC721Matcha.sol#999-1001)
tokensOfOwner(address) should be declared external:
        - ERC721Full.tokensOfOwner(address) (ERC721Matcha.sol#1003-1005)
setTokenURI(uint256,string) should be declared external:
        - ERC721Full.setTokenURI(uint256,string) (ERC721Matcha.sol#1007-1009)
autoMint(string,address) should be declared external:
        - ERC721Full.autoMint(string,address) (ERC721Matcha.sol#1016-1023)
```



Solidity Static Analysis

StakesAlmond.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in StakesAlmond.start(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 505:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 555:39:

Gas & Economy

Gas costs:

Gas requirement of function EIP20.name 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: 385:4:

Gas costs:

Gas requirement of function StakesAlmond.get_gains2 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: 585:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 102:4:

Constant/View/Pure functions:

StakesAlmond.get_gains2(address,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 585:4:

Email: audit@EtherAuthority.io

Similar variable names:

StakesAlmond.(contract EIP20,contract EIP20,address,uint8,uint8,uint256,uint8,uint256,uint256,uint256) : Variables have very similar names "_owner" and "_lower". Note: Modifiers are currently not considered by this static analysis.
Pos: 502:23:

Similar variable names:

StakesAlmond.(contract EIP20,contract EIP20,address,uint8,uint8,uint256,uint8,uint256,uint256,uint256) : Variables have very similar names "_rate" and "_rate2". Note: Modifiers are currently not considered by this static analysis.
Pos: 498:24:

No return:

EIP20Interface.allowance(address,address): Defines a return type but never explicitly returns a value.
Pos: 363:4:

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.
more
Pos: 506:8:

ERC721Matcha.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ERC721Matcha.buy(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1376:4:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp". "block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1472:12:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1570:32:

Gas & Economy

Gas costs:

Gas requirement of function ERC721.balanceOf 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: 488:4:

Gas costs:

Gas requirement of function ERC721Matcha.withdraw 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: 1522:4:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 1581:60:

Miscellaneous

Constant/View/Pure functions:

ERC721._isApprovedOrOwner(address,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 658:4:

Similar variable names:

ERC721Matcha.auctionFinalize(uint256) : Variables have very similar names "success" and "success2". Note: Modifiers are currently not considered by this static analysis.
Pos: 1571:20:

No return:

ERC721Matcha.withdraw(uint256): Defines a return type but never explicitly returns a value.
Pos: 1522:4:

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.
more
Pos: 1464:8:

Solhint Linter

StakesAlmond.sol

```
StakesAlmond.sol:60:1: Error: Compiler version ^0.8.0 does not
satisfy the r semver requirement
StakesAlmond.sol:237:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
StakesAlmond.sol:293:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
StakesAlmond.sol:389:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
StakesAlmond.sol:411:9: Error: Variable name must be in mixedCase
StakesAlmond.sol:472:18: Error: Variable name must be in mixedCase
StakesAlmond.sol:473:18: Error: Variable name must be in mixedCase
StakesAlmond.sol:476:20: Error: Variable name must be in mixedCase
StakesAlmond.sol:490:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
StakesAlmond.sol:491:23: Error: Variable name must be in mixedCase
StakesAlmond.sol:508:40: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:518:12: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:523:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:524:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:524:40: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:525:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:553:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:554:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:555:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:555:40: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:556:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakesAlmond.sol:562:32: Error: Variable name must be in mixedCase
StakesAlmond.sol:576:5: Error: Function name must be in mixedCase
StakesAlmond.sol:576:42: Error: Variable name must be in mixedCase
StakesAlmond.sol:577:9: Error: Variable name must be in mixedCase
StakesAlmond.sol:577:35: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:578:9: Error: Variable name must be in mixedCase
StakesAlmond.sol:585:5: Error: Function name must be in mixedCase
StakesAlmond.sol:585:43: Error: Variable name must be in mixedCase
StakesAlmond.sol:586:9: Error: Variable name must be in mixedCase
StakesAlmond.sol:586:35: Error: Avoid to make time-based decisions in
your business logic
StakesAlmond.sol:587:9: Error: Variable name must be in mixedCase
```

```
StakesAlmond.sol:590:9: Error: Variable name must be in mixedCase
StakesAlmond.sol:600:5: Error: Function name must be in mixedCase
```

ERC721Matcha.sol

```
ERC721Matcha.sol:63:1: Error: Compiler version ^0.5.7 does not
satisfy the r semver requirement
ERC721Matcha.sol:1159:5: Error: Explicitly mark visibility of state
ERC721Matcha.sol:1258:5: Error: Explicitly mark visibility of state
ERC721Matcha.sol:1260:20: Error: Variable name must be in mixedCase
ERC721Matcha.sol:1263:5: Error: Explicitly mark visibility of state
ERC721Matcha.sol:1397:28: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1397:28: Error: Avoid using low level calls.
ERC721Matcha.sol:1401:29: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1401:29: Error: Avoid using low level calls.
ERC721Matcha.sol:1439:13: Error: Avoid to make time-based decisions
in your business logic
ERC721Matcha.sol:1472:13: Error: Avoid to make time-based decisions
in your business logic
ERC721Matcha.sol:1488:32: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1488:32: Error: Avoid using low level calls.
ERC721Matcha.sol:1508:21: Error: Avoid to make time-based decisions
in your business logic
ERC721Matcha.sol:1528:32: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1528:32: Error: Avoid using low level calls.
ERC721Matcha.sol:1539:13: Error: Avoid to make time-based decisions
in your business logic
ERC721Matcha.sol:1566:32: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1566:32: Error: Avoid using low level calls.
ERC721Matcha.sol:1570:33: Error: Avoid to use ".call.value()()"
ERC721Matcha.sol:1570:33: Error: Avoid using low level calls.
```

Software analysis result:

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

Email: audit@EtherAuthority.io