FUSION TECH

TYRION TOKEN

Security Assessment

https://tyrion.io/

TABLE OF CONTENTS

Summary

This report has been prepared to discover issues and vulnerabilities in the source code of the project as well as any contract dependencies that were not part of an officially recognized library.A comprehensive examination has been performed, utilizing Static Analysisand Manual Reviewtechniques. The auditingprocess pays specialattention to the following considerations:

Testing the smart contracts against both commonand uncommon attackvectors. Assessing the codebase to ensure compliancewith current best practices and industry standards.Ensuring contract logic meets the specifications and intentions of the client.
Cross referencing contract structure and implementation againstsimilar smart contracts produced by industryleaders.
Thorough line-by-line manual review of the entirecodebase by industryexperts.

The security assessment resulted in findings that ranged from critical to informational. We recommendaddressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could betterserve the projectfrom the securityperspective:

Enhance generalcoding practices for better structures of source codes;Add enough unit tests to cover the possible use cases;
Provide more comments per each function for readability, especiallycontracts that are verified in public;
Provide more transparency on privileged activities once the protocolis live.

Project Summary

Project Name	TYRION - (http://tyrion.io/)
Platform	BINANCE SMART CHAIN
Language	Solidity
Codebase	https://bscscan.com/address/0x7Ee43f72b5431082993AE81356472AfbB42F9dAc
Commit	0xF57jG68jVft5jkwb6544JJG79ga4epi8903AAT4360xc

Audit Summary

Delivery Date	MAY 1, 2023
Audit Methodology	Static Analysis, Manual Review
Key Components	CoinToken

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	2	0	2
Medium	1	0	0	2	0	1
Minor	0	0	0	3	0	0
Informational	3	0	0	6	0	3
Discussion	0	0	0	0	0	0

SWAM SOURCE

ID	File	SHA256 Checksum
CKP	contract.sol	0x60806040526004361061004e5760003560e01c80633659cfe

Overview

External Dependencies

Thecontract serves as the underlying entity to interactwith third-party protocols (token- wapping). The scopeof the audit treats third-party entities as blackboxes and assumestheir functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolenassets.

Privileged Functions

The contract contains the followingprivileged functions that are restricted by role with the modifier. They are used to modify the contractconfigurations and addressattributes. We grouped these functions below.

- excludeFromReward() / includeInReward()
- excludeFromFee() / includeInFee()
- setTaxFeePercent()
- setDevAddress()
- setLiquidityFeePercent()
- setMaxAmount()
- setDevWalletAddress()
- setSwapAndLiquifyEnabled()
- setRouterAddress()
- setNumTokensSellToAddToLiquidity()

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the Timelock contract.

01 | Centralization Risk in Function

Description

The addLiquidity()_hasLiqBeenAdded() function calls the uniswapV2Router.addLiquidityETH function with the to() address specified as owner() for acquiring the generated LP tokens from the corresponding pool. As a result, over time the _owner address will accumulate a significant portionof LP tokens. If _owner the is an EOA (Externally Owned Account),mishandling of its private key can have devastating consequences to the projectas a whole.

Recommendation

We advise to() the address of the uniswapV2Router.addLiquidityBSC() function call to be replaced by the contract() itself, i.e. address(this) , and to restrict the managementof the LP tokens within the scope of the contract's businesslogic. This will also protectthe LP tokens from being stolen if the _owner() account is compromised. In general, we strongly recommend centralized privileges or roles in the protocolto be improved via a decentralized mechanism or via smart-contract based accounts with enhanced securitypractices, f.e.Multisignature wallets().

Indicatively, here are some feasible solutionsthat would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to preventsingle point of failure due to the private key;
- Introduction of a DAO / governance / voting moduleto increase transparency and user involvement

02 | Centralization Risk in Contract

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/contract.sol (98ba012): 603, 640, 644, 648, 652, 656, 660, 665, 906, 912, 612, 636	Acknowledged

Description

In the contract CoinTokens(), the role _owner() has the authority over the following function:

- excludeFromReward() / includeInReward() ; the owner of the contractcan exclude/include an account from/inrewards.
- excludeFromFee() / includeInFee() : the owner of the contractcan exclude/include an account from/in fee.
- setTaxFeePercent() : the owner of the contractcan set the percentage of the tax fee.
- setDevFeePercent() : the owner of the contract can set the percentageof the dev fee.
- setLiquidityFeePercent() : the owner of the contract can set the percentageof liquidity fee.
- setMaxTxPercent() : the owner of the contract can set the maximum transaction amount.
- setDevWalletAddress() : the owner of the contract can update the arbitrary address.
- setRouterAddress() : the owner of the contractcan set any arbitrary addressas the router address.
- setNumTokensSellToAddToLiquidity() : the owner of the contract can set the thresholdto trigger liquidity-adding process.

