AUDIT REPORT

DATE APRIL 1ST

FOR THUNDERBOLT.FINANCE PROJECT

Solid Group Auditing Service
Telegram: @solid_1
Twitter: https://twitter.com/solid_group_1

Disclaimer

SolidGroup reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team.
SolidGroup Audits do 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. SolidGroup Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**
SolidGroup Reports represent an extensive auditing 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. **SolidGroup’s position is that each company and individual are responsible for their own due diligence and continuous security.** SolidGroup in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Overview

Network: on-chain Binance Smart Chain
Website: https://thunderboltfinance.io/
Telegram Group: https://t.me/ThunderBoltBurn
Twitter: https://twitter.com/ThunderBoltBurn

Description

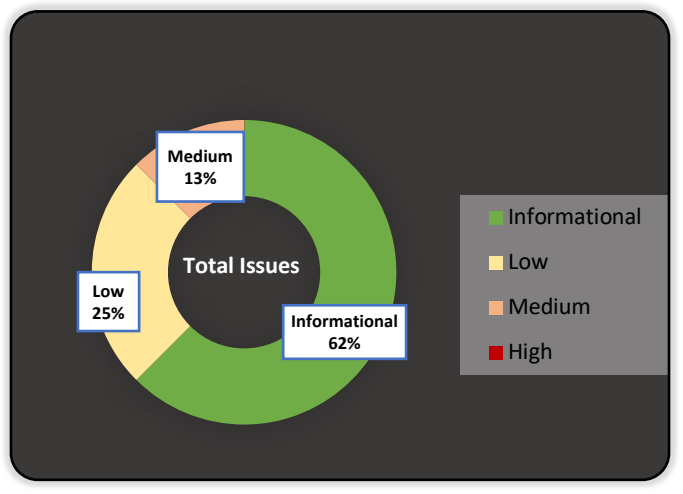
ThunderBolt is a deflationary token with positive rebase. Total of 156 cycles, each cycle has a TAX amount on buy/sell and send TXs. Tax starts at 5% and scales to 12% per cycle. The fee gradually increases as 1.275M tokens are burnt per cycle. Cycle concludes when it reaches a total of 1.275M tokens burnt. Upon that 50% of the total burnt is minted back to $BOLT holders. Cycle starts again at 5% tax.

Files In Scope

Contract Name	Address	MD5
ThunderBolt.sol	0x01eC3200a0895F2C92a9E38a104362442D77658b	8CD9138DEAF914BA29E313D96D4EFAA1

