

etherauthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: **agEUR (agEUR)**
Website: **angle.money**
Platform: **Base Chain Network**
Language: **Solidity**
Date: **June 15th, 2024**

Table of contents

Introduction	4
Project Background	4
Audit Scope	5
Code Audit History	6
Severity Definitions	6
Claimed Smart Contract Features	7
Audit Summary	8
Technical Quick Stats	9
Business Risk Analysis	10
Code Quality	11
Documentation	11
Use of Dependencies	11
AS-IS overview	12
Audit Findings	13
Conclusion	16
Our Methodology	17
Disclaimers	19
Appendix	
• Code Flow Diagram	20
• Slither Results Log	21
• Solidity static analysis	24
• Solhint Linter	26

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

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the agEUR smart contract from angle.money was audited extensively. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 15th, 2024.

The purpose of this audit was to address the following:

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

Project Background

Website Details



Angle Protocol provides USDA and EURA stablecoins, offering transparency, stability, and liquidity. Users can earn yield, buy, borrow, and trade these stablecoins, which are fully collateralized and audited for security. The platform supports seamless asset swaps, onchain forex, and integration with various DeFi projects. Angle Protocol operates as a DAO, governed by ANGLE token holders.

Code Details

- The `AgTokenSideChainMultiBridgeNameable` contract extends the `AgTokenSideChainMultiBridge` contract, allowing the token's name and symbol to be updated dynamically by a governor.
- This contract is useful for updating the token's branding while maintaining the core functionalities of `AgTokenSideChainMultiBridge`. Only authorized users (governors) can update the name and symbol.
- This contract enables updating the token's name and symbol dynamically, inheriting the core functionalities from `AgTokenSideChainMultiBridge`. It includes access control to ensure that only authorized users (governors) can update these values. This is particularly useful for managing token branding and identification post-deployment.

Audit scope

Name	Code Review and Security Analysis Report for agEUR(agEUR) Smart Contract
Platform	Base Chain Network
Language	Solidity
File	AgTokenSideChainMultiBridgeNameable.sol
Smart Contract Code	0xb5ecaa1a867feccd6d87604bc16a2b6b53d706bf
Audit Date	June 15th,2024
Audit Result	Passed

Code Audit History



0
Total Findings

0
Critical






0
High

0
Medium

0
Low

0
Informational

Severity Definitions

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

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
<p>Governor Control:</p> <ul style="list-style-type: none">• Updates the name and symbol of the token.• Add/Remove support for a bridge token.• Recovers any ERC20 token. <p>Governor Or Guardian Control:</p> <ul style="list-style-type: none">• Updates the `limit` amount for `bridgeToken`.• Updates the `hourly limit` amount for `bridgeToken`.• Updates the `chainTotalHourlyLimit` amount.• Updates the `fee` value for `bridgeToken`.• Pauses or unpauses swapping in and out for a token.• Toggles fees for the address `address`.	<p>YES, This is valid.</p>

Audit Summary

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



You are here 

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

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

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

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	The solidity version is not specified	Passed
	The solidity version is too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	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 is not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Business Risk Analysis

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

Overall Audit Result: PASSED

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

Email: audit@EtherAuthority.io

Code Quality

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

The libraries in agEUR 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 agEUR.