Any compromise to the _owner() account may allow the hacker to take advantageof this and modify the significant state of the contract,thus introducing centralization risk.

03 | Contract Gains Non-withdrawable BSC via the __owner Function

Function

Category	Severity	Location	Status
Logical Issue	Medium	projects/contract.sol (98ba012): 817	Acknowledged

Description

The swapAndLiquify() function converts half of the contractTokenBalance() tokens to BSC. The other half of the tokens and partof the converted BSC are deposited into the corresponding pool on Uniswap as liquidity. For every swap&liquify() function call, a small amountof BSC leftover in the contract. This is due to the price of drops after swapping the first half of tokens into BSCs, and the other half of tokens require less than the converted BSC to be paired with it when adding liquidity. The contract doesn't appear to provide a way to withdraw those BSC, and they will be locked in the contract forever.

Recommendation

It'snot ideal that more and more BSCare lockedinto the contract over time. The simplest solution is to add a withdraw() function in the contract to withdrawBSC. Other approaches that benefit the token holders add a can be:

- Distribute BSC to token holders proportional to the amount of token they hold.
- Use leftover BSC to buy back tokens from the market to increase the token price.

04 | Regaining Ownership After Renouncing the Contract Ownership

Category	Severity	Location	Status
Logical Issue	Medium	projects/contract.sol (98ba012): 243	Acknowledged

Description

Generally, renouncing the ownership should leave the contract without an owner, thereby removing any functionality that is only available owner to the owner. However, the owner of the cointoken is possible to gain ownership of the contract again even if the ownerhas called the function renounceOwnership() is possibleto gain to renounce() the ownership. This can be achieved by performing the following operations:

- Call lock() Call to lock the contract.The variable _previousownwer() to unlockthe contract.
- Call unlock() to unlock the contract
- would be set to the current owner.
- Call renounce() to renounce the contract ownership. to regain ownership.
- Call unlock() to gain the ownership

Recommendation

We advise the client to review the logic and ensure if it is the intended design. If timelock functionality should be introduced, we recommend usingthe implementation of Compound financeas reference.

05 | Initial Token Distribution

Category	Severity	Location	Status
Logical Issue	Minor	projects/contract.sol (98ba012): 497	Acknowledged

Description

All of the tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute those tokens withoutobtaining the consensusof the community.

Recommendation

We recommend the team to be transparent regarding the initialtoken distribution process.

06 | Lack of Return Value Handling

Category	Severity	Location	Status
Volatile Code	Minor	projects/contract.sol (98ba012): 843	Acknowledged

Description

The return valuesof function addLiquidityBSC() are properly handled.

Recommendation

We recommend using variables to receive the return value of the functions mentionedabove and handleboth success and failure cases if neededby the business logic.

07 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	projects/contract.sol (98ba012): 832~838, 843~850	Acknowledged

Description

A sandwich attack might happen when an attackerobserves a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (beforBSCe transaction being attacked) a transaction to purchase one of the assets and make profitsby backrunning (afterthe transaction beingattacked) a transaction to sell the asset.

The following functionsare called withoutsetting restrictions on slippage or minimum outputamount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

Recommendation

We recommend settingreasonable minimum output amounts, insteadof 0, based on token prices when calling the aforementioned functions.

08 | Lack of Error Message

Category	Severity	Location	Status
Coding Style	Informational	projects/contract.sol (98ba012): 560	Acknowledged

Description

The require statement can be used to check for conditions and throw an exception if the condition is not met.It is better to providea string messagecontaining details about the errorthat will be passed back to the caller.

Recommendation

We advise refactoring the linked codes as below:

```
560          _approve(_msgSender(), spender, _allowances[_msgSender()]
[spender].add(addedValue), "increase allowance overflow");
```

09 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	projects/contract.sol (98ba012): 862	Acknowledged

Description

The condition! _isExcluded[sender] & !_isExcluded[recipient]
can be included in else .

Recommendation

The following code can be removed:

```
861  ... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
862      _transferStandard(sender, recipient, amount);
863  } ...
```

10 | Typos In The Contract

Category	Severity	Location	Status
Coding Style	Informational	projects/contract.sol (98ba012): 470, 670	Acknowledged

Description

There are several typos in the code and comments.

