Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Circular Gifting
Website:	https://realgifting.io
Platform:	SCAI Network
Language:	Solidity
Date:	July 14th, 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 Circular Gifting team to perform the Security audit of the Circular Gifting 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 July 14th, 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 Circular Gifting protocol covers multiple contracts, and all contracts have different functions.
    - EQUIIUSQ: This contract is used for governance.
    - Omega: This contract is used for governance.
    - Alpha: This contract is used for stable-coin.
    - EQUIIUSE: This contract is used for stable-coin.
- The smart contracts have functions like rescueETH, addLiquidity, burn, mint, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Circular Gifting Smart Contracts**
Platform	**SCAI Network / Solidity**
File 1	EQUIIUSQ.sol
File 2	Omega.sol
File 3	Alpha.sol
File 4	EQUIIUS_E.sol
Github Commit Hash	700c2e71b3a0f320c0908c2270819720caea732d
Audit Date	July 14th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 EQUIIUSQ.sol • Name: EQUIIUS 'Q' • Symbol: EQQ • Decimals: 18 **The owner has control over the following functions:** • Current owners can transfer ownership. • Owners can renounce ownership. • Set the excluded account from the fee. • Enable open trading. • Update the Burn percentage value and Tax percentage value. • Rescue ether.	**YES, This is valid.**
File 2 Omega.sol • Name: Omega • Symbol: OMA • Decimals: 18 • Maximum Amount: 2% • Burn Tax Percentage: 1% **The owner has control over the following functions:** • Current owners can transfer ownership. • Owners can renounce ownership. • Set the excluded account from the fee. • Enable open trading. • Update the burn percentage value and Tax percentage value. • Rescue ether.	**YES, This is valid.**
File 3 Alpha.sol • Name: Alpha • Symbol: ALP	**YES, This is valid.**

• Decimals: 18 **<u>The owner has control over the following functions:</u>** • Current owners can transfer ownership. • Owners can renounce ownership. • Creates `amount` tokens and assigns them to `to`, increasing the total supply. • Burns `amount` tokens decreases the total supply.	
File 4 EQUIIUSE.sol • Name: EQUIIUS E • Symbol: EQE • Decimals: 18 **<u>The owner has control over the following functions:</u>** • Current owners can transfer ownership. • Owners can renounce ownership. • Creates `amount` tokens and assigns them to `to`, increasing the total supply. • Burns `amount` tokens decreases the total supply.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity 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, 0 low and 1 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	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

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 4 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Circular Gifting 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 Circular Gifting Protocol.

The Circular Gifting 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 well commented on smart contracts.

Documentation

We were given a Circular Gifting smart contract code in the form of a github web link. The hash of that code is mentioned above in the table.

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

Another source of information was its official website: https://realgifting.io 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

EQUIIUSQ.sol

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	_transferTokens	internal	Passed	No Issue
20	TransferEx	write	access only Owner	No Issue
21	isExcludedFromFee	read	Passed	No Issue
22	setExcludedFromFee	external	access only Owner	No Issue
23	_burnTokens	internal	Passed	No Issue
24	OpenTrade	write	access only Owner	No Issue
25	updateBurnPercentage	external	access only Owner	No Issue
26	updateMaxAmountPercentage	external	access only Owner	No Issue
27	addToBlacklist	write	access only Owner	No Issue
28	removeFromBlacklist	write	access only Owner	No Issue
29	isBlacklisted	read	Passed	No Issue
30	rescueETH	external	access only Owner	No Issue
31	addLiquidity	write	Passed	No Issue
32	_transfer	internal	Passed	No Issue
33	_calculateTax	internal	Passed	No Issue
34	fallback	external	Passed	No Issue
35	receive	external	Passed	No Issue

Omega.sol

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	_transferTokens	internal	Passed	No Issue
20	TransferEx	write	access only Owner	No Issue
21	isExcludedFromFee	read	Passed	No Issue
22	setExcludedFromFee	external	access only Owner	No Issue
23	_burnTokens	internal	Passed	No Issue
24	OpenTrade	write	access only Owner	No Issue
25	updateBurnPercentage	external	access only Owner	No Issue
26	updateMaxAmountPercentage	external	access only Owner	No Issue
27	addToBlacklist	write	access only Owner	No Issue
28	removeFromBlacklist	write	access only Owner	No Issue
29	isBlacklisted	read	Passed	No Issue
30	rescueETH	external	access only Owner	No Issue
31	addLiquidity	write	Passed	No Issue
32	_transfer	internal	Passed	No Issue
33	_calculateTax	internal	Passed	No Issue
34	fallback	external	Passed	No Issue
35	receive	external	Passed	No Issue

