Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Shibaverse Protocol
Platform: Polygon
Language: Solidity
Date: March 4th, 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 Shibaverse team to perform the Security audit of the Shibaverse Protocol 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 March 4th, 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

Shibaverse protocol Contract is a smart contract, having functions like stake, withdraw, createAuction, makeBid, settleAuction, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Shibaverse Protocol Smart Contracts**
Platform	**Polygon / Solidity**
File 1	SimpleAuction.sol
File 1 MD5 Hash	164BDD94F317B8B00F5EDF79C1F07938
File 2	StakingRewards.sol
File 2 MD5 Hash	8D45DE4362F7A66120520DEB7C0FD531
Audit Date	March 4th,2022

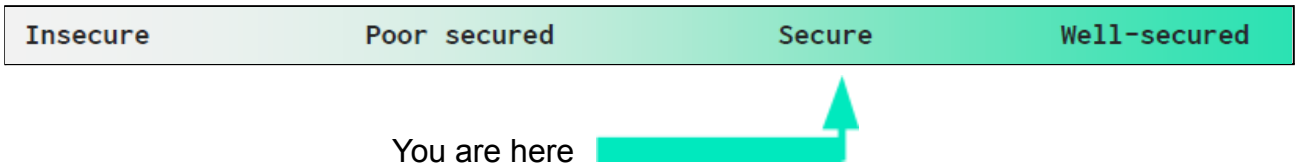
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 SimpleAuction.sol ● Bid Increase Percentage ● Minimum Auction Duration ● Maximum Auction Duration ● Auction Bid Period	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 2 StakingRewards.sol ● The StakingRewards can access functions like: rewardPerToken, stake, withdraw,etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. 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 1 low and some very low level issues. These issues are not critical ones.

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	Passed
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 Shibaverse 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 Shibaverse Protocol.

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

Code parts are **not** well commented on smart contracts.

Documentation

We were given a Shibaverse Protocol smart contract code in the form of a File. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not 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 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

SimpleAuction.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createAuction	external	Passed	No Issue
3	makeBid	external	Passed	No Issue
4	settleAuction	external	Passed	No Issue
5	setAuctionBidPeriod	external	Function input parameters lack of check	Refer Audit Findings
6	setBidIncreasePercentage	external	Function input parameters lack of check	Refer Audit Findings
7	setErc20Token	external	access only Owner	No Issue
8	setAuctionDuration	external	Function input parameters lack of check	Refer Audit Findings
9	addSeller	external	access only Owner	No Issue
10	removeSeller	external	access only Owner	No Issue
11	owner	read	Passed	No Issue
12	onlyOwner	modifier	Passed	No Issue
13	renounceOwnership	write	access only Owner	No Issue
14	transferOwnership	write	access only Owner	No Issue
15	_setOwner	write	Passed	No Issue

StakingRewards.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	transferOwnership	external	access only Owner	No Issue
4	pause	write	access only Owner	No Issue
5	unpause	write	access only Owner	No Issue
6	rewardPerToken	read	Passed	No Issue
7	earned	read	Passed	No Issue
8	updateReward	modifier	Passed	No Issue
9	changeRate	write	access only Owner	No Issue
10	stake	external	access by update Reward	No Issue
11	withdraw	external	access by update Reward	No Issue
12	getReward	external	access by update Reward	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 Medium severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

SimpleAuction.sol

In the below functions, there is no validation as stated in comments.

```
uint32 public bidIncreasePercentage; // 100 == 1% -> every bid must be higher than the previous
uint64 public auctionBidPeriod; // in seconds. The lenght of time between last bid and auction end. Auction duration increases if new bid is made in this period before auction end.
uint64 public minAuctionDuration; // in seconds 86400 = 1 day
uint64 public maxAuctionDuration; // in seconds 2678400 = 1 month
```

Functions are:

- setAuctionBidPeriod
- setAuctionDuration
- setBidIncreasePercentage

Resolution: We suggest adding some validation as per the requirement stated in comments.

Very Low / Informational / Best practices:

(1) Make variables immutable: **StakingRewards.sol**

Variable that is defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties

Resolution: We suggest making these variables immutable.