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

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

Documentation

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

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

Use of Dependencies

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

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

AS-IS overview

AgTokenSideChainMultiBridgeNameable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	setNameAndSymbol	external	access only governor	No Issue
5	_swapLeverage	internal	Passed	No Issue
6	onlyGovernor	modifier	Passed	No Issue
7	onlyGovernorOrGuardian	modifier	Passed	No Issue
8	allBridgeTokens	external	Passed	No Issue
9	currentUsage	external	Passed	No Issue
10	currentTotalUsage	external	Passed	No Issue
11	swapIn	external	Passed	No Issue
12	swapOut	external	Passed	No Issue
13	addBridgeToken	external	access only governor	No Issue
14	removeBridgeToken	external	access only governor	No Issue
15	recoverERC20	external	access only governor	No Issue
16	setLimit	external	access only governor	No Issue
17	setHourlyLimit	external	access only Governor Or Guardian	No Issue
18	setChainTotalHourlyLimit	external	access only Governor Or Guardian	No Issue
19	setSwapFee	external	access only Governor Or Guardian	No Issue
20	toggleBridge	external	access only Governor Or Guardian	No Issue
21	toggleFeesForAddress	external	access only Governor Or Guardian	No Issue
22	initialize	external	Passed	No Issue
23	_initialize	internal	Passed	No Issue
24	onlyTreasury	modifier	Passed	No Issue
25	onlyMinter	modifier	Passed	No Issue
26	burnStablecoin	external	Passed	No Issue

27	burnSelf	external	access only Minter	No Issue
28	burnFrom	external	access only Minter	No Issue
29	mint	external	access only Minter	No Issue
30	addMinter	external	access only treasury	No Issue
31	removeMinter	external	Passed	No Issue
32	setTreasury	external	access only treasury	No Issue

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium-severity vulnerabilities were found.

Low

No Low-severity vulnerabilities were found.

Very Low / Informational / Best practices:

No Very-Low severity vulnerabilities were found.

Centralization

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

Centralized

Decentralized

You are here



The following are owner functions:

AgTokenSideChainMultiBridgeNameable.sol

- setNameAndSymbol: Updates the name and symbol of the token only by the Governor.
- addBridgeToken: Adds support for a bridge token only by the Governor.
- removeBridgeToken: Removes support for a token only by the Governor.
- recoverERC20: Recovers any ERC20 token only by the Governor.
- setLimit: Updates the `limit` amount for `bridgeToken` only by the Governor Or Guardian.
- setHourlyLimit: Updates the `hourlyLimit` amount for `bridgeToken` only by the Governor Or Guardian.
- setChainTotalHourlyLimit: Updates the `chainTotalHourlyLimit` amount only by the Governor Or Guardian.
- setSwapFee: Updates the `fee` value for `bridgeToken` only by the Governor Or Guardian.
- toggleBridge: Pauses or unpauses swapping in and out for a token only by the Governor Or Guardian.
- toggleFeesForAddress: Toggles fees for the address `theAddress` only by the Governor Or Guardian.

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

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

Email: audit@EtherAuthority.io

Conclusion

We were given a contract code in the form of a [basescan](#) web link. And we have used all possible tests based on given objects as files. We observed no 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 the maximum possible test cases to scan everything.

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and 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