1. In the following code snippet, tokensIntoLiquidity() should be tokensIntoLiquidity()

```
event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

2. recieve() should be recieve() _swapping() should be _swapping() in the line of comment //to _recieve BSC from uniswapV2Router when swaping() .

Recommendation

We recommend correcting all typos in the contract.

11 | Function and VariableNaming Doesn't Match the OperatingEnvironment

Category	Severity	Location	Status
Coding Style	Informational	projects/contract.sol (98ba012): 1	Acknowledged

Description

There are multiplenaming issues inside the current contract, which can be misleading to use uniswap() and BSC() instead of Uniswap() and BSC() if the project landingon BSC.

For example, the cointoken() contract uses Uniswap() for swapping and adding liquidity to the Uniswap pool but names it uniswap()

Recommendation

Change "Uniswap" and "BSC" to "Uniswap" and "BSC" in the contract respectively to match the operating environment and avoid confusion.

12 | Potential Resource Exhaustion

Category	Severity	Location	Status
Logical Issue	Informational	projects/contract.sol (98ba012): 614, 709	Acknowledged

Description

The farloop() within functions includeInReward(address) and _getCurrentSupply() takes the variable _excluded.length(), as the maximal iterationtimes. If the size of the array is very large, it could exceed the gas limit to execute the functions. In this case, the contract might suffer from DoS (Denial of Service) situation.

Recommendation

We recommendthe team reviewthe design and ensure this would not cause loss to the project.

13 | Inconsistency Between Comment and Code

Category	Severity	Location	Status
Inconsistency	Informational	projects/contract.sol (98ba012): 230~236	Acknowledged

Description

According to the commentin L238, the lock() function will lock the contract **for a given time period**. However, the code implementation will lock the contract **until the given timestamp.**

```
238     //Unlocks the contract for owner when _lockTime is exceeds
239     function unlock() public virtual {
240         require(_previousOwner == msg.sender, "You don't have permission to
unlock.");
241         require(block.timestamp > _lockTime , "Contract is locked.");
242         emit OwnershipTransferred(_owner, _previousOwner);
243         _owner = _previousOwner;
244     }
```

Recommendation

We recommend the team reviewthe design and update either comments or code implementation to ensure consistent logic between code and comment.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanismto relocate funds.

Logical Issue

Logical Issue findingsdetail a faultin the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findingsrefer to segmentsof code that behave unexpectedly on certain edge cases that may resultin a vulnerability.

Coding Style

Coding Style findingsusually do not affect the generated byte-code but rather commenton how to make the codebase more legible and, as a result, easilymaintainable.

Inconsistency

Inconsistency findings referto functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing differentrequire statements on the input variables than a setterfunction.

Checksum Calculation MBSCod

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specifiedcommit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" commandagainst the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimerand limitation of liability) set forth in the ServicesAgreement, or the scope of services, and terms and conditions providedto you ("Customer" or the "Company") in connection with the Agreement. This report providedin connection with the Servicesset forth in the Agreementshall be used by the Company only to the extent permittedunder the terms and conditions set forth in the Agreement.
This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes,nor may copiesbe delivered to any otherperson other than the Company,without FusionTech prior written consentin each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts FusionTech to perform a security assessment. This report 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, businessmodel 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 in no way provides investment advice, nor shouldbe leveraged as investment advice of any sort. This report represents an extensive assessing 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. FusionTech's position is that each companyand individual are responsible for their own due diligenceand continuous security.
FusionTech's goal is to help reduce the attack vectorsand 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 FusionTech is subject to dependencies and under continuing development. You agree that your access and/or use, includingbut 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 tokensare 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.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTSOR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUTWARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, FusionTech HEREBY DISCLAIMS ALL WARRANTIES, WHBSCER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, FusionTech SPECIFICALLY DISCLAIMS ALL IMPLIEDWARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, FusionTech MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE,ACCURATE, COMPLETE, FREE OF HARMFULCODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, FusionTech PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDSOR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHERFusionTech NOR ANY OF FusionTech'SAGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. FusionTech WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTINGFROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES,ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED"AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THBSCIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSENOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIESBE DELIVERED TO, ANY OTHERPERSON WITHOUT FusionTech'SPRIOR WRITTEN CONSENTIN EACH INSTANCE.

NOTHIRD PARTY OR ANYONE ACTINGON BEHALF OF ANY THEREOF,SHALL BE A THIRD PARTYOR OTHER BENEFICIARY OF SUCH SERVICES,ASSESSMENT REPORT, AND ANY ACCOMPANYING

FORAVOIDANCE OF DOUBT, THE SERVICES,INCLUDING ANY ASSOCIATEDASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2022 by leading academics in the field of Computer Science, FusionTech is going to be a leading blockchain security company that serves to verify the security and correctness of smart contracts KYC and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of ourclients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

FUSION TECH

https://fusiontech.live

https://twitter.com/fusiontechh

https://t.me/fusiontechofficial

https://github.com/fusiontechofficial

Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Tyrion Token
Website: https://tyrion.io
Platform: Binance Smart Chain
Language: Solidity
Date: April 6th, 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 Tyrion Token team to perform the Security audit of the Tyrion Token smart contract 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 April 6