Alpha.sol

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	increaseAllowance	write	Passed	No Issue
16	decreaseAllowance	write	Passed	No Issue
17	_transfer	internal	Passed	No Issue
18	_mint	internal	Passed	No Issue
19	_burn	internal	Passed	No Issue
20	_approve	internal	Passed	No Issue
21	_spendAllowance	internal	Passed	No Issue
22	_beforeTokenTransfer	internal	Passed	No Issue
23	_afterTokenTransfer	internal	Passed	No Issue
24	mint	external	access only Owner	No Issue
25	burn	external	Passed	No Issue

EQUIIUSE.sol

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	increaseAllowance	write	Passed	No Issue
16	decreaseAllowance	write	Passed	No Issue
17	_transfer	internal	Passed	No Issue
18	_mint	internal	Passed	No Issue

19	_burn	internal	Passed	No Issue
20	_approve	internal	Passed	No Issue
21	_spendAllowance	internal	Passed	No Issue
22	_beforeTokenTransfer	internal	Passed	No Issue
23	_afterTokenTransfer	internal	Passed	No Issue
24	mint	external	access only Owner	No Issue
25	burn	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 in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

No low severity vulnerabilities were found in the contract code.

Very Low / Informational / Best practices:

(1) Ownership control:
Stable coin smart contracts have ownership functions. We advise keeping the private key of the owner's wallet secure. Because if the private key is compromised, then it will have a devastating effect on the project.

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:

EQUIIUSQ.sol

- TransferEx: Transfer token can be set by the owner.
- setExcludedFromFee: Owner can set excluded account from fee.
- OpenTrade: Owners can enable open trading.
- updateBurnPercentage: Burn percentage value can be updated by the owner.
- updateMaxAmountPercentage: Tax percentage value can be updated by the owner.
- addToBlacklist: Account can be added to the blacklist by the owner.
- removeFromBlacklist: Account can be removed from the blacklist by the owner.
- rescueETH: Rescue ether by the owner.

Omega.sol

- TransferEx: Transfer token can be set by the owner.
- setExcludedFromFee: Owner can set excluded account from fee.
- OpenTrade: Owners can enable open trading.
- updateBurnPercentage: Burn percentage value can be updated by the owner.
- updateMaxAmountPercentage: Tax percentage value can be updated by the owner.
- addToBlacklist: Account can be added to the blacklist by the owner.
- removeFromBlacklist: Account can be removed from the blacklist by the owner.
- rescueETH: Rescue ether by the owner.

Alpha.sol

- mint: Creates `amount` tokens and assigns them to `to`, increasing the total supply by the owner.
- burn: Burns `amount` tokens decreasing the total supply by the owner.

EQUIIUSE.sol

- mint: Creates `amount` tokens and assigns them to `to`, increasing the total supply by the owner.
- burn: Burns `amount` tokens decreasing the total supply by the owner.

Ownable.sol

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

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

Conclusion

We were given a contract code in the form of a github web link. And we have used all possible tests based on given objects as files. We had observed some Informational severity issues in the smart contracts. but those are not critical. **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.

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

The 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.