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:

- transferOwnership: The StakingRewards Owner can transfer new Ownership.
- pause: The StakingRewards Owner can set status false.
- unpause: The StakingRewards Owner can set status true.
- changeRate: The StakingRewards Owner can set a new rate.
- stake: The StakingRewards updateReward can add new stake.
- withdraw: The StakingRewards updateReward can withdraw the amount.
- getReward: The StakingRewards updateReward can get rewards.
- setAuctionBidPeriod: The SimpleAuction owner can set the length of time between last bid and auction end.
- setBidIncreasePercentage: The SimpleAuction owner can set a bid increase percentage.
- setErc20Token: The SimpleAuction owner can set ERC20 Tokens.
- setAuctionDuration: The SimpleAuction owner can set auction duration time.
- addSeller: The SimpleAuction owner can add a new seller address.
- removeSeller: The SimpleAuction owner can remove the seller address.

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 have observed some issues in the smart contracts, but those are not critical. So, **it's good to go to 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 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.

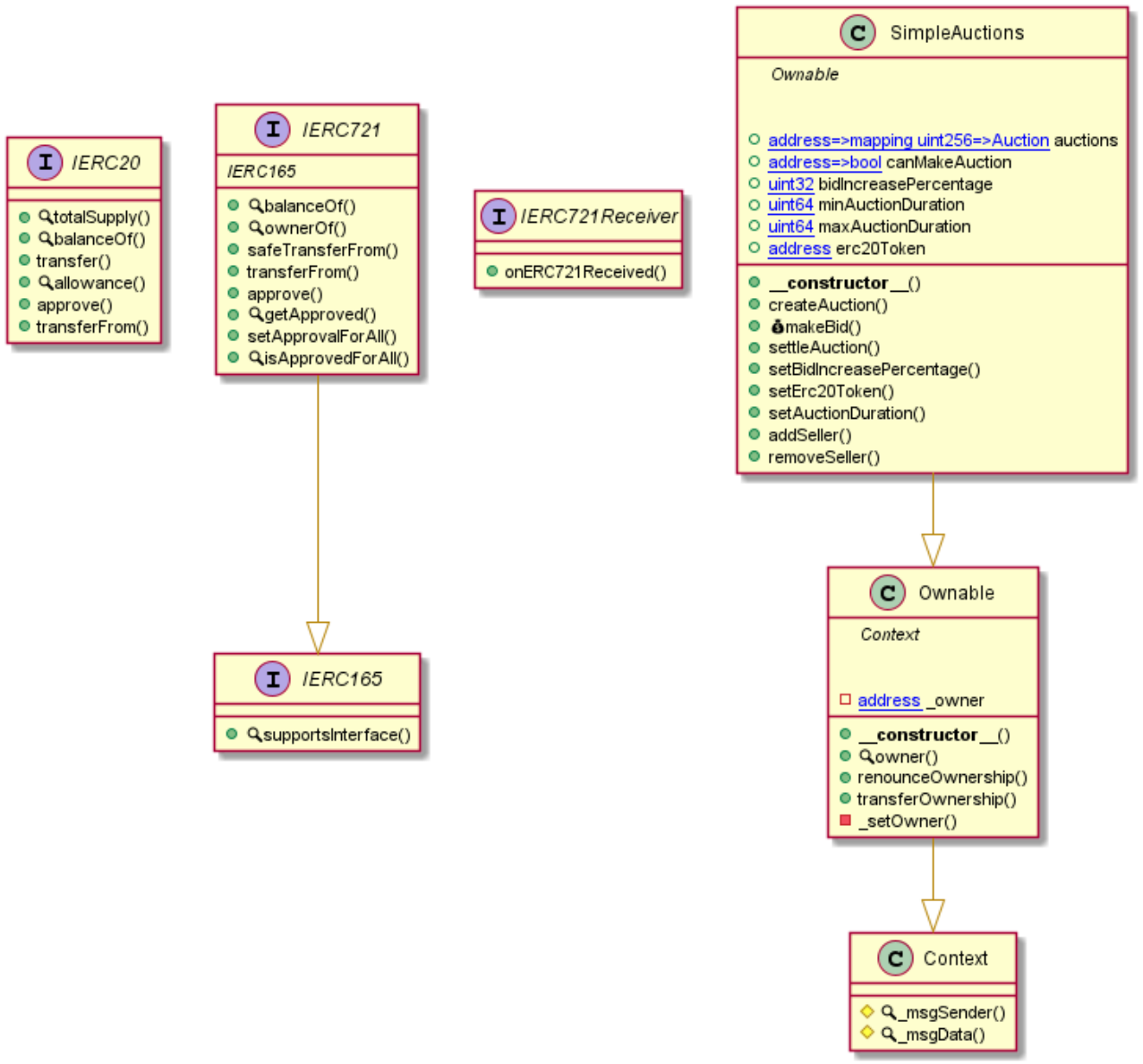
Technical Disclaimer

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.

