Cyberscope

Audit Report

AN on BSC

June 2024

Network	BSC
Address	0x68D806380CE01e994f7583D796D0aEA9aB470219
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	TSD	Total Supply Diversion	Acknowledged
●	MSC	Missing Sanity Check	Unresolved
●	RC	Repetitive Calculations	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

Table of Contents

Review

Contract Name	ANToken
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x68d806380ce01e994f7583d796d0aea9ab470219
Address	0x68d806380ce01e994f7583d796d0aea9ab470219
Network	BSC
Symbol	AN
Decimals	18
Total Supply	100,000,000,000

Audit Updates

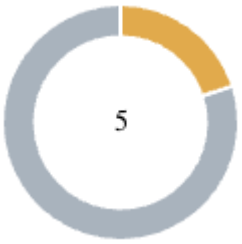
Initial Audit	11 Jun 2024

Source Files

Filename	SHA256
contracts/ANToken.sol	173a6cf54ff0d1b4852ff468cd88837cfa10818c0a6c084db167ea3d43b4e729
contracts/interfaces/IWormholeRelayer.sol	c091c37347461cb687c6d34a444e3bec1e420aa8a01cf2d30ff55e66b3bb1371
contracts/interfaces/IWormholeReceiver.sol	8776834a95a3f62ac4a1c36876b37571f349037cea8a23871c7d3844f8013021

@prb/math/contracts/PRBMathUD60x18.sol	f433763fb95aea1fec7bea4829262c4d95c73acb21ef5d926c9e3aba89e544ca
@prb/math/contracts/PRBMath.sol	3ef522031838f10557feebf5cfd49cb2d0c15f42038b648bc33a7f3ebb3d6500
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	a64e5d0e83019d9caa51e6fe6f68ac54b583ac15792b8557cb8e4fab20711b9f
@openzeppelin/contracts/token/ERC20/IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/access/Ownable.sol	a8e4e1ae19d9bd3e8b0a6d46577eec098c01fbaffd3ec1252fd20d799e73393b

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	1	0	0
Minor / Informative	4	0	0	0

TSD - Total Supply Diversion

Criticality	Medium
Location	contracts/ANToken.sol#L113
Status	Acknowledged

Description

The contract implements a burning and reflection mechanism where it can burn up to 4% of the total supply once a month through the `burn` function. This mechanism introduces the `cumulativeAdjustmentFactor` variable, which is used for balance normalization in transactions for users not excluded from this mechanism. However, continuous multiplication and division by the `cumulativeAdjustmentFactor` can lead to precision loss over time. As a result, the sum of individual user balances may diverge from the total supply.

The following issues may arise from this implementation:

1. Continuous normalization of user balances through multiplication and division by the `cumulativeAdjustmentFactor` can cause precision loss. Eventually, user balances may not accurately reflect their holdings.
2. If the owner calls the `burn` function and all holders are excluded from the burn mechanism, the sum of balances of excluded accounts could exceed the total supply. This discrepancy can cause an underflow in the calculation `currentTotalSupply - nonBurnableSupply`, leading to a transaction revert.

Scenario

Initial State This is the initial state of the example scenario.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	500000000000000000000000000	500000000000000000000000000	Yes	100000000000000000000000000	100000000000000000000000000	100000000000000000
B	500000000000000000000000000	500000000000000000000000000	No			

Burn

The owner calls the burn function. The total supply, adjustment factor and balanceOf of the not protected users is affected.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	500000000000000000000000000	500000000000000000000000000	Yes	995000000000000000000000000	995000000000000000004950000	101010101010101010101
B	500000000000000000000000000	495000000000000000000000000	No			

Remove A from protected

User A is removed from protected accounts. The value of `_balances` changes.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	5050505050505050505000000000	5000000000000000000000000000	No	995000000000000000000000000	995000000000000000000000000	101010101010101010101
B	5000000000000000000000000000	4950000000000000000000000000	No			

Transfer tokens from A to B

User A transfers 5050505050505050505000000000 tokens to user B. `_balances` , `balanceOf` and sum of balanceOf values change.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	4999489847974696459545964697 5	4949494949494949495000000000	No	995000000000000000000000000	995000000000000000000099999	101010101010101010101
B	5051015202530354045454035302 5	5000505050505050505049499999 9	No			

Burn

The owner calls the burn function. The total supply, adjustment factor are affected.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	499948984797469645954596469751	490000000000000000051000000	No	985050000000000000014751000000	985050000000000000102420449999	102030405060708091010
B	505101520253035404545403530251	495050000000000000048910449999	No			

Add B to protected

User B is added to protected accounts. The value of `_balances` changes.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	499948984797469645954596469751	490000000000000000051000000	No	985050000000000000014751000000	985050000000000000102420449999	102030405060708091010
B	495050000000000000048910449999	495050000000000000048910449999	Yes			