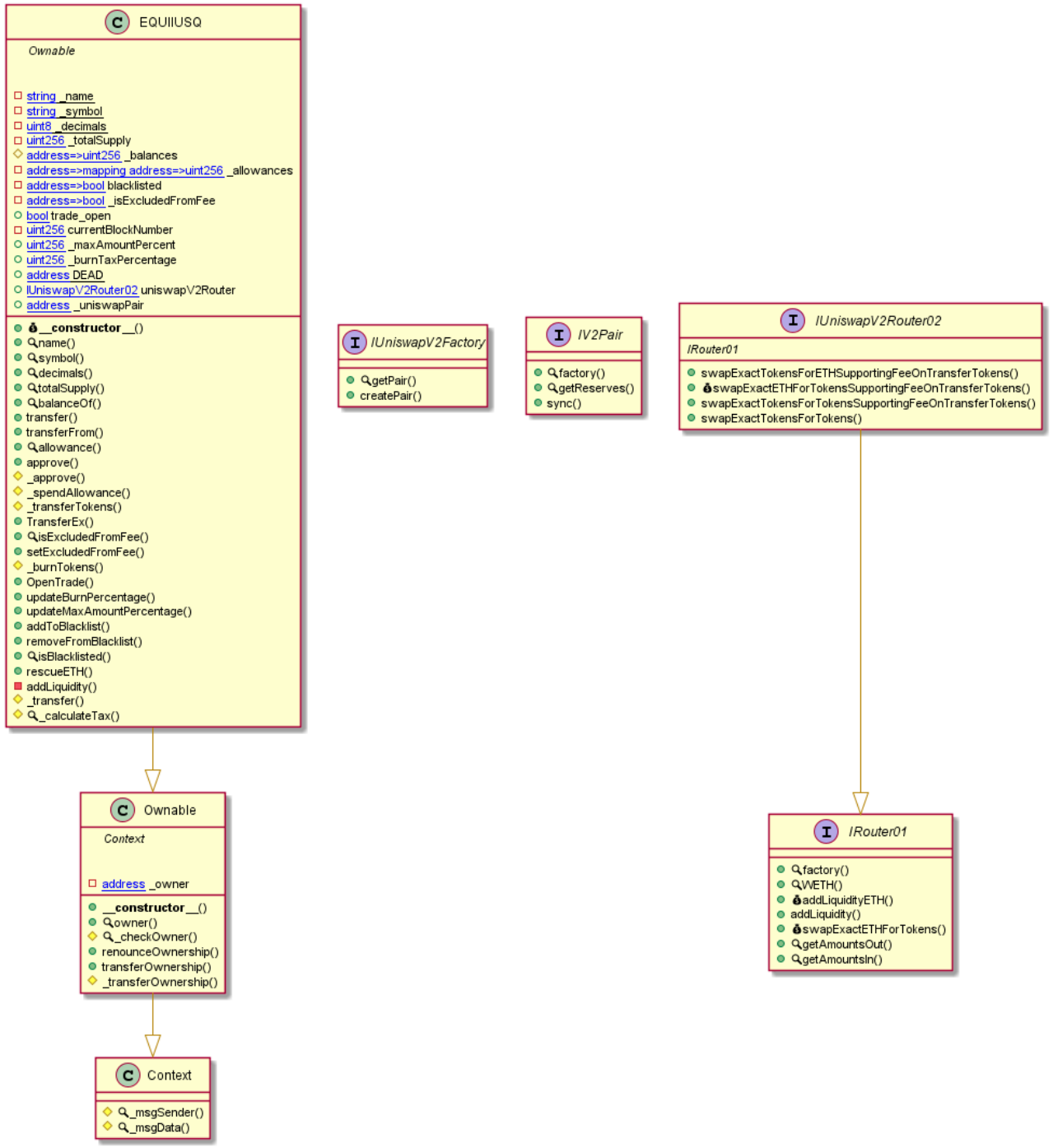
Technical Disclaimer

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.