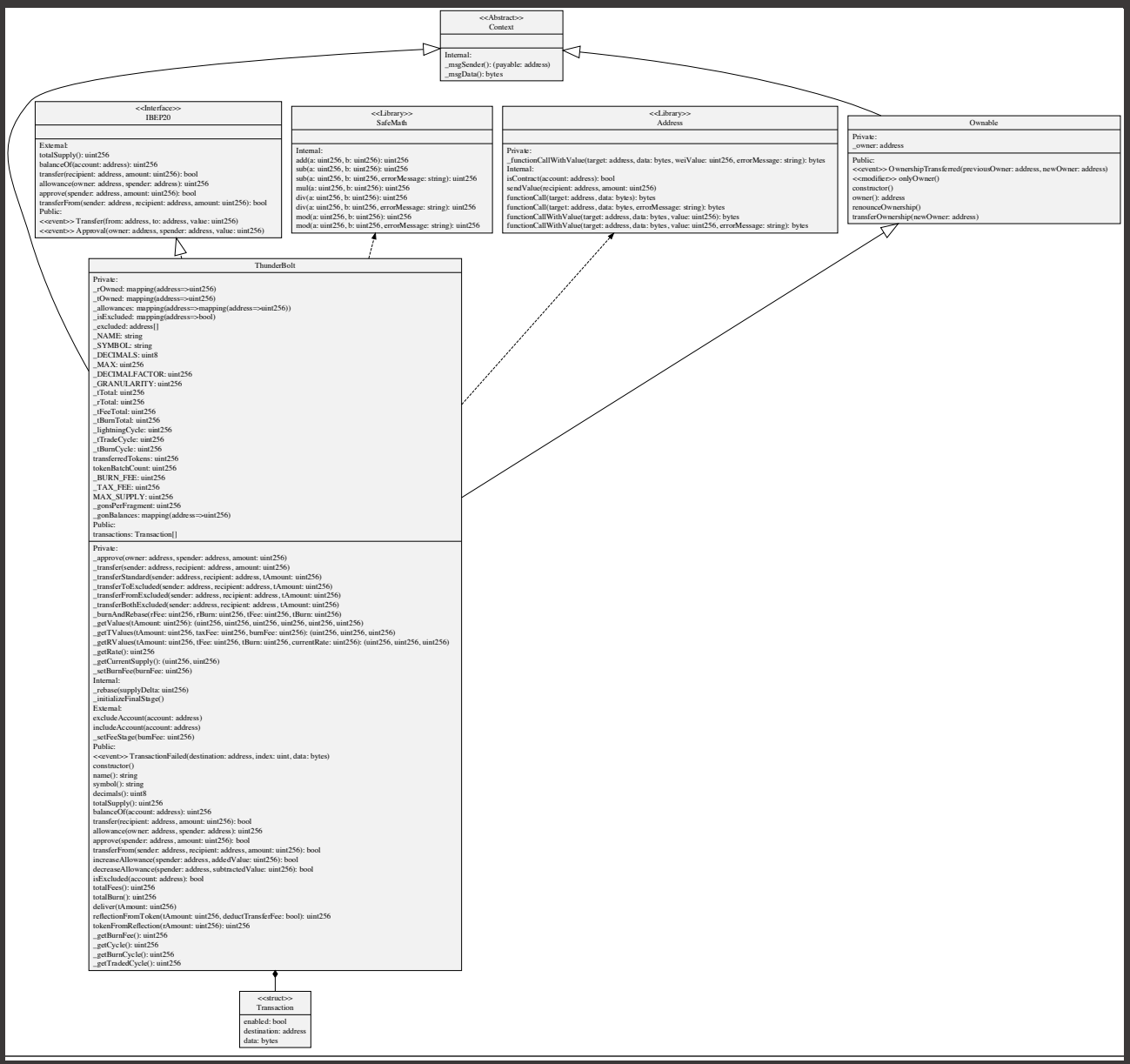
Vulnerability Summary

Informational severity Issues	5
Low severity issues	2
Medium severity issues	1
High severity issues	0



UML

ThunderBolt.sol



BEP-20's Conformance

This test checks for BEP-20's conformance.

- All the functions are present
- All the events are present
- Functions return the correct type
- Functions that must be view are view
- Events' parameters are correctly indexed
- The functions emit the events
- Derived contracts do not break the conformance

Function	present	type	Correct Return value	events	
totalSupply	✅	✅ view	✅		
balanceOf(address)	✅	✅ view	✅		
transfer(address,uint256)	✅	✅ external	✅	✅ Transfer	
transferFrom(address, address, uint256)	✅	✅ external	✅	✅ Transfer	
approve(address,uint256)	✅	✅ external	✅	✅ Approval	
allowance(address, address)	✅	✅ view	✅		
name	✅	✅ view	✅		
symbol	✅	✅ view	✅		

Check Events:

✅ Transfer

✅ Approve

Findings

ThunderBolt.sol

Issue #1:

Type	Severity	Location
Optimization - Unused Variables	● Informational	ThunderBolt.sol

Description:

The state variables **transferredTokens**, **tokenBatchCount**, **MAX_SUPPLY**, **_gonsPerFragment**, and **_gonBalances**, are never used in the contract.

Recommendation:

Remove unused variables to save on storage.

Issue #2:

Type	Severity	Location
Coding Style - Magic numbers	● Informational	ThunderBolt.sol L654-L843, L844, L813, L808, L757, L758

Description:

There are many magic numbers in the code.

