

**Smart Contract**

# **Security Assessment**

**For TigerEyeToken  
16 Oct 2022**



**Ascendant**

**Ascendant**

@ascendantproj  
www.ascendant.finance



**Ascendant**

# Table of Contents

**3** Disclaimer

---

**4** Executive Summary

---

**5** Overview

---

**6** Findings Summary & Legend

---

**8** Manual Review

- Issue Checking Status
- Audit Findings
- Functional Test Status
- Omitted Results

---

**22** Automated Review

- Unified Model Language

---

**24** Conclusion

---

# DISCLAIMER

Ascendant Finance (“Ascendant”) has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided ‘as is’, and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team. Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

# Executive Summary

Severity	Found
● High	0
● Medium	3
● Low	23
● Informational	34
Total	60

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

# Overview

This report has been prepared for Tiger Eye Token on the Ethereum network. Ascendant provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

## Summary





<b>Project Name</b>	TigerEyeToken
<b>Platform</b>	Ethereum
<b>Language</b>	Solidity

## Contracts Assessed





Name	Location
Tigereytoken.sol	0xd9F23956711feFA1B7c08FFdE49Fb02da5AFFC4F
SafeMath.sol	In TigerEyeToken contract
Context.sol	In TigerEyeToken contract
SafeMathInt.sol	In TigerEyeToken contract
SafeMathUInt.sol	In TigerEyeToken contract
IERC20.sol	In TigerEyeToken contract

Name	Location
IERC20Metadata.sol	In TigerEyeToken contract
ERC20.sol	In TigerEyeToken contract
Ownable.sol	In TigerEyeToken contract
IterableMapping.sol	In TigerEyeToken contract
DividendPayingTokenInterface.sol	In TigerEyeToken contract
DividendPayingTokenOptionalInterface.sol	In TigerEyeToken contract
DividendPayingToken.sol	In TigerEyeToken contract
IUniswapV2Factory.sol	In TigerEyeToken contract
IUniswapV2Pair.sol	In TigerEyeToken contract
IUniswapV2Router01.sol	In TigerEyeToken contract
IUniswapV2Router02.sol	In TigerEyeToken contract
ERC20DividendTracker.sol	In TigerEyeToken contract

# Findings Summary

Severity	Found
 High	0
 Medium	3
 Low	23
 Informational	34
Total	60

## Classification of Issues

 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

# Manual Review



Ascendant



# Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

# Audit Findings

Severity	Medium
Contract	Tigereytoken.sol
Description	Checks-Effects-Interactions
Code Snippet	<pre>function  _withdrawDividendOfUser(address  payable user)  internal  returns (uint256)  {  ....  _withdrawableDividend);  bool success =  IERC20(USDT).transfer(user,  _withdrawableDividend);   if (!success) {  withdrawnDividends[user] =  withdrawnDividends[user].sub(  _withdrawableDividend  ... </pre>
Recommendation	Function does not follow checks-effects-interactions flow, which requires state updates to come before transfers. This makes the function vulnerable to reentrancy attack, but can be easily mitigated with a Reentrancy Guard.
Status	

# Audit Findings

Severity	Medium
Contract	Tigereytoken.sol
Description	Multiplication performed after division
Code Snippet	<pre>Tigereytoken.swapAndSendToFee(uint256) {     -MarketingBnb =     transferredBalance.div(liquidityFee).mul(marketingFee)</pre>
Recommendation	Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. Multiply first and then divide.
Status	

# Audit Findings

Severity	Medium
Contract	Tigereytoken.sol
Description	Multiplication performed after division
Code Snippet	<pre>Tigereytoken.swapAndSendToFee(uint256){     -DevBnb =     transferredBalance.div(liquidityFee).mul(devFee)</pre>
Recommendation	Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. Multiply first and then divide.
Status	

# Audit Findings

Severity	Lowx3
Contract	Tigereytoken.sol
Description	Functions lack zero-check.
Code Snippet	<pre>Tigereytoken.setMarketingWallet( address)     _marketingWalletAddress = address(wallet)  Tigereytoken.setDevWallet(addre ss).wallet     _devWalletAddress = address(wallet)  Tigereytoken.transferToAddressE TH(address,uint256).recipient recipient.transfer(amount) (Tigereytoken.sol#1924)</pre>
Recommendation	Require statements should be added to the above functions to prevent being set to the zero address.
Status	