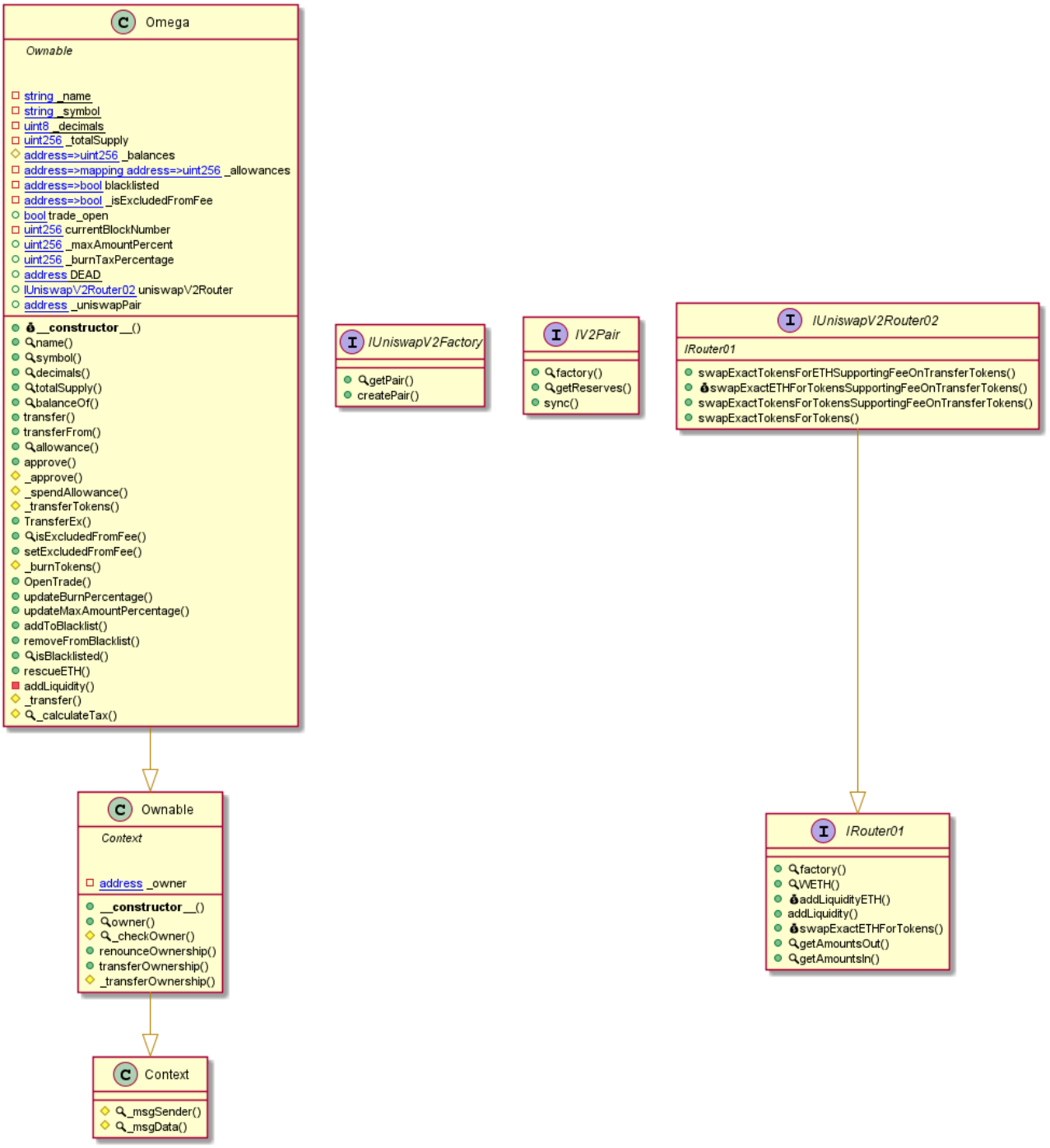
Appendix

Code Flow Diagram - Circular Gifting

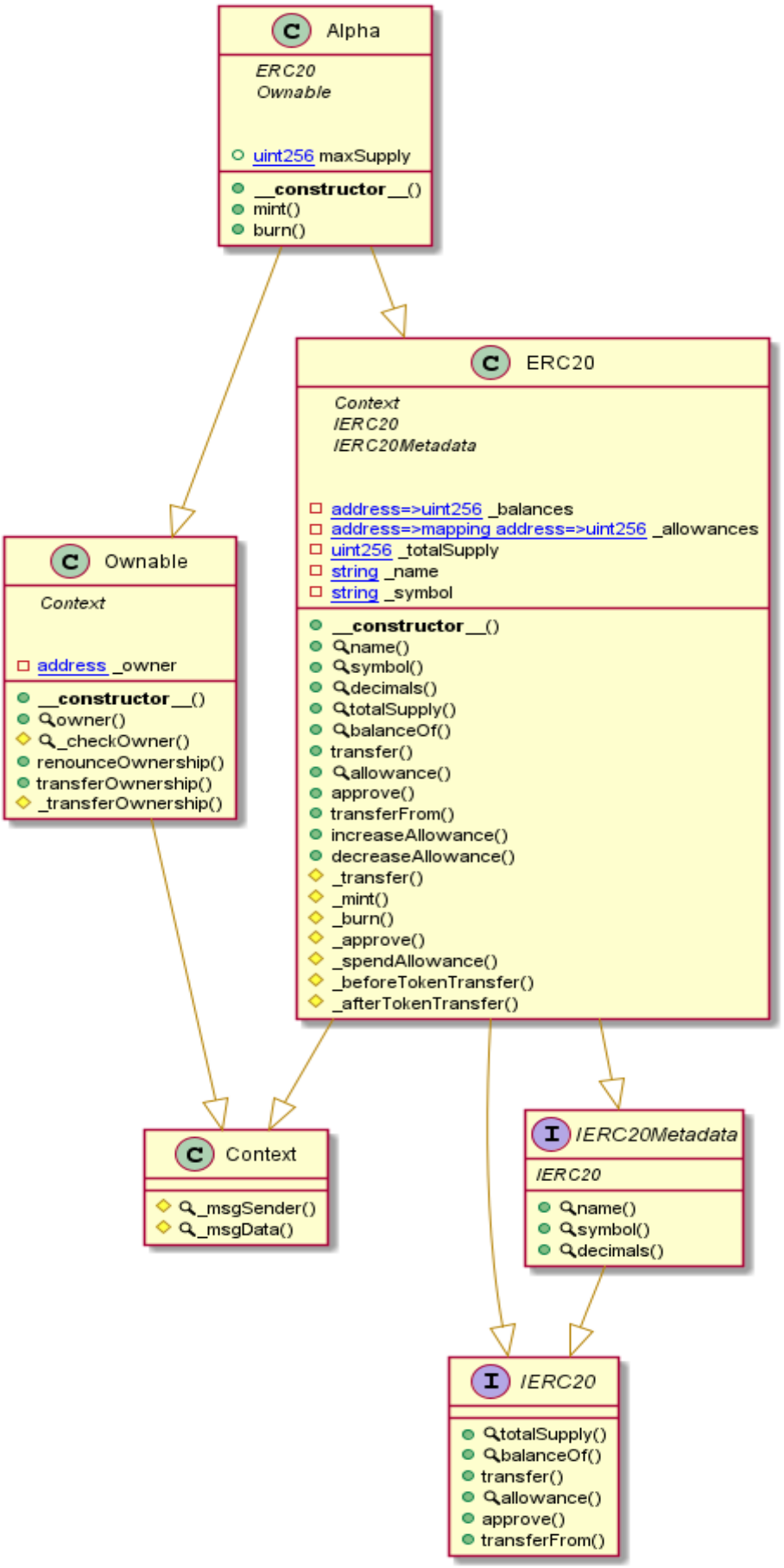
EQUIIUSQ Diagram



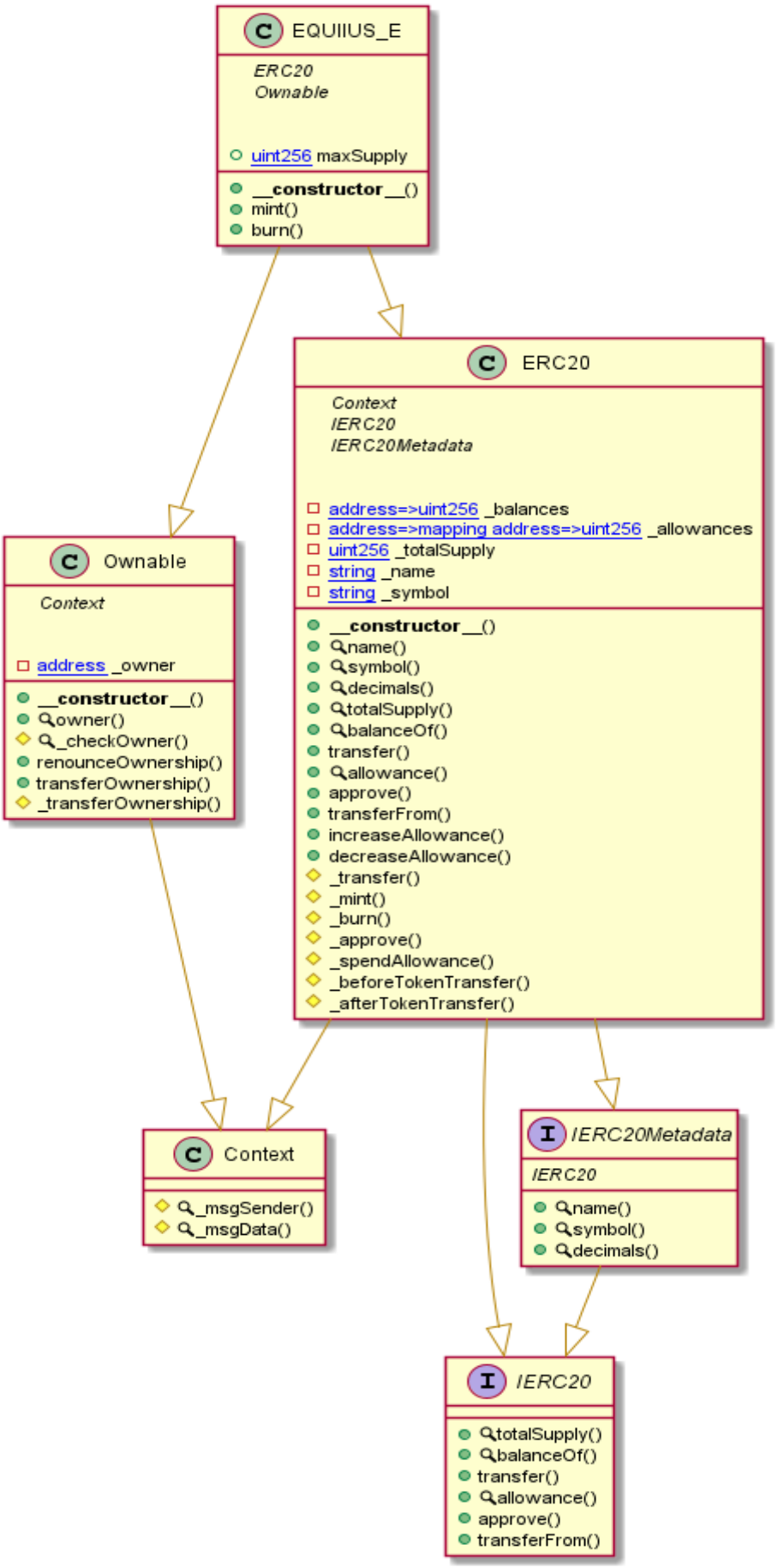
Omega Diagram



Alpha Diagram



EQUIIUSE Diagram



Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> EQUIIUSQ.sol

```
EQUIIUSQ.transfer(address,uint256).owner (EQUIIUSQ.sol#278) shadows:
        - Ownable.owner() (EQUIIUSQ.sol#64-66) (function)
EQUIIUSQ.allowance(address,address).owner (EQUIIUSQ.sol#295) shadows:
        - Ownable.owner() (EQUIIUSQ.sol#64-66) (function)
EQUIIUSQ._spendAllowance(address,address,uint256).owner (EQUIIUSQ.sol#315) shadows:
        - Ownable.owner() (EQUIIUSQ.sol#64-66) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

EQUIIUSQ.updateBurnPercentage(uint256) (EQUIIUSQ.sol#398-401) should emit an event for:
        - _burnTaxPercentage = _burnPercentage (EQUIIUSQ.sol#400)
EQUIIUSQ.updateMaxAmountPercentage(uint256) (EQUIIUSQ.sol#403-406) should emit an event for:
        - _maxAmountPercent = _maxAmountPercentage (EQUIIUSQ.sol#405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

EQUIIUSQ.OpenTrade(bool) (EQUIIUSQ.sol#391-396) compares to a boolean constant:
        -_enable == true (EQUIIUSQ.sol#393)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (EQUIIUSQ.sol#21-23) is never used and should be removed
