Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	AlphasElephantCoin
Website:	https://alphaselephant.net
Platform:	Ethereum
Language:	Solidity
Date:	October 10th, 2023

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 AlphasElephantCoin team to perform the Security audit of the ALPHAS token smart contract 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 October 10th, 2023.

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 AlphasElephantCoin Token is a standard smart contract that allows users to update marketing and charity wallet addresses, set buy and sell limits, and more.

Audit scope

Name	**Code Review and Security Analysis Report for AlphasElephantCoin (ALPHAS) Token Smart Contract**
Platform	**Ethereum**
File	Alphas.sol
Github commit hash	5ff3d843f29079ee35e21e9fd753283d114c3576
Audit Date	October 10th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: AlphasElephantCoin • Symbol: ALPHAS • Decimals: 18 • Total supply: 1 Trillion	**YES, This is valid.**
Ownership Control: • Set the market maker pair address. • Set the charity wallet address. • Set the lottery wallet address. • Set the marketing wallet address • Set the developer wallet address. • Set the burn tax percentage. • Set the percentage values. • Withdraw ERC20 tokens that are potentially stuck in the contract. • The current owner can transfer ownership. • Owners can renounce ownership.	**YES, This is valid. We advise to renounce ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. **Also, these contracts 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 few very low level issues.

These issues are acknowledged by the project team

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	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	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**

Business Risk Analysis

Category	Result
Buy Tax	7%
Sell Tax	9%
Cannot Buy	Passed
Cannot Sell	Passed
Modify Tax	Passed
Fee Check	Passed
Is Honeypot	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	No
Pause Transfer?	Not Detected
Max Tax?	Passed
Is it Anti-whale?	Not Detected
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	Passed
Can Mint?	No
Is it Proxy?	Not Detected
Can Take Ownership?	Not Detected
Hidden Owner?	Not Detected
Self Destruction?	Not Detected
Auditor Confidence	High

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

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

The libraries in ALPHAS 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 ALPHAS 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 not well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given an ALPHAS Token smart contract code in the form of a github.com web link.

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

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that 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

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	name	read	Passed	No Issue
9	symbol	read	Passed	No Issue
10	decimals	read	Passed	No Issue
11	totalSupply	read	Passed	No Issue
12	balanceOf	read	Passed	No Issue
13	transfer	write	Passed	No Issue
14	transferFrom	write	Passed	No Issue
15	allowance	read	Passed	No Issue
16	approve	write	Passed	No Issue
17	_approve	internal	Passed	No Issue
18	_spendAllowance	internal	Passed	No Issue
19	_burnTokens	internal	Passed	No Issue
20	_transferTokens	internal	Passed	No Issue
21	TransferEx	write	access only Owner	No Issue
22	setAutomatedMarketMakerPair	write	access only Owner	No Issue
23	_setAutomatedMarketMakerPair	write	Passed	No Issue
24	setExcludedFromFee	external	access only Owner	No Issue
25	setMarketingWallet	external	access only Owner	No Issue
26	setCharityWallet	external	access only Owner	No Issue
27	setLotteryWallet	external	access only Owner	No Issue
28	setDevWallet	external	access only Owner	No Issue
29	updateShares	internal	Passed	No Issue
30	setBurnTaxPercentage	external	access only Owner	No Issue
31	setMarketingPercentage	external	access only Owner	No Issue
32	setDevPercentage	external	access only Owner	No Issue
33	setCharityPercentage	external	access only Owner	No Issue
34	setLotteryPercentage	external	access only Owner	No Issue
35	setTaxThreshold	external	access only Owner	No Issue
36	setMaxAmount	external	access only Owner	No Issue
37	updateFees	internal	Passed	No Issue
38	updateMaxLimit	internal	Passed	No Issue
39	recoverTokensFromContract	external	access only Owner	No Issue
40	recoverETHfromContract	external	access only Owner	No Issue
41	swapTokensForEth	write	Passed	No Issue
42	swapTokens	internal	Passed	No Issue

43	addLiquidity	write	Passed	No Issue
44	_transfer	internal	Passed	No Issue
45	_calculateTax	internal	Passed	No Issue
46	fallback	external	Passed	No Issue
47	receive	external	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

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Consider renouncing ownership:

Once all the administrative functions are over, then we advise to renounce the ownership of the contract. This will make it fully decentralized. Fully decentralized contracts increase the trust in the users.