# Audit Findings

Severity	Low
Contract	Tigereytoken.sol
Description	Outdated solidity version used
Code Snippet	N/A
Recommendation	Current version used is 0.6.2. Use at least version 0.8.6 or higher.
Status	

# Functional Test Status

Function Name	Type/Return Type	Score
ERC20		
allowance	public	PASS
approve	public	PASS
balanceOf	public	PASS
totalSupply	public	PASS
transfer	public	PASS
transferFrom	public	PASS
increaseAllowance	public	PASS
decreaseAllowance	public	PASS
name	public	PASS
symbol	public	PASS
decimals	public	PASS
_approve	internal	PASS
_beforeTokenTransfer	internal	PASS
_burn	internal	PASS
_mint	internal	PASS
TigerEyetoken		
_setAutomatedMarketMakerPair	private	PASS
_transfer	interal	PASS



Function Name	Type/Return Type	Score
claim	internal	PASS
claimAddress	external	PASS
dividentTokenBalanceOf	read/public	PASS
excludeFromDividends	write/external	PASS
excludeFromFees	write/public	PASS
excludeMultipleAccountsFromFees	write/public	PASS
getAccountDividendsInfo	read/external	PASS
getAccountDividendsInfoAtIndex	read/external	PASS
getClaimWait	read/external	PASS
getLastProcessedIndex()	read/external	PASS
getNumberOfDividendTokenHolders	read/external	PASS
getTotalDividendsDistributed	read/external	PASS
isExcludedFromFees	read/public	PASS
processDividendTracker	write/external	PASS
receive	external	PASS
setAutomatedMarketMakerPair	write/public	PASS
setDevFee	write/external	PASS
setDevWallet	write/external	PASS

Function Name	Type/Return Type	Score
setLastProcessedIndex	write/external	PASS
setLiquiditFee	write/external	PASS
setMarketingFee	write/external	PASS
setMarketingWallet	write/external	PASS
setSwapTokensAtAmount	write/external	PASS
setUSDTRewardsFee	write/external	PASS
swapAndLiquify	private	PASS
swapAndSendDividends	private	PASS
swapAndSendToFee	private	PASS
swapTokensForEth	private	PASS
swapTokensForUsdt	private	PASS
transferToAddressETH	write/public	PASS
updateClaimWait	write/external	PASS
updateDividendTracker	write/public	PASS
updateGasForProcessing	write/public	PASS
updateUniswapV2Router	write/public	PASS
withdrawableDividendOf	write/public	PASS

Context		
<b>_msgData()</b>	internal	PASS
<b>_msgSender()</b>	internal	PASS
Ownable		
<b>owner</b>	read/public	PASS
<b>renounceOwnership</b>	write/public	PASS
<b>transferOwnership</b>	write/public	PASS
ERC20DividendTracker		
<b>_setBalance</b>	internal	PASS
<b>_withdrawDividendOfUser</b>	internal	PASS
<b>accumulativeDividendOf</b>	read/public	PASS
<b>distributeUSDTDividends</b>	write/public	PASS
<b>dividendOf</b>	read/public	PASS
<b>withdrawDividend</b>	write/public	PASS
<b>withdrawableDividendOf</b>	read/public	PASS
<b>withdrawnDividendOf</b>	read/public	PASS
ERC20DividendTracker		
<b>canAutoClaim</b>	private	PASS
<b>excludeFromDividends</b>	write/external	PASS
<b>getAccount</b>	read/public	PASS
<b>getAccountAtIndex</b>	read/public	PASS
<b>getLastProcessedIndex</b>	read/external	PASS

getNumberOfTokenHolders	read/external	PASS
process	write/public	PASS
processAccount	write/public	PASS
setBalance	write/external	PASS
setLastProcessedIndex	write/external	PASS
updateClaimWait	write/external	PASS
withdrawDividend	write/public	PASS