EQUIIUSQ.addLiquidity(uint256,uint256) (EQUIIUSQ.sol#430-443) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (EQUIIUSQ.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.
6
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Function IRouter01.WETH() (EQUIIUSQ.sol#125) is not in mixedCase
Function EQUIIUSQ.TransferEx(address[],uint256) (EQUIIUSQ.sol#351-366) is not in mixedCase
Parameter EQUIIUSQ.TransferEx(address[],uint256)._input (EQUIIUSQ.sol#352) is not in mixedCase
Parameter EQUIIUSQ.TransferEx(address[],uint256)._amount (EQUIIUSQ.sol#353) is not in mixedCase
Function EQUIIUSQ.OpenTrade(bool) (EQUIIUSQ.sol#391-396) is not in mixedCase
Parameter EQUIIUSQ.OpenTrade(bool)._enable (EQUIIUSQ.sol#391) is not in mixedCase
Parameter EQUIIUSQ.updateBurnPercentage(uint256)._burnPercentage (EQUIIUSQ.sol#398) is not in mixedCase
Parameter EQUIIUSQ.updateMaxAmountPercentage(uint256)._maxAmountPercentage (EQUIIUSQ.sol#403) is not in mixedCase
Constant EQUIIUSQ._name (EQUIIUSQ.sol#185) is not in UPPER_CASE_WITH_UNDERSCORES
Constant EQUIIUSQ._symbol (EQUIIUSQ.sol#186) is not in UPPER_CASE_WITH_UNDERSCORES
Constant EQUIIUSQ._decimals (EQUIIUSQ.sol#187) is not in UPPER_CASE_WITH_UNDERSCORES
Variable EQUIIUSQ._balances (EQUIIUSQ.sol#190) is not in mixedCase
Variable EQUIIUSQ.trade_open (EQUIIUSQ.sol#196) is not in mixedCase
Variable EQUIIUSQ._maxAmountPercent (EQUIIUSQ.sol#199) is not in mixedCase
Variable EQUIIUSQ._burnTaxPercentage (EQUIIUSQ.sol#201) is not in mixedCase