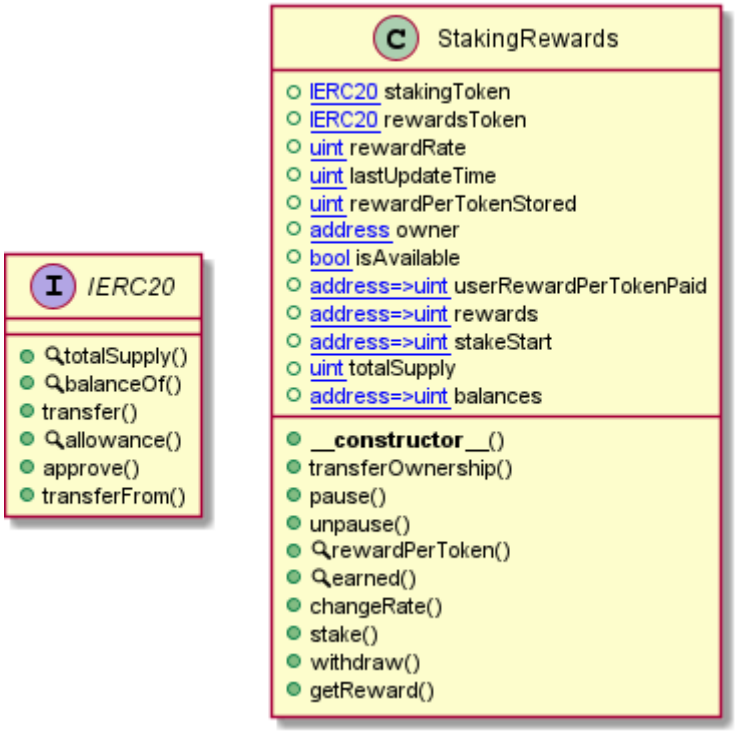
Appendix

Code Flow Diagram - Shibaverse Protocol

SimpleAuction Diagram



StakingRewards Diagram



Slither Results Log

Slither log >> SimpleAuction.sol

```
INFO:Detectors:
SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443) ignores return value by IERC20(erc20Token).transfer(auctions
nftContractAddress][_tokenId].nftHighestBidder,auctions[_nftContractAddress][_tokenId].nftHighestBid) (SimpleAuction.sol#421-424)
SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443) ignores return value by IERC20(erc20Token).transferFrom(msg.
nder,address(this),_tokenAmount) (SimpleAuction.sol#427-431)
SimpleAuctions.settleAuction(address,uint256) (SimpleAuction.sol#446-490) ignores return value by IERC20(erc20Token).transfer(_nftSeller
nftHighestBid) (SimpleAuction.sol#467)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Contract locking ether found:
        Contract SimpleAuctions (SimpleAuction.sol#307-513) has payable functions:
         - SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
Reentrancy in SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443):
        External calls:
        - IERC20(erc20Token).transfer(auctions[_nftContractAddress][_tokenId].nftHighestBidder,auctions[_nftContractAddress][_tokenId].n
HighestBid) (SimpleAuction.sol#421-424)
        - IERC20(erc20Token).transferFrom(msg.sender,address(this),_tokenAmount) (SimpleAuction.sol#427-431)
        State variables written after the call(s):
        - auctions[_nftContractAddress][_tokenId].nftHighestBid = _tokenAmount (SimpleAuction.sol#433)
        - auctions[_nftContractAddress][_tokenId].nftHighestBidder = msg.sender (SimpleAuction.sol#435)
Reentrancy in SimpleAuctions.settleAuction(address,uint256) (SimpleAuction.sol#446-490):
        External calls:
        - IERC721(_nftContractAddress).transferFrom(address(this),_nftSeller,_tokenId) (SimpleAuction.sol#459-463)
        - IERC20(erc20Token).transfer(_nftSeller,_nftHighestBid) (SimpleAuction.sol#467)
        - IERC721(_nftContractAddress).transferFrom(address(this),_nftHighestBidder,_tokenId) (SimpleAuction.sol#469-473)
        State variables written after the call(s):
        - auctions[_nftContractAddress][_tokenId].nftHighestBidder = address(0) (SimpleAuction.sol#477)
        - auctions[_nftContractAddress][_tokenId].nftHighestBid = 0 (SimpleAuction.sol#478)
        - auctions[_nftContractAddress][_tokenId].minPrice = 0 (SimpleAuction.sol#479)
        - auctions[_nftContractAddress][_tokenId].auctionEnd = 0 (SimpleAuction.sol#480)
        - auctions[_nftContractAddress][_tokenId].nftSeller = address(0) (SimpleAuction.sol#481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
SimpleAuctions.constructor(address,address,uint32,uint64,uint64)._erc20Token (SimpleAuction.sol#352) lacks a zero-check on :
                - erc20Token = _erc20Token (SimpleAuction.sol#359)
SimpleAuctions.setErc20Token(address)._erc20Token (SimpleAuction.sol#497) lacks a zero-check on :
                - erc20Token = _erc20Token (SimpleAuction.sol#498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in SimpleAuctions.createAuction(address,uint256,uint128,uint64) (SimpleAuction.sol#366-402):
        External calls:
        - IERC721(_nftContractAddress).transferFrom(msg.sender,address(this),_tokenId) (SimpleAuction.sol#379-383)
        State variables written after the call(s):
        - auctions[_nftContractAddress][_tokenId].minPrice = _minPrice (SimpleAuction.sol#390)
        - auctions[_nftContractAddress][_tokenId].nftSeller = msg.sender (SimpleAuction.sol#391-392)
        - auctions[_nftContractAddress][_tokenId].auctionEnd = _auctionEnd (SimpleAuction.sol#393)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in SimpleAuctions.createAuction(address,uint256,uint128,uint64) (SimpleAuction.sol#366-402):
        External calls:
        - IERC721(_nftContractAddress).transferFrom(msg.sender,address(this),_tokenId) (SimpleAuction.sol#379-383)
        Event emitted after the call(s):
        - AuctionCreated(_nftContractAddress,_tokenId,msg.sender,_minPrice,_auctionEnd) (SimpleAuction.sol#395-401)
Reentrancy in SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443):