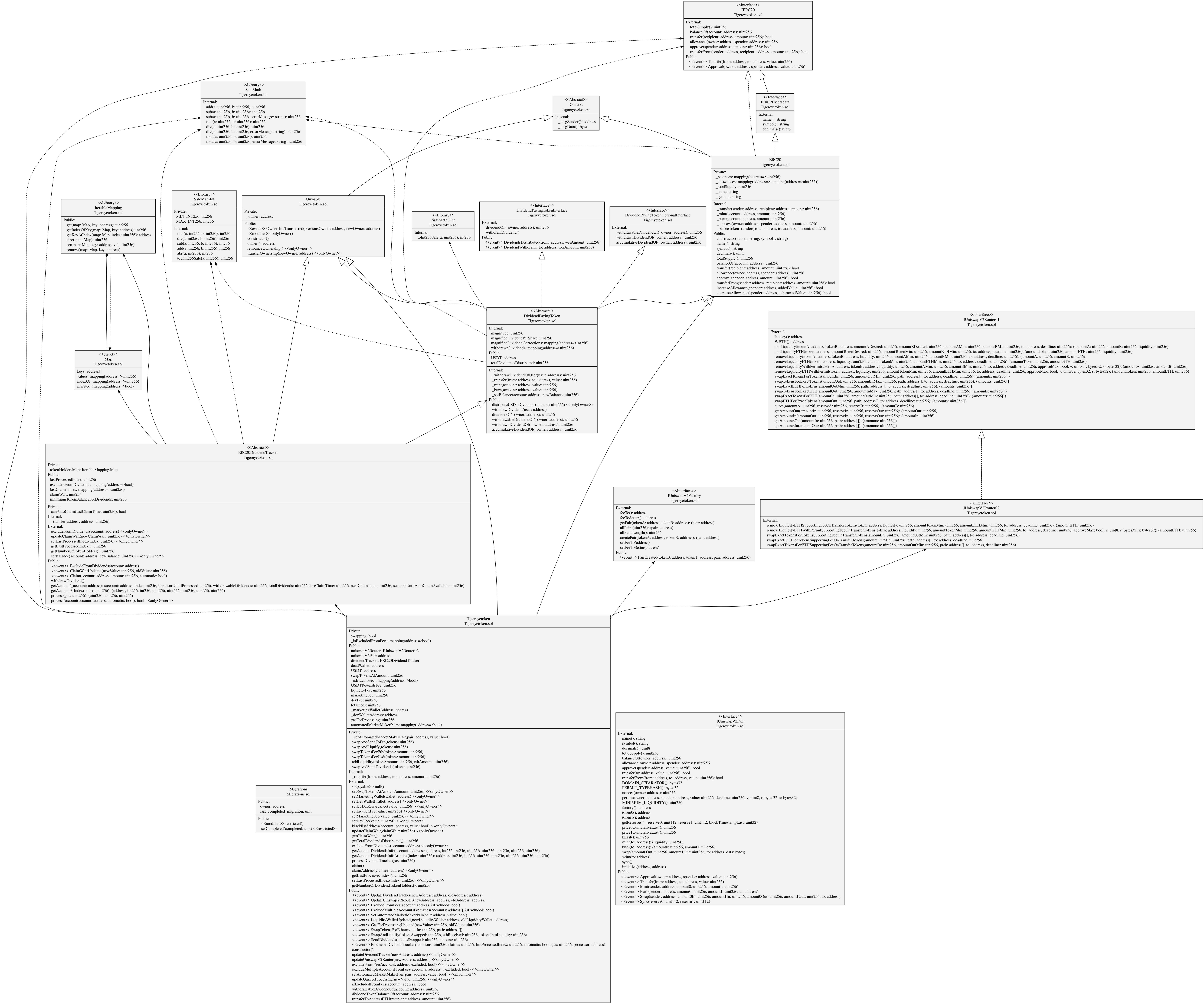
# Omitted Results

**Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.**

# Automated Review



Ascendant



# Conclusion

**The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.**

*Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.*





Ascendant

**Ascendant**

@ascendantproj

[www.ascendant.finance](http://www.ascendant.finance)



Ascendant