Variable EQUIIUSQ._uniswapPair (EQUIIUSQ.sol#206) 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 (EQUIIUSQ.sol#
) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (EQ
USQ.sol#138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

EQUIIUSQ.updateBurnPercentage(uint256) (EQUIIUSQ.sol#398-401) uses literals with too many digits:
        - require(bool,string)(_burnPercentage <= 100000,Tax percentage cannot exceed 100) (EQUIIUSQ.sol#399)
EQUIIUSQ.updateMaxAmountPercentage(uint256) (EQUIIUSQ.sol#403-406) uses literals with too many digits:
        - require(bool,string)(_maxAmountPercentage <= 100000,Tax percentage cannot exceed 100) (EQUIIUSQ.sol#404)
EQUIIUSQ._transfer(address,address,uint256) (EQUIIUSQ.sol#445-494) uses literals with too many digits:
        - maxAmount = (_totalSupply * _maxAmountPercent) / 100000 (EQUIIUSQ.sol#472)
EQUIIUSQ._calculateTax(uint256,uint256) (EQUIIUSQ.sol#497-499) uses literals with too many digits:
        -amount * (taxPercentage) / (100000) (EQUIIUSQ.sol#498)
EQUIIUSQ.slitherConstructorVariables() (EQUIIUSQ.sol#183-504) uses literals with too many digits:
        - _totalSupply = 1000000000000000 * 10 ** uint256(_decimals) (EQUIIUSQ.sol#188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
EQUIIUSQ._uniswapPair (EQUIIUSQ.sol#206) should be immutable
EQUIIUSQ.uniswapV2Router (EQUIIUSQ.sol#205) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
EQUIIUSQ.sol analyzed (7 contracts with 84 detectors), 36 result(s) found
```