        External calls:
        - IERC20(erc20Token).transfer(auctions[_nftContractAddress][_tokenId].nftHighestBidder,auctions[_nftContractAddress][_tokenId].n
HighestBid) (SimpleAuction.sol#421-424)
        - IERC20(erc20Token).transferFrom(msg.sender,address(this),_tokenAmount) (SimpleAuction.sol#427-431)
        Event emitted after the call(s):
        - BidMade(_nftContractAddress,_tokenId,msg.sender,_tokenAmount) (SimpleAuction.sol#437-442)
Reentrancy in SimpleAuctions.settleAuction(address,uint256) (SimpleAuction.sol#446-490):
        External calls:
        - IERC721(_nftContractAddress).transferFrom(address(this),_nftSeller,_tokenId) (SimpleAuction.sol#459-463)
        - IERC20(erc20Token).transfer(_nftSeller,_nftHighestBid) (SimpleAuction.sol#467)
        - IERC721(_nftContractAddress).transferFrom(address(this),_nftHighestBidder,_tokenId) (SimpleAuction.sol#469-473)
        Event emitted after the call(s):
        - NFTTransferredAndSellerPaid(_nftContractAddress,_tokenId,_nftSeller,_nftHighestBid,_nftHighestBidder) (SimpleAuction.sol#483-4
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
SimpleAuctions.createAuction(address,uint256,uint128,uint64) (SimpleAuction.sol#366-402) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp + (minAuctionDuration * 60) <= _auctionEnd && block.timestamp + (maxAuctionDuration * 60)
= _auctionEnd,Invalid auctionEnd) (SimpleAuction.sol#377)
SimpleAuctions.makeBid(address,uint256,uint128) (SimpleAuction.sol#405-443) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < auctions[_nftContractAddress][_tokenId].auctionEnd,Auction has ended) (SimpleAuction.so
413)
SimpleAuctions.settleAuction(address,uint256) (SimpleAuction.sol#446-490) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= auctions[_nftContractAddress][_tokenId].auctionEnd,Auction ongoing) (SimpleAuction.sol
49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (SimpleAuction.sol#248-250) 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 (SimpleAuction.sol#2) 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 SimpleAuctions.createAuction(address,uint256,uint128,uint64)._nftContractAddress (SimpleAuction.sol#367) is not in mixedCase
Parameter SimpleAuctions.createAuction(address,uint256,uint128,uint64)._tokenId (SimpleAuction.sol#368) is not in mixedCase
Parameter SimpleAuctions.createAuction(address,uint256,uint128,uint64)._minPrice (SimpleAuction.sol#369) is not in mixedCase
Parameter SimpleAuctions.createAuction(address,uint256,uint128,uint64)._auctionEnd (SimpleAuction.sol#370) is not in mixedCase
Parameter SimpleAuctions.makeBid(address,uint256,uint128)._nftContractAddress (SimpleAuction.sol#406) is not in mixedCase
Parameter SimpleAuctions.makeBid(address,uint256,uint128)._tokenId (SimpleAuction.sol#407) is not in mixedCase
Parameter SimpleAuctions.makeBid(address,uint256,uint128)._tokenAmount (SimpleAuction.sol#408) is not in mixedCase
Parameter SimpleAuctions.settleAuction(address,uint256)._nftContractAddress (SimpleAuction.sol#446) is not in mixedCase
Parameter SimpleAuctions.settleAuction(address,uint256)._tokenId (SimpleAuction.sol#446) is not in mixedCase
Parameter SimpleAuctions.setBidIncreasePercentage(uint32)._bidIncreasePercentage (SimpleAuction.sol#493) is not in mixedCase
Parameter SimpleAuctions.setErc20Token(address)._erc20Token (SimpleAuction.sol#497) is not in mixedCase
Parameter SimpleAuctions.setAuctionDuration(uint64,uint64)._minAuctionDuration (SimpleAuction.sol#501) is not in mixedCase
Parameter SimpleAuctions.setAuctionDuration(uint64,uint64)._maxAuctionDuration (SimpleAuction.sol#501) is not in mixedCase
Parameter SimpleAuctions.addSeller(address)._seller (SimpleAuction.sol#506) is not in mixedCase
Parameter SimpleAuctions.removeSeller(address)._seller (SimpleAuction.sol#510) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SimpleAuction.sol#286-288)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SimpleAuction.sol#294-297)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SimpleAuction.sol analyzed (7 contracts with 75 detectors), 35 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> StakingRewards.sol