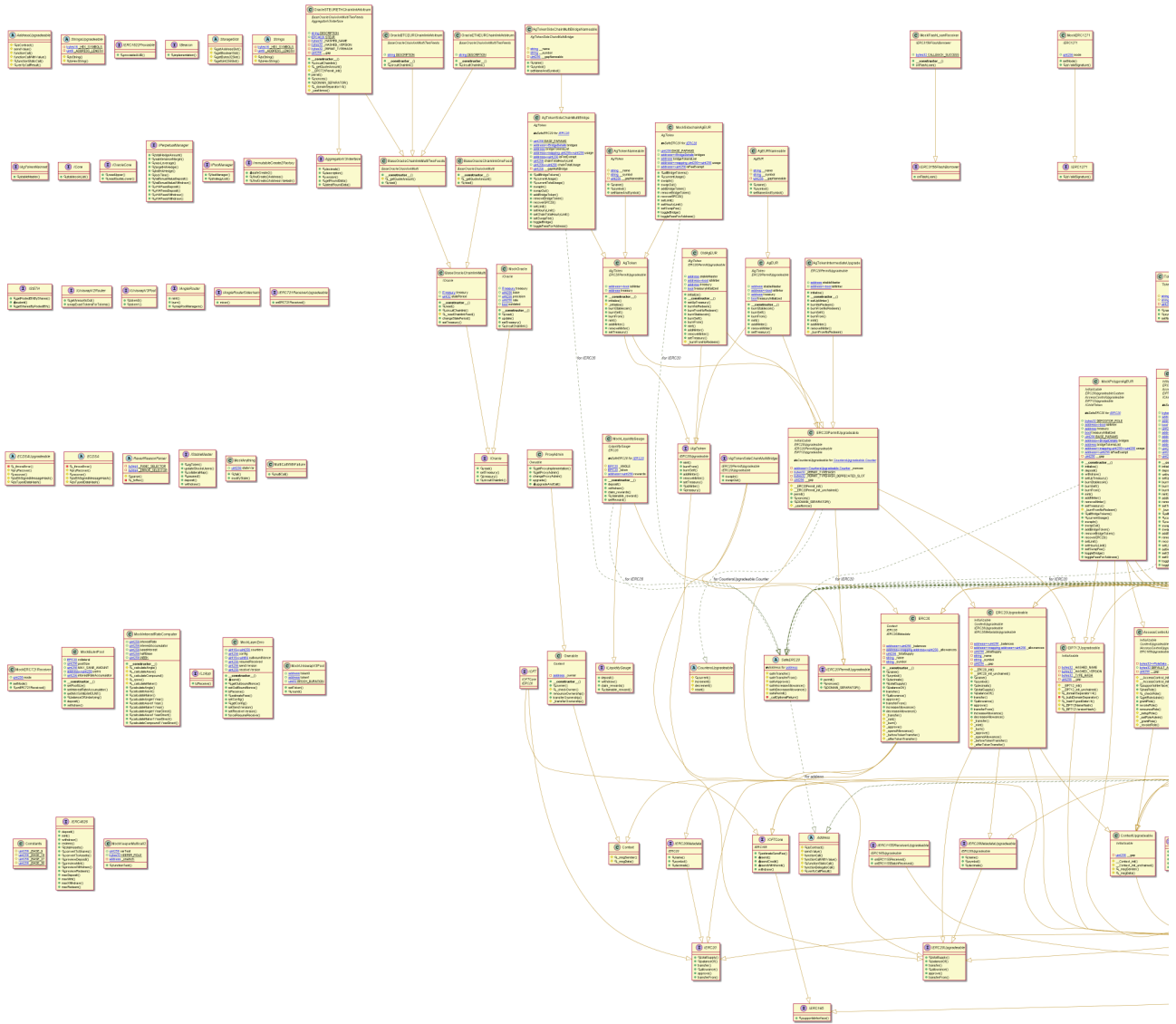
EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation,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

Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free 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 the explicit security of the audited smart contracts.

Code Flow Diagram - agEUR



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.

AgTokenSideChainMultiBridgeNameable.sol

```
INFO:Detectors:
AgEUR.setTreasury(address)._treasury (AgTokenSideChainMultiBridgeNameable.sol#4955)
lacks a zero-check on :
    - treasury = _treasury (AgTokenSideChainMultiBridgeNameable.sol#4956)
AgToken.setTreasury(address)._treasury (AgTokenSideChainMultiBridgeNameable.sol#5033)
lacks a zero-check on :
    - treasury = _treasury (AgTokenSideChainMultiBridgeNameable.sol#5034)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Modifier TransparentUpgradeableProxy.ifAdmin()
(AgTokenSideChainMultiBridgeNameable.sol#6829-6835) does not always execute _; or
revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
INFO:Detectors:
Reentrancy in AgTokenSideChainMultiBridge.swapIn(address,uint256,address)
(AgTokenSideChainMultiBridgeNameable.sol#5100-5125):
    External calls:
    - IERC20(bridgeToken).safeTransferFrom(msg.sender,address(this),amount)
(AgTokenSideChainMultiBridgeNameable.sol#5118)
    State variables written after the call(s):
    - _mint(to,canonicalOut) (AgTokenSideChainMultiBridgeNameable.sol#5123)
      - _balances[account] += amount (AgTokenSideChainMultiBridgeNameable.sol#2246)
    - _mint(to,canonicalOut) (AgTokenSideChainMultiBridgeNameable.sol#5123)
      - _totalSupply += amount (AgTokenSideChainMultiBridgeNameable.sol#2245)
INFO:Detectors:
Reentrancy in AgTokenSideChainMultiBridge.recoverERC20(address,address,uint256)
(AgTokenSideChainMultiBridgeNameable.sol#5180-5183):
    External calls:
    - IERC20(tokenAddress).safeTransfer(to,amountToRecover)
(AgTokenSideChainMultiBridgeNameable.sol#5181)
    Event emitted after the call(s):
    - Recovered(tokenAddress,to,amountToRecover)
(AgTokenSideChainMultiBridgeNameable.sol#5182)
```

INFO:Detectors:

Pragma version^0.8.0 (AgTokenSideChainMultiBridgeNameable.sol#2) allows old versions solc-0.8.25 is not recommended for deployment

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

INFO:Detectors:

Variable AgTokenSideChainMultiBridgeNameable.__name

(AgTokenSideChainMultiBridgeNameable.sol#8752) is not in mixedCase

Variable AgTokenSideChainMultiBridgeNameable.__symbol

(AgTokenSideChainMultiBridgeNameable.sol#8754) is not in mixedCase

Variable AgTokenSideChainMultiBridgeNameable.__gapNameable

(AgTokenSideChainMultiBridgeNameable.sol#8756) is not in mixedCase

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

INFO:Detectors:

Variable OldVaultManagerStorage.xLiquidationBoost

(AgTokenSideChainMultiBridgeNameable.sol#6770) is too similar to

OldVaultManagerStorage.yLiquidationBoost

(AgTokenSideChainMultiBridgeNameable.sol#6771)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar>

INFO:Detectors:

LayerZeroBridge.slitherConstructorConstantVariables()

(AgTokenSideChainMultiBridgeNameable.sol#7911-7994) uses literals with too many digits:

- EXTRA_GAS = 200000 (AgTokenSideChainMultiBridgeNameable.sol#5410)

LayerZeroBridgeToken.slitherConstructorConstantVariables()

(AgTokenSideChainMultiBridgeNameable.sol#8109-8213) uses literals with too many digits:

- EXTRA_GAS = 200000 (AgTokenSideChainMultiBridgeNameable.sol#5410)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits>

INFO:Detectors:

BaseOracleChainlinkOneFeed (AgTokenSideChainMultiBridgeNameable.sol#7765-7776) does not implement functions:

- BaseOracleChainlinkMulti.circuitChainlink()

(AgTokenSideChainMultiBridgeNameable.sol#7740)

OracleSTEURETHChainlinkArbitrum (AgTokenSideChainMultiBridgeNameable.sol#7834-7908)

does not implement functions:

- AggregatorV3Interface.decimals() (AgTokenSideChainMultiBridgeNameable.sol#6-11)

- AggregatorV3Interface.description() (AgTokenSideChainMultiBridgeNameable.sol#13-18)

- AggregatorV3Interface.getRoundData(uint80)

(AgTokenSideChainMultiBridgeNameable.sol#27-38)

- AggregatorV3Interface.latestRoundData()

(AgTokenSideChainMultiBridgeNameable.sol#40-49)

- AggregatorV3Interface.version() (AgTokenSideChainMultiBridgeNameable.sol#20-25)

LayerZeroBridge (AgTokenSideChainMultiBridgeNameable.sol#7911-7994) does not implement functions:

- IERC165.supportsInterface(bytes4) (AgTokenSideChainMultiBridgeNameable.sol#653)

LayerZeroBridgeToken (AgTokenSideChainMultiBridgeNameable.sol#8109-8213) does not

implement functions:

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

Email: audit@EtherAuthority.io

- IERC165.supportsInterface(bytes4) (AgTokenSideChainMultiBridgeNameable.sol#653)
OldLayerZeroBridgeToken (AgTokenSideChainMultiBridgeNameable.sol#8238-8342) does not
implement functions:

- IERC165.supportsInterface(bytes4) (AgTokenSideChainMultiBridgeNameable.sol#653)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions>

INFO:Detectors:

OwnableUpgradeable.__gap (AgTokenSideChainMultiBridgeNameable.sol#2404) is never used
in OwnableUpgradeable (AgTokenSideChainMultiBridgeNameable.sol#2363-2405)

KeeperRegistry.__gap (AgTokenSideChainMultiBridgeNameable.sol#3832) is never used in
KeeperRegistry (AgTokenSideChainMultiBridgeNameable.sol#3825-3861)

AgEURNameable.__gapNameable (AgTokenSideChainMultiBridgeNameable.sol#5556) is never
used in AgEURNameable (AgTokenSideChainMultiBridgeNameable.sol#5546-5570)

AgTokenNameable.__gapNameable (AgTokenSideChainMultiBridgeNameable.sol#5584) is
never used in AgTokenNameable (AgTokenSideChainMultiBridgeNameable.sol#5573-5598)

AgTokenSideChainMultiBridgeNameable.__gapNameable

(AgTokenSideChainMultiBridgeNameable.sol#8756) is never used in

AgTokenSideChainMultiBridgeNameable

(AgTokenSideChainMultiBridgeNameable.sol#8751-8770)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable>

INFO:Detectors:

AgEUR.stableMaster (AgTokenSideChainMultiBridgeNameable.sol#4895) should be constant

AgEUR.treasuryInitialized (AgTokenSideChainMultiBridgeNameable.sol#4902) should be
constant

Settlement.collateralStablecoinExchangeRate

(AgTokenSideChainMultiBridgeNameable.sol#3885) should be constant

Settlement.exchangeRateComputed (AgTokenSideChainMultiBridgeNameable.sol#3887) should
be constant

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant>

INFO:Detectors:

MockVaultManager.oracle (AgTokenSideChainMultiBridgeNameable.sol#3694) should be
immutable

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable>

INFO:Slither:AgTokenSideChainMultiBridgeNameable.sol analyzed (162 contracts with 93
detectors), 749 result(s) found

INFO:Slither:AgTokenSideChainMultiBridgeNameable.sol analyzed (162 contracts with 93
detectors), 749 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.

AgTokenSideChainMultiBridgeNameable.sol

Inline assembly: The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

Pos: 9003:12:

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.

Pos: 8835:12:

Low level calls:

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

Pos: 8994:46:

Gas costs:

Gas requirement of function MockSidechainAgEUR.removeBridgeToken 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: 7681:4:

This on local calls:

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

Pos: 6462:12:

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.

Pos: 4513:8:

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.

Pos: 6703:8:

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.

Pos: 8966:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

Pos: 8744:8:

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: 8948:40:

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.

AgTokenSideChainMultiBridgeNameable.sol

```
Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement
Pos: 1:1
Function name must be in mixedCase
Pos: 5:89
Error message for require is too long
Pos: 9:136
Error message for require is too long
Pos: 9:5857
Code contains empty blocks
Pos: 94:5863
Code contains empty blocks
Pos: 93:5865
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:5888
Code contains empty blocks
Pos: 31:8919
Avoid to use low level calls.
Pos: 47:8993
Avoid to use inline assembly. It is acceptable only in rare cases
Pos: 13:9002
Code contains empty blocks
Pos: 32:9018
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:9036
Code contains empty blocks
Pos: 60:9041
```

Software analysis result:

This software reported many false positive results and some were informational issues. So, those issues can be safe



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

Email: audit@EtherAuthority.io