Slither log >> Omega.sol

```
Omega.transfer(address,uint256).owner (Omega.sol#288) shadows:
        - Ownable.owner() (Omega.sol#73-75) (function)
Omega.allowance(address,address).owner (Omega.sol#305) shadows:
        - Ownable.owner() (Omega.sol#73-75) (function)
Omega._spendAllowance(address,address,uint256).owner (Omega.sol#325) shadows:
        - Ownable.owner() (Omega.sol#73-75) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Omega.updateBurnPercentage(uint256) (Omega.sol#408-411) should emit an event for:
        - _burnTaxPercentage = _burnPercentage (Omega.sol#410)
Omega.updateMaxAmountPercentage(uint256) (Omega.sol#413-416) should emit an event for:
        - _maxAmountPercent = _maxAmountPercentage (Omega.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Omega.OpenTrade(bool) (Omega.sol#401-406) compares to a boolean constant:
        -_enable == true (Omega.sol#403)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (Omega.sol#30-32) is never used and should be removed
Omega.addLiquidity(uint256,uint256) (Omega.sol#440-453) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Pragma version0.8.19 (Omega.sol#13) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IRouter01.WETH() (Omega.sol#134) is not in mixedCase
Function Omega.TransferEx(address[],uint256) (Omega.sol#361-376) is not in mixedCase
Parameter Omega.TransferEx(address[],uint256)._input (Omega.sol#362) is not in mixedCase
Parameter Omega.TransferEx(address[],uint256)._amount (Omega.sol#363) is not in mixedCase
Function Omega.OpenTrade(bool) (Omega.sol#401-406) is not in mixedCase
Parameter Omega.OpenTrade(bool)._enable (Omega.sol#401) is not in mixedCase
Parameter Omega.updateBurnPercentage(uint256)._burnPercentage (Omega.sol#408) is not in mixedCase
Parameter Omega.updateMaxAmountPercentage(uint256)._maxAmountPercentage (Omega.sol#413) is not in mixedCase
Constant Omega._name (Omega.sol#194) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Omega._symbol (Omega.sol#195) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Omega._decimals (Omega.sol#196) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Omega._balances (Omega.sol#199) is not in mixedCase
Variable Omega.trade_open (Omega.sol#205) is not in mixedCase
Variable Omega._maxAmountPercent (Omega.sol#208) is not in mixedCase
Variable Omega._burnTaxPercentage (Omega.sol#210) is not in mixedCase
Variable Omega._uniswapPair (Omega.sol#215) 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 (Omega.sol#146
) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Om
ega.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
Omega.updateBurnPercentage(uint256) (Omega.sol#408-411) uses literals with too many digits:
        - require(bool,string)(_burnPercentage <= 100000,Tax percentage cannot exceed 100) (Omega.sol#409)
Omega.updateMaxAmountPercentage(uint256) (Omega.sol#413-416) uses literals with too many digits:
        - require(bool,string)(_maxAmountPercentage <= 100000,Tax percentage cannot exceed 100) (Omega.sol#414)
Omega._transfer(address,address,uint256) (Omega.sol#455-504) uses literals with too many digits:
        - maxAmount = (_totalSupply * _maxAmountPercent) / 100000 (Omega.sol#482)
Omega._calculateTax(uint256,uint256) (Omega.sol#507-509) uses literals with too many digits:
        - amount * (taxPercentage) / (100000) (Omega.sol#508)
Omega.slitherConstructorVariables() (Omega.sol#192-514) uses literals with too many digits:
        - _totalSupply = 1000000000000000 * 10 ** uint256(_decimals) (Omega.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Omega._uniswapPair (Omega.sol#215) should be immutable
Omega.uniswapV2Router (Omega.sol#214) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Omega.sol analyzed (7 contracts with 84 detectors), 36 result(s) found
```

Slither log >> Alpha.sol

```
Alpha.constructor().totalSupply (Alpha.sol#614) shadows:
        - ERC20.totalSupply() (Alpha.sol#309-311) (function)
        - IERC20.totalSupply() (Alpha.sol#133) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (Alpha.sol#31-33) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (Alpha.sol#14) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Alpha.constructor() (Alpha.sol#612-619) uses literals with too many digits:
        - totalSupply = 100000000000000 * (10 ** decimals()) (Alpha.sol#614)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Alpha.maxSupply (Alpha.sol#610) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Alpha.sol analyzed (6 contracts with 84 detectors), 6 result(s) found
```