```
INFO:Detectors:
StakingRewards.stake(uint256) (StakingRewards.sol#161-169) ignores return value by stakingToken.transferFrom(msg.sender,address(this),_a
mount) (StakingRewards.sol#166)
StakingRewards.withdraw(uint256) (StakingRewards.sol#171-193) ignores return value by stakingToken.transfer(msg.sender,_amountToWithdraw
) (StakingRewards.sol#180)
StakingRewards.withdraw(uint256) (StakingRewards.sol#171-193) ignores return value by stakingToken.transfer(msg.sender,_amount) (Staking
Rewards.sol#187)
StakingRewards.getReward() (StakingRewards.sol#195-213) ignores return value by rewardsToken.transfer(msg.sender,reward) (StakingRewards
.sol#201)
StakingRewards.getReward() (StakingRewards.sol#195-213) ignores return value by rewardsToken.transfer(msg.sender,reward_scope_0) (Stakin
gRewards.sol#208)
StakingRewards.withdrawStakingTokenLeftovers() (StakingRewards.sol#215-220) ignores return value by stakingToken.transfer(owner,leftover
) (StakingRewards.sol#219)
StakingRewards.withdrawRewardsToken(uint256) (StakingRewards.sol#222-226) ignores return value by rewardsToken.transfer(owner,_amount) (
StakingRewards.sol#225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
StakingRewards.transferOwnership(address) (StakingRewards.sol#119-121) should emit an event for:
        - owner = _newOwner (StakingRewards.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StakingRewards.changeRate(uint256) (StakingRewards.sol#157-159) should emit an event for:
        - rewardRate = _newRate (StakingRewards.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
StakingRewards.transferOwnership(address)._newOwner (StakingRewards.sol#119) lacks a zero-check on :
                - owner = _newOwner (StakingRewards.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StakingRewards.getReward() (StakingRewards.sol#195-213):
        External calls:
        - rewardsToken.transfer(msg.sender,reward) (StakingRewards.sol#201)
        Event emitted after the call(s):
        - WitdrawRewards(msg.sender,reward,true) (StakingRewards.sol#203)
Reentrancy in StakingRewards.getReward() (StakingRewards.sol#195-213):
        External calls:
        - rewardsToken.transfer(msg.sender,reward_scope_0) (StakingRewards.sol#208)
```