Status: Acknowledged

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:

Alphas.sol

- TransferEx: The owner can transfer the amount.
- setAutomatedMarketMakerPair: The market maker pair address can be set by the owner.
- setExcludedFromFee: An excluded address can be set by the owner.
- setMarketingWallet: The marketing wallet address can be set by the owner.
- setCharityWallet: The charity wallet address can be set by the owner.
- setLotteryWallet: The lottery wallet address can be set by the owner.
- setDevWallet: The developer wallet address can be set by the owner.
- setBurnTaxPercentage: The burn tax percentage can be set by the owner.
- setMarketingPercentage: The marketing percentage can be set by the owner.
- setDevPercentage: The developer percentage can be set by the owner.
- setCharityPercentage: The charge percentage can be set by the owner.
- setLotteryPercentage: The lottery percentage can be set by the owner.
- setTaxThreshold: The tax threshold value can be set by the owner.
- recoverTokensFromContract: Withdraw ERC20 tokens that are potentially stuck in contract by the owner.
- recoverETHfromContract: Withdraw ether tokens that are potentially stuck in a contract by the owner.

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: The current owner can transfer ownership of the contract to a new account.

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

Conclusion

We were given a contract code in the form of github.com web links. And we have used all possible tests based on given objects as files. We had not observed any issues in the smart contracts. 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 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 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 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).

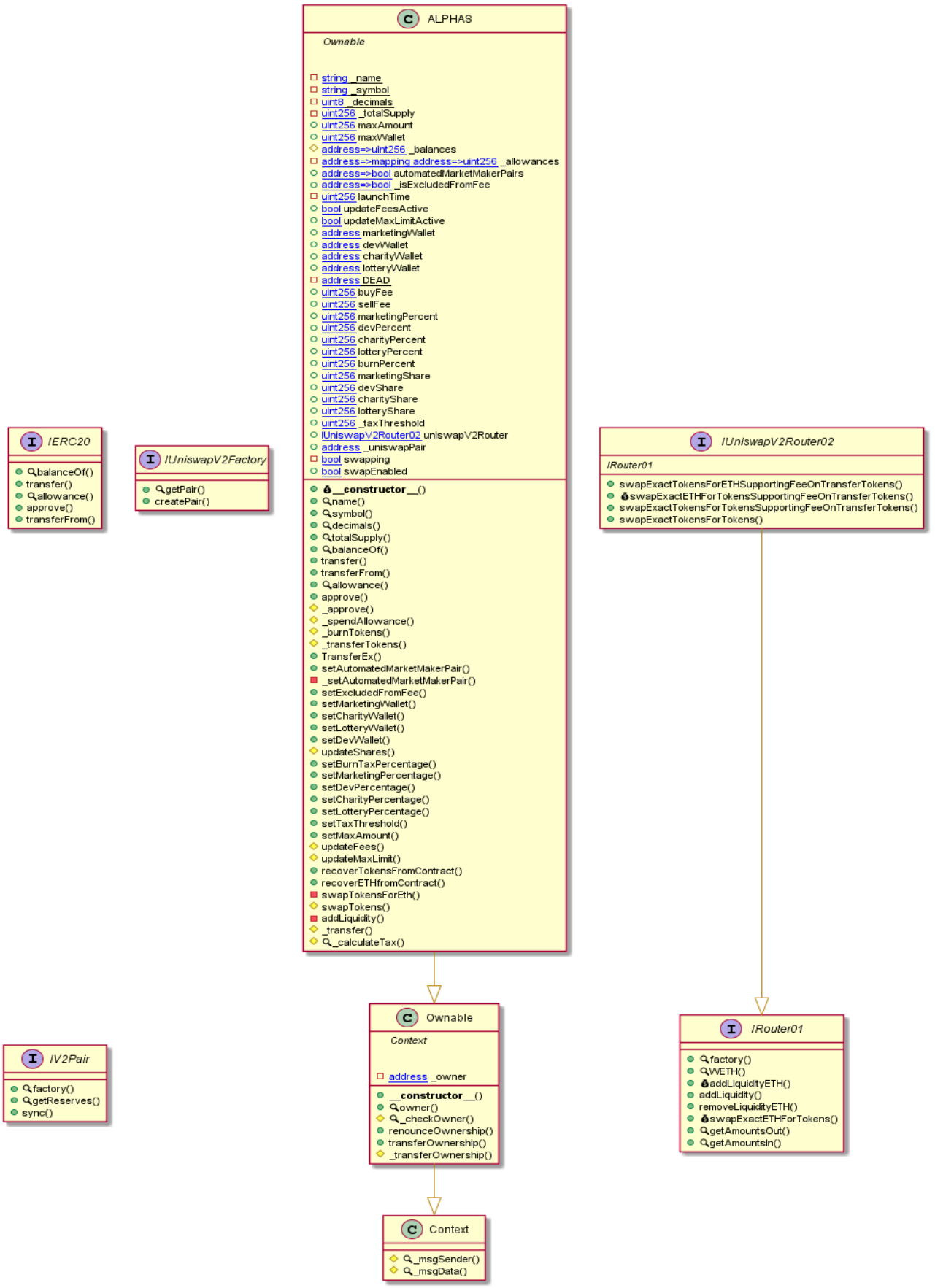
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - AlphasElephantCoin Token