Transfer tokens from A to B

User A transfers 994898479746964595459646975 tokens to user B. _balances, balanceOf and sum of balanceOf values change.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	4897979953091830976906315371 8	4800510152025303540989135302 4	No	9850500000000000000014751000 00	985050000000000000010242044998	1020304050607080910
B	5049989847974696460035069197 4	5049989847974696460035069197 4	Yes			

Add A to protected

User A is added to protected accounts. The value of _balances changes.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	4800510152025303540989135302 4	4800510152025303540989135302 4	Yes	9850500000000000000014751000 00	985050000000000000010242044998	1020304050607080910
B	5049989847974696460035069197 4	5049989847974696460035069197 4	Yes			

Burn

The owner calls the burn function. The sum of balances of the non protected accounts is greater than the `totalSupply`. So the subtraction will overflow and revert the transaction.

User	_balances[account]	balanceOf(account)	Burn Protected	totalSupply	Total Supply (Sum of balances)	Adjustment factor
A	4800510152025303540989 1353024		Yes			
B	5049989847974696460035 0691974		Yes			

```
function burn(uint256 percentage_) external onlyOwner {
    ...
    uint256 currentTotalSupply = _totalSupply;
    uint256 nonBurnableSupply = totalSupplyOfBurnProtectedAccounts();
    uint256 burnableSupply = currentTotalSupply - nonBurnableSupply;
    uint256 burnAmount = burnableSupply * percentage_ / BASE_PERCENTAGE;
    uint256 adjustmentFactor = burnableSupply.div(burnableSupply - burnAmount);
    cumulativeAdjustmentFactor = cumulativeAdjustmentFactor.mul(adjustmentFactor);
    _totalSupply = nonBurnableSupply + burnableSupply.div(adjustmentFactor);
    lastBurnTimestamp = block.timestamp;
    emit Transfer(address(this), address(0), currentTotalSupply - _totalSupply);
}
```

Recommendation

The team is advised to address this issue by implementing comprehensive checks to prevent underflows during the burning process. Specifically, the team should ensure that the total supply is greater than the non-burnable supply before proceeding with the subtraction. If this condition is met, the subtraction can be performed safely. Otherwise, the `burnableSupply` variable should be assigned a value of 0.

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/ANToken.sol#L196,237
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
function transferMultichain(
    uint16 targetChain_,
    address targetAddress_,
    address to_,
    uint256 amount_
)
    external
    payable
    returns (bool)
{ ... }

function transferFromMultichain(
    uint16 targetChain_,
    address targetAddress_,
    address from_,
    address to_,
    uint256 amount_
)
    external
    payable
    returns (bool)
{ ... }
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	contracts/ANToken.sol#L133
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The contract does not check if the `accounts_[i]` is the zero address.

```
if (!_liquidityPools.add(accounts_[i])) {
    revert AlreadyInLiquidityPoolsSet(accounts_[i]);
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

RC - Repetitive Calculations

Criticality	Minor / Informative
Location	contracts/ANToken.sol#L389,393,397
Status	Unresolved

Description

The contract contains methods with multiple occurrences of the same calculation being performed. The calculation is repeated without utilizing a variable to store its result, which leads to redundant code, hinders code readability, and increases gas consumption. Each repetition of the calculation requires computational resources and can impact the performance of the contract, especially if the calculation is resource-intensive.

```
uint256 adjustedAmount = amount_.mul(cumulativeAdjustmentFactor);
```

Recommendation

To address this finding and enhance the efficiency and maintainability of the contract, it is recommended to refactor the code by assigning the calculation result to a variable once and then utilizing that variable throughout the method. By storing the calculation result in a variable, the contract eliminates the need for redundant calculations and optimizes code execution.

Refactoring the code to assign the calculation result to a variable has several benefits. It improves code readability by making the purpose and intent of the calculation explicit. It also reduces code redundancy, making the method more concise, easier to maintain, and gas effective. Additionally, by performing the calculation once and reusing the variable, the contract improves performance by avoiding unnecessary computations

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/ANToken.sol#L124,125
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 adjustmentFactor = burnableSupply.div(burnableSupply - burnAmount)
cumulativeAdjustmentFactor =
cumulativeAdjustmentFactor.mul(adjustmentFactor)
```

Recommendation

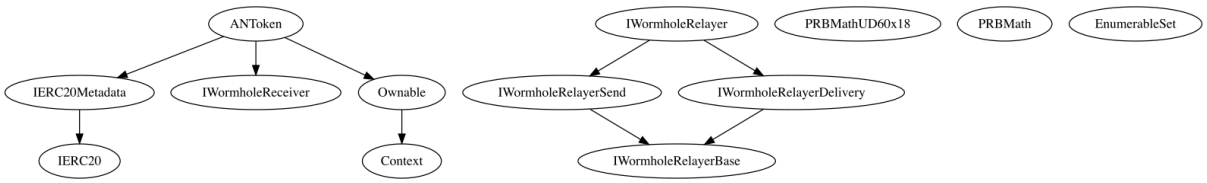
To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Functions Analysis

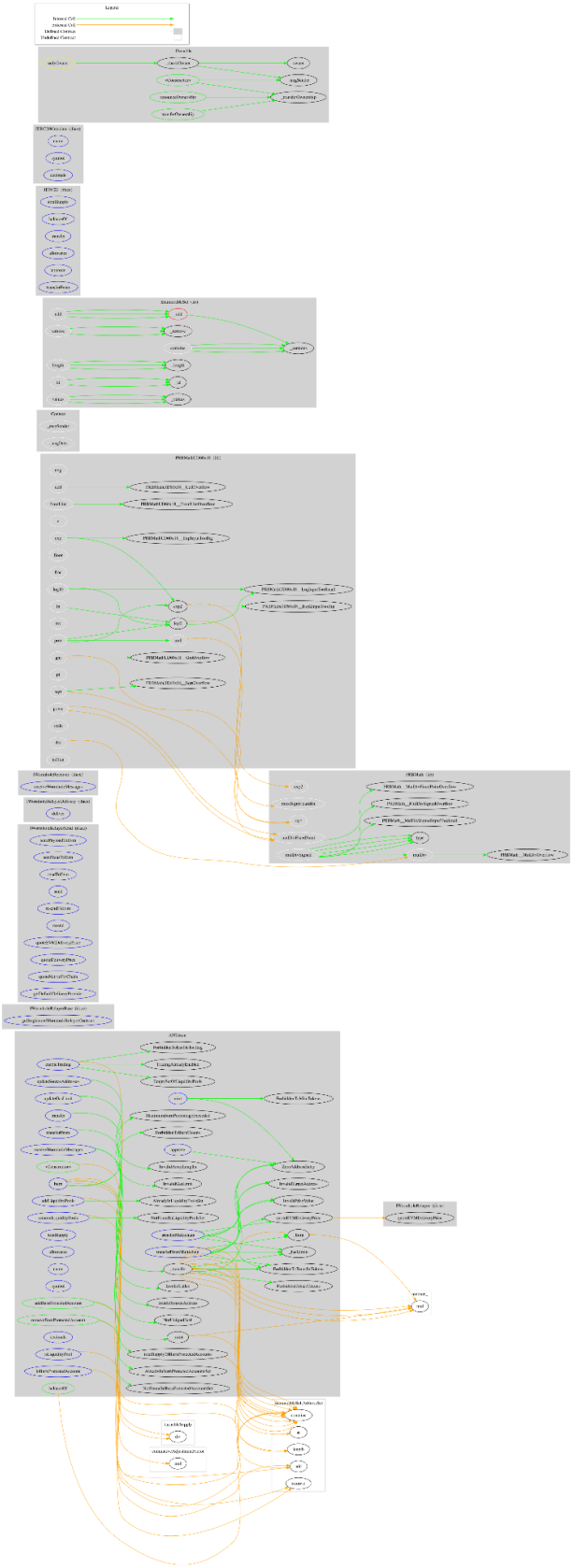
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ANToken	Implementation	IERC20Metadata, IWormholeReceiver, Ownable		
		Public	✓	-
	enableTrading	External	✓	onlyOwner
	mint	External	✓	onlyOwner
	burn	External	✓	onlyOwner
	addLiquidityPools	External	✓	onlyOwner
	removeLiquidityPools	External	✓	onlyOwner
	updateSourceAddresses	External	✓	onlyOwner
	updateGasLimit	External	✓	onlyOwner
	approve	External	✓	-
	transfer	External	✓	-
	transferMultichain	External	Payable	-
	transferFrom	External	✓	-
	transferFromMultichain	External	Payable	-
	receiveWormholeMessages	External	Payable	-
	totalSupply	External		-
	allowance	External		-
	name	External		-

	symbol	External		-
	isLiquidityPool	External		-
	isBurnProtectedAccount	External		-
	decimals	External		-
	addBurnProtectedAccount	Public	✓	onlyOwner
	removeBurnProtectedAccount	Public	✓	onlyOwner
	balanceOf	Public		-
	quoteEVMDeliveryPrice	Public		-
	totalSupplyOfBurnProtectedAccounts	Public		-
	_transfer	Private	✓	
	_mint	Private	✓	
	_burn	Private	✓	
	_hasLimits	Private		

Inheritance Graph



Flow Graph



Summary

AN on BSC contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. AN on BSC is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io