```
        Event emitted after the call(s):
        - WitdrawRewards(msg.sender,reward_scope_0,false) (StakingRewards.sol#210)
Reentrancy in StakingRewards.stake(uint256) (StakingRewards.sol#161-169):
        External calls:
        - stakingToken.transferFrom(msg.sender,address(this),_amount) (StakingRewards.sol#166)
        Event emitted after the call(s):
        - StartStaked(msg.sender,_amount) (StakingRewards.sol#168)
Reentrancy in StakingRewards.withdraw(uint256) (StakingRewards.sol#171-193):
        External calls:
        - stakingToken.transfer(msg.sender,_amountToWithdraw) (StakingRewards.sol#180)
        Event emitted after the call(s):
        - WitdrawStaked(msg.sender,_amountToWithdraw,true) (StakingRewards.sol#182)
Reentrancy in StakingRewards.withdraw(uint256) (StakingRewards.sol#171-193):
        External calls:
        - stakingToken.transfer(msg.sender,_amount) (StakingRewards.sol#187)
        Event emitted after the call(s):
        - WitdrawStaked(msg.sender,_amount,false) (StakingRewards.sol#189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewards.withdraw(uint256) (StakingRewards.sol#171-193) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((block.timestamp - stakeStart[msg.sender]) >= initialTime,Not time yet) (StakingRewards.sol#172)
        - (block.timestamp - stakeStart[msg.sender]) < lockedTime (StakingRewards.sol#176)
StakingRewards.getReward() (StakingRewards.sol#195-213) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((block.timestamp - stakeStart[msg.sender]) >= initialTime,Not time yet) (StakingRewards.sol#196)
        - (block.timestamp - stakeStart[msg.sender]) < lockedTime (StakingRewards.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
StakingRewards.stake(uint256) (StakingRewards.sol#161-169) compares to a boolean constant:
        -require(bool,string)(isAvailable == true,The Staking is Paused) (StakingRewards.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version^0.8 (StakingRewards.sol#2) is too complex
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 StakingRewards.transferOwnership(address)._newOwner (StakingRewards.sol#119) is not in mixedCase
Parameter StakingRewards.changeRate(uint256)._newRate (StakingRewards.sol#157) is not in mixedCase
Parameter StakingRewards.stake(uint256)._amount (StakingRewards.sol#161) is not in mixedCase
Parameter StakingRewards.withdraw(uint256)._amount (StakingRewards.sol#171) is not in mixedCase
Parameter StakingRewards.withdrawRewardsToken(uint256)._amount (StakingRewards.sol#222) is not in mixedCase
Parameter StakingRewards.changeLockedTime(uint256)._lockedTime (StakingRewards.sol#228) is not in mixedCase
Parameter StakingRewards.changeInitialTime(uint256)._initialTime (StakingRewards.sol#232) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
pause() should be declared external:
        - StakingRewards.pause() (StakingRewards.sol#122-124)
unpause() should be declared external:
        - StakingRewards.unpause() (StakingRewards.sol#125-127)
changeRate(uint256) should be declared external:
        - StakingRewards.changeRate(uint256) (StakingRewards.sol#157-159)
changeLockedTime(uint256) should be declared external:
        - StakingRewards.changeLockedTime(uint256) (StakingRewards.sol#228-230)
changeInitialTime(uint256) should be declared external:
        - StakingRewards.changeInitialTime(uint256) (StakingRewards.sol#232-234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StakingRewards.sol analyzed (2 contracts with 75 detectors), 32 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