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.

Slither Log >> Alphas.sol

```
ALPHAS.transfer(address,uint256).owner (Alphas.sol#390) shadows:
        - Ownable.owner() (Alphas.sol#62-64) (function)
ALPHAS.allowance(address,address).owner (Alphas.sol#407) shadows:
        - Ownable.owner() (Alphas.sol#62-64) (function)
ALPHAS._spendAllowance(address,address,uint256).owner (Alphas.sol#428) shadows:
        - Ownable.owner() (Alphas.sol#62-64) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

ALPHAS.setBurnTaxPercentage(uint256) (Alphas.sol#546-549) should emit an event for:
        - burnPercent = taxPercentage (Alphas.sol#548)
ALPHAS.setMarketingPercentage(uint256) (Alphas.sol#551-555) should emit an event for:
        - marketingPercent = taxPercentage (Alphas.sol#553)
ALPHAS.setDevPercentage(uint256) (Alphas.sol#557-561) should emit an event for:
        - devPercent = taxPercentage (Alphas.sol#559)
ALPHAS.setCharityPercentage(uint256) (Alphas.sol#563-567) should emit an event for:
        - charityPercent = taxPercentage (Alphas.sol#565)
ALPHAS.setLotteryPercentage(uint256) (Alphas.sol#569-573) should emit an event for:
        - lotteryPercent = taxPercentage (Alphas.sol#571)
ALPHAS.setTaxThreshold(uint256) (Alphas.sol#575-577) should emit an event for:
        - _taxThreshold = threshold (Alphas.sol#576)
ALPHAS.setMaxAmount(uint256) (Alphas.sol#580-582) should emit an event for:
        - maxAmount = amount (Alphas.sol#581)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

ALPHAS.constructor(address,address,address,address)._marketingWallet (Alphas.sol#331) lacks a zero-check on :
                - marketingWallet = _marketingWallet (Alphas.sol#351)
ALPHAS.constructor(address,address,address,address)._devWallet (Alphas.sol#331) lacks a zero-check on :
                - devWallet = _devWallet (Alphas.sol#352)
ALPHAS.constructor(address,address,address,address)._charityWallet (Alphas.sol#331) lacks a zero-check on :
                - charityWallet = _charityWallet (Alphas.sol#353)
ALPHAS.constructor(address,address,address,address)._lotteryWallet (Alphas.sol#331) lacks a zero-check on :
                - lotteryWallet = _lotteryWallet (Alphas.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in ALPHAS._transfer(address,address,uint256) (Alphas.sol#737-816):
        External calls:
        - swapTokens() (Alphas.sol#775)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.tim
estamp) (Alphas.sol#686-692)
                - (success,None) = marketingWallet.call{gas: 35000,value: marketingAmount}() (Alphas.sol#715)
                - (success1,None) = devWallet.call{gas: 35000,value: devAmount}() (Alphas.sol#716)
                - (success2,None) = charityWallet.call{gas: 35000,value: charityAmount}() (Alphas.sol#717)
                - (success3,None) = lotteryWallet.call{gas: 35000,value: lotteryAmount}() (Alphas.sol#718)
        External calls sending eth:
        - swapTokens() (Alphas.sol#775)
                - (success,None) = marketingWallet.call{gas: 35000,value: marketingAmount}() (Alphas.sol#715)
                - (success1,None) = devWallet.call{gas: 35000,value: devAmount}() (Alphas.sol#716)
                - (success2,None) = charityWallet.call{gas: 35000,value: charityAmount}() (Alphas.sol#717)
                - (success3,None) = lotteryWallet.call{gas: 35000,value: lotteryAmount}() (Alphas.sol#718)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (Alphas.sol#476)
                - _transferTokens(sender,address(this),sellTax - burnTax) (Alphas.sol#807)
        - Transfer(from,to,amount) (Alphas.sol#476)
                - _transferTokens(sender,address(this),buyTax - burnTax) (Alphas.sol#795)
        - Transfer(from,to,amount) (Alphas.sol#476)
                - _transferTokens(sender,recipient,amount) (Alphas.sol#814)
        - Transfer(account,address(0),amount) (Alphas.sol#455)
                - _burnTokens(sender,DEAD,burnTax) (Alphas.sol#796)