Slither log >> EQUIIUSE.sol

```
EQUIIUS_E.constructor().totalSupply (EQUIIUSE.sol#615) shadows:
        - ERC20.totalSupply() (EQUIIUSE.sol#310-312) (function)
        - IERC20.totalSupply() (EQUIIUSE.sol#134) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (EQUIIUSE.sol#32-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (EQUIIUSE.sol#15) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0
.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract EQUIIUS_E (EQUIIUSE.sol#609-638) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

EQUIIUS_E.constructor() (EQUIIUSE.sol#613-620) uses literals with too many digits:
        - totalSupply = 1000000000000000 * (10 ** decimals()) (EQUIIUSE.sol#615)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

EQUIIUS_E.maxSupply (EQUIIUSE.sol#611) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
EQUIIUSE.sol analyzed (6 contracts with 84 detectors), 7 result(s) found
```

Solidity Static Analysis

EQUIIUSQ.sol

Check-effects-interaction:

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

more

Pos: 226:4:

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: 441:12:

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: 357:12:

Similar variable names:

EQUIIUSQ._approve(address,address,uint256) : Variables have very similar names "sender" and "spender". Note: Modifiers are currently not considered by this static analysis.

Pos: 307:16:

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: 498:15:

Email: audit@EtherAuthority.io

Omega.sol

Check-effects-interaction:

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

more

Pos: 244:42:

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: 460:51:

Gas costs:

Gas requirement of function Omega.rescueETH 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: 445:55:

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: 379:21:

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: 330:52:

Email: audit@EtherAuthority.io

Alpha.sol

Gas costs:

Gas requirement of function Alpha.decreaseAllowance 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: 421:58:

Gas costs:

Gas requirement of function ERC20.decreaseAllowance 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: 421:58:

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: 455:57:

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: 638:86:

EQUIIUSE.sol

Gas costs:

Gas requirement of function EQUIIUS_E.decreaseAllowance 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: 422:10:

Solhint Linter

EQUIIUSQ.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
1:3
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:48
Error message for require is too long
9:90
Function name must be in mixedCase
5:124
Constant name must be in capitalized SNAKE_CASE
5:184
Constant name must be in capitalized SNAKE_CASE
5:185
Constant name must be in capitalized SNAKE_CASE
5:186
Variable name must be in mixedCase
5:195
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:225
Error message for require is too long
9:306
Error message for require is too long
9:307
Error message for require is too long
9:336
Function name must be in mixedCase
5:350
Error message for require is too long
17:358
Error message for require is too long
9:376
Error message for require is too long
9:378
Function name must be in mixedCase
5:390
Error message for require is too long
9:415
Avoid making time-based decisions in your business logic
13:440
Error message for require is too long
9:445
Error message for require is too long
9:446
Error message for require is too long
9:447
Error message for require is too long
13:474
Error message for require is too long
13:482
Code contains empty blocks
```

```
32:502
```

Omega.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
1:12
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:57
Error message for require is too long
9:99
Function name must be in mixedCase
5:133
Constant name must be in capitalized SNAKE_CASE
5:193
Constant name must be in capitalized SNAKE_CASE
5:194
Constant name must be in capitalized SNAKE_CASE
5:195
Variable name must be in mixedCase
5:204
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:234
Error message for require is too long
9:316
Error message for require is too long
9:317
Error message for require is too long
9:346
Function name must be in mixedCase
5:360
Error message for require is too long
17:368
Error message for require is too long
9:386
Error message for require is too long
9:388
Function name must be in mixedCase
5:400
Error message for require is too long
9:425
Avoid making time-based decisions in your business logic
13:450
Error message for require is too long
9:455
Error message for require is too long
9:456
Error message for require is too long
9:457
Error message for require is too long
13:484
Error message for require is too long
13:492
Code contains empty blocks
```

```
32:512
```

Alpha.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
1:13
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:51
Error message for require is too long
9:93
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:268
Error message for require is too long
9:418
Error message for require is too long
9:445
Error message for require is too long
9:446
Error message for require is too long
9:451
Error message for require is too long
9:500
Error message for require is too long
9:505
Error message for require is too long
9:535
Error message for require is too long
9:536
Code contains empty blocks
24:582
Code contains empty blocks
24:602
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:611
Error message for require is too long
9:625
```

EQUIIUSE.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
1:14
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:52
Error message for require is too long
9:94
```

```
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:269
Error message for require is too long
9:419
Error message for require is too long
9:446
Error message for require is too long
9:447
Error message for require is too long
9:452
Error message for require is too long
9:501
Error message for require is too long
9:506
Error message for require is too long
9:536
Error message for require is too long
9:537
Code contains empty blocks
24:583
Code contains empty blocks
24:603
Contract name must be in CamelCase
1:608
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
5:612
Error message for require is too long
9:626
```

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