SimpleAuction.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SimpleAuctions.createAuction(address,uint256,uint128,uint64): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 366:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SimpleAuctions.makeBid(address,uint256,uint128): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 405:4:

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: 377:16:

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: 377:78:

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: 449:16:

Gas & Economy

Gas costs:

Gas requirement of function SimpleAuctions.createAuction 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: 366:4:

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function SimpleAuctions.settleAuction 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: 446:4:

Gas costs:

Gas requirement of function SimpleAuctions.setAuctionDuration 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: 501:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 23:4:

Similar variable names:

SimpleAuctions.(address,address,uint32,uint64,uint64) : Variables have very similar names "minAuctionDuration" and "maxAuctionDuration". Note: Modifiers are currently not considered by this static analysis.
Pos: 360:8:

Similar variable names:

SimpleAuctions.(address,address,uint32,uint64,uint64) : Variables have very similar names "minAuctionDuration" and "maxAuctionDuration". Note: Modifiers are currently not considered by this static analysis.
Pos: 361:8:

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: 413:8:

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: 414:8:

StakingRewards.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in StakingRewards.stake(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 161:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in StakingRewards.withdraw(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 171:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in StakingRewards.getReward(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 195:4:

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: 172:18:

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: 176:12:

Gas & Economy

Gas costs:

Gas requirement of function StakingRewards.rewardPerToken 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: 129:4:

Gas costs:

Gas requirement of function StakingRewards.earned 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: 138:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 21:4:

Similar variable names:

StakingRewards.getReward() : Variables have very similar names "reward" and "rewards". Note: Modifiers are currently not considered by this static analysis.

Pos: 200:12:

Similar variable names:

StakingRewards.getReward() : Variables have very similar names "reward" and "rewards". Note: Modifiers are currently not considered by this static analysis.

Pos: 201:46:

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: 196:8:

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: 218:8:

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: 218:8:

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: 224:8:

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: 135:13:

Solhint Linter

SimpleAuction.sol

```
SimpleAuction.sol:2:1: Error: Compiler version ^0.8.0 does not
satisfy the r semver requirement
SimpleAuction.sol:260:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
SimpleAuction.sol:351:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
SimpleAuction.sol:377:17: Error: Avoid to make time-based decisions
in your business logic
SimpleAuction.sol:377:79: Error: Avoid to make time-based decisions
in your business logic
SimpleAuction.sol:413:17: Error: Avoid to make time-based decisions
in your business logic
SimpleAuction.sol:433:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:435:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:449:17: Error: Avoid to make time-based decisions
in your business logic
SimpleAuction.sol:477:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:478:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:479:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:480:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
SimpleAuction.sol:481:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
```

StakingRewards.sol

```
StakingRewards.sol:2:1: Error: Compiler version ^0.8 does not satisfy
the r semver requirement
StakingRewards.sol:105:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
StakingRewards.sol:135:16: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:150:26: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:165:34: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:172:19: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:176:13: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:185:13: Error: Possible reentrancy
```

```
vulnerabilities. Avoid state changes after transfer.
StakingRewards.sol:186:13: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
StakingRewards.sol:196:19: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:198:13: Error: Avoid to make time-based decisions
in your business logic
StakingRewards.sol:207:13: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
```

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