        - Transfer(account,address(0),amount) (Alphas.sol#455)
                - _burnTokens(sender,DEAD,burnTax) (Alphas.sol#808)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
ALPHAS.updateFees() (Alphas.sol#584-621) uses timestamp for comparisons
        Dangerous comparisons:
        - updateFeesActive && block.timestamp <= launchTime + 86400 (Alphas.sol#586)
        - block.timestamp <= launchTime + 300 (Alphas.sol#588)
        - block.timestamp > launchTime + 300 && block.timestamp <= launchTime + 900 (Alphas.sol#593)
        - block.timestamp > launchTime + 900 && block.timestamp <= launchTime + 1800 (Alphas.sol#598)
        - block.timestamp > launchTime + 1800 && block.timestamp <= launchTime + 7200 (Alphas.sol#602)
        - block.timestamp > launchTime + 7200 && block.timestamp <= launchTime + 86400 (Alphas.sol#606)
        - block.timestamp > launchTime + 86400 (Alphas.sol#616)
```

```
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses timestamp for comparisons
        Dangerous comparisons:
        - updateMaxLimitActive && block.timestamp <= launchTime + 1800 (Alphas.sol#625)
        - block.timestamp <= launchTime + 600 (Alphas.sol#627)
        - block.timestamp > launchTime + 600 && block.timestamp <= launchTime + 1200 (Alphas.sol#632)
        - block.timestamp > launchTime + 1200 && block.timestamp <= launchTime + 1800 (Alphas.sol#637)
        - block.timestamp > launchTime + 1800 (Alphas.sol#647)
ALPHAS._transfer(address,address,uint256) (Alphas.sol#737-816) uses timestamp for comparisons
        Dangerous comparisons:
        - launchTime == 0 && recipient == _uniswapPair (Alphas.sol#744)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ALPHAS.addLiquidity(uint256,uint256) (Alphas.sol#722-735) is never used and should be removed
Context._msgData() (Alphas.sol#19-21) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (Alphas.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.1
6
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in ALPHAS.swapTokens() (Alphas.sol#695-720):
        - (success,None) = marketingWallet.call{gas: 35000,value: marketingAmount}() (Alphas.sol#715)
        - (success1,None) = devWallet.call{gas: 35000,value: devAmount}() (Alphas.sol#716)
        - (success2,None) = charityWallet.call{gas: 35000,value: charityAmount}() (Alphas.sol#717)
        - (success3,None) = lotteryWallet.call{gas: 35000,value: lotteryAmount}() (Alphas.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

ALPHAS (Alphas.sol#266-826) should inherit from IERC20 (Alphas.sol#107-182)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
```

```
Function IRouter01.WETH() (Alphas.sol#200) is not in mixedCase
Function ALPHAS.TransferEx(address[],uint256) (Alphas.sol#479-494) is not in mixedCase
Parameter ALPHAS.TransferEx(address[],uint256)._input (Alphas.sol#480) is not in mixedCase
Parameter ALPHAS.TransferEx(address[],uint256)._amount (Alphas.sol#481) is not in mixedCase
Parameter ALPHAS.recoverTokensFromContract(address,uint256)._tokenAddress (Alphas.sol#656) is not in mixedCase
Constant ALPHAS._name (Alphas.sol#268) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ALPHAS._symbol (Alphas.sol#269) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ALPHAS._decimals (Alphas.sol#270) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ALPHAS._balances (Alphas.sol#276) is not in mixedCase
Variable ALPHAS._isExcludedFromFee (Alphas.sol#280) is not in mixedCase
Variable ALPHAS._taxThreshold (Alphas.sol#306) is not in mixedCase
Variable ALPHAS._uniswapPair (Alphas.sol#309) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Alphas.sol#212
) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Alp
has.sol#213)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
ALPHAS.updateShares() (Alphas.sol#535-544) uses literals with too many digits:
        - marketingShare = (marketingPercent * 100000) / totalTaxPercent (Alphas.sol#539)
ALPHAS.updateShares() (Alphas.sol#535-544) uses literals with too many digits:
        - devShare = (devPercent * 100000) / totalTaxPercent (Alphas.sol#540)
ALPHAS.updateShares() (Alphas.sol#535-544) uses literals with too many digits:
        - charityShare = (charityPercent * 100000) / totalTaxPercent (Alphas.sol#541)
ALPHAS.updateShares() (Alphas.sol#535-544) uses literals with too many digits:
        - lotteryShare = (lotteryPercent * 100000) / totalTaxPercent (Alphas.sol#542)
ALPHAS.setBurnTaxPercentage(uint256) (Alphas.sol#546-549) uses literals with too many digits:
        - require(bool,string)(marketingPercent + devPercent + charityPercent + lotteryPercent + burnPercent <= 100000,Tax perc
entage cannot exceed 100%) (Alphas.sol#547)
ALPHAS.setMarketingPercentage(uint256) (Alphas.sol#551-555) uses literals with too many digits:
        - require(bool,string)(marketingPercent + devPercent + charityPercent + lotteryPercent + burnPercent <= 100000,Tax perc
entage cannot exceed 100%) (Alphas.sol#552)
ALPHAS.setDevPercentage(uint256) (Alphas.sol#557-561) uses literals with too many digits:
        - require(bool,string)(marketingPercent + devPercent + charityPercent + lotteryPercent + burnPercent <= 100000,Tax perc
entage cannot exceed 100%) (Alphas.sol#558)
ALPHAS.setCharityPercentage(uint256) (Alphas.sol#563-567) uses literals with too many digits:
        - require(bool,string)(marketingPercent + devPercent + charityPercent + lotteryPercent + burnPercent <= 100000,Tax perc
entage cannot exceed 100%) (Alphas.sol#564)
ALPHAS.setLotteryPercentage(uint256) (Alphas.sol#569-573) uses literals with too many digits:
        - require(bool,string)(marketingPercent + devPercent + charityPercent + lotteryPercent + burnPercent <= 100000,Tax perc
entage cannot exceed 100%) (Alphas.sol#570)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxAmount = (totalSupply() * 280) / 100000 (Alphas.sol#628)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxWallet = (totalSupply() * 280) / 100000 (Alphas.sol#629)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxAmount = (totalSupply() * 550) / 100000 (Alphas.sol#633)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxWallet = (totalSupply() * 550) / 100000 (Alphas.sol#634)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxAmount = (totalSupply() * 1110) / 100000 (Alphas.sol#638)
ALPHAS.updateMaxLimit() (Alphas.sol#623-652) uses literals with too many digits:
        - maxWallet = (totalSupply() * 1110) / 100000 (Alphas.sol#639)
ALPHAS.recoverTokensFromContract(address,uint256) (Alphas.sol#655-670) uses literals with too many digits:
```

```
        - _tokenAmount = _tokenBalance * percent / 100000 (Alphas.sol#666)
ALPHAS.swapTokens() (Alphas.sol#695-720) uses literals with too many digits:
        - marketingAmount = (newBalance * marketingShare) / 100000 (Alphas.sol#705)
ALPHAS.swapTokens() (Alphas.sol#695-720) uses literals with too many digits:
        - devAmount = (newBalance * devShare) / 100000 (Alphas.sol#706)
ALPHAS.swapTokens() (Alphas.sol#695-720) uses literals with too many digits:
        - charityAmount = (newBalance * charityShare) / 100000 (Alphas.sol#707)
ALPHAS.swapTokens() (Alphas.sol#695-720) uses literals with too many digits:
        - lotteryAmount = (newBalance * lotteryShare) / 100000 (Alphas.sol#708)
ALPHAS._calculateTax(uint256,uint256) (Alphas.sol#818-820) uses literals with too many digits:
        - amount * (taxPercentage) / (100000) (Alphas.sol#819)
ALPHAS.slitherConstructorVariables() (Alphas.sol#266-826) uses literals with too many digits:
        - _totalSupply = 1000000000000 * 10 ** uint256(_decimals) (Alphas.sol#271)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ALPHAS.swapEnabled (Alphas.sol#312) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
Alphas.sol analyzed (8 contracts with 84 detectors), 65 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.

Alphas.sol

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: 586:32:

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: 586:32:

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: 718:22:

Gas costs:

Gas requirement of function ALPHAS.TransferEx 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: 479:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 485:12:

Similar variable names:

ALPHAS._transfer(address,address,uint256) : Variables have very similar names "burnTax" and "buyTax". Note: Modifiers are currently not considered by this static analysis.

Pos: 795:59:

No return:

IRouter01.getAmountsIn(uint256,address[]): Defines a return type but never explicitly returns a value.

Pos: 233:4:

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: 739:8:

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: 819:15:

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.

Alphas.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:46
Error message for require is too long
Pos: 9:88
Function name must be in mixedCase
Pos: 5:199
Contract has 28 states declarations but allowed no more than 15
Pos: 1:265
Constant name must be in capitalized SNAKE_CASE
Pos: 5:267
Constant name must be in capitalized SNAKE_CASE
Pos: 5:268
Constant name must be in capitalized SNAKE_CASE
Pos: 5:269
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:330
Error message for require is too long
Pos: 9:419
Error message for require is too long
Pos: 9:420
Error message for require is too long
Pos: 9:444
Error message for require is too long
Pos: 9:446
Error message for require is too long
Pos: 9:464
Function name must be in mixedCase
Pos: 5:478
Error message for require is too long
Pos: 17:486
Error message for require is too long
Pos: 9:499
Error message for require is too long
Pos: 9:515
Error message for require is too long
Pos: 9:520
Error message for require is too long
Pos: 9:525
Error message for require is too long
```

```
Pos: 9:530
Error message for require is too long
Pos: 9:546
Error message for require is too long
Pos: 9:551
Error message for require is too long
Pos: 9:557
Error message for require is too long
Pos: 9:563
Error message for require is too long
Pos: 9:569
Avoid making time-based decisions in your business logic
Pos: 33:585
Avoid making time-based decisions in your business logic
Pos: 16:587
Avoid making time-based decisions in your business logic
Pos: 21:592
Avoid making time-based decisions in your business logic
Pos: 65:592
Avoid making time-based decisions in your business logic
Pos: 21:597
Avoid making time-based decisions in your business logic
Pos: 66:597
Avoid making time-based decisions in your business logic
Pos: 21:601
Avoid making time-based decisions in your business logic
Pos: 66:601
Avoid making time-based decisions in your business logic
Pos: 21:605
Avoid making time-based decisions in your business logic
Pos: 67:605
Avoid making time-based decisions in your business logic
Pos: 17:615
Avoid making time-based decisions in your business logic
Pos: 37:624
Avoid making time-based decisions in your business logic
Pos: 16:626
Avoid making time-based decisions in your business logic
Pos: 21:631
Avoid making time-based decisions in your business logic
Pos: 66:631
Avoid making time-based decisions in your business logic
Pos: 21:636
Avoid making time-based decisions in your business logic
Pos: 66:636
Avoid making time-based decisions in your business logic
Pos: 17:646
Error message for require is too long
Pos: 9:658
Avoid making time-based decisions in your business logic
Pos: 13:690
Avoid making time-based decisions in your business logic
Pos: 13:732
Error message for require is too long
Pos: 9:737
Error message for require is too long
Pos: 9:738
Error message for require is too long
Pos: 9:739
```

```
Avoid making time-based decisions in your business logic
Pos: 30:744
Error message for require is too long
Pos: 21:791
Error message for require is too long
Pos: 21:803
Code contains empty blocks
Pos: 32:823
```

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