Recommendation:

Magic numbers should be refactored with constant variables. It improves the readability of the code, and easier to maintain.

Issue #3:

Type	Severity	Location
Missing events	● Low	ThunderBolt.sol

Description:

Function that affect the status of sensitive variables should emit event. The function _setFeeStage sets the _BURN_FEE variable which determines the amount of % that is burned in every transaction.

Recommendation:

Our recommendation is to add events in critical parts of the contract, such as when the burn rate is changed, when fee is collected, and the amount of tokens that were burned in a transaction. Events are great for integrating with DApps in the future. We recommend considering emitting events when state is changed.

```
function _setFeeStage(uint256 burnFee) external onlyOwner() {
      require(burnFee >= 0 && burnFee <= 1500, 'burnFee should be in 0 - 15');
      _BURN_FEE = burnFee;
      emit SetBurnFee(_BURN_FEE)
   }
```

Issue #4:

Type	Severity	Location
Missing events	● Low	ThunderBolt.sol

Description:

Lack of events in the contract.

Events are important for the integration with blockchain explorers which are commonly used by non-technical investors(for example BSCSCAN). However, when a burn occurs, the amount of tokens the investor actually gets differs from the amount of tokens that is shown in the explorers.

Recommendation:

Our recommendation is to emit appropriate Transfer events when burning tokens and collecting fees.

For example:

```
    function _transferStandard(address sender, address recipient, uint256 tAmount) private {
        uint256 currentRate =  _getRate();
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256
tFee, uint256 tBurn) = _getValues(tAmount);
        uint256 rBurn =  tBurn.mul(currentRate);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _burnAndRebase(rFee, rBurn, tFee, tBurn);
        emit Transfer(sender, address(0), tBurn);
        emit Fee(sender, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }
```

This fix should be applied to all _transfer* functions.

Issue #5:

Type	Severity	Location
Gas Optimization	● Informational	ThunderBolt.sol L804,L809

Description:

uint256 variables always greater than zero. burnFee type is uint256, burnFee >=0 will be always True.

Example:

```
        require(burnFee >= 0 && burnFee <= 1500, 'burnFee should be in 0 - 15');
```

Recommendation:

Fix the incorrect comparison by changing the value type or the comparison.

possible fix:

```
        require(burnFee <= 1500, 'burnFee should be in 0 - 15');
```

Issue #6:

Type	Severity	Location
Best Practice	● Informational	ThunderBolt.sol

Description:

Transfer function uses literals with too many digits. Literals with too many digits are difficult to read and review.

Recommendation:

Our recommendation is instead of having the number 1000000 consider replacing it to: 10e6 for readability.
For example:

```
} else if(_tTradeCycle >= (1*(10e6) * _DECIMALFACTOR) && _tTradeCycle <= (2000000 *
_DECIMALFACTOR)){
                _setBurnFee(550);
```

Issue #7:

Type	Severity	Location
Logic Error	● Medium	ThunderBolt.sol

Description:

The function name doesn't do what it's supposed to. The function name is _setFeeStage, but it sets the _BURN_FEE variable which controls the amount of % that is burned in every transaction.

Recommendation:

Our recommendation is to remove this function. This function gives the owner of the contract the ability to control the amount of % that is burned in every transaction.

```
    function _setFeeStage(uint256 burnFee) external onlyOwner() {
        require(burnFee >= 0 && burnFee <= 1500, 'burnFee should be in 0 - 15');
        _BURN_FEE = burnFee;
    }
```

Issue #8:

Type	Severity	Location
Coding Style - Dead Code	● Informational	ThunderBolt.sol

Description:

_TAX_FEE variable is set to zero. Making all the lines of code, which are related to this variable dead-code (code which can never be executed at run-time)

Recommendation:

We recommend removing those "dead code" in the contract. The execution of dead code wastes computation time and memory. Remove all the lines related to "tFee", "_TAX_FEE", "rFee", and "_tFeeTotal".

Summary

Informational severity Issues	5
Low severity issues	2
Medium severity issues	1
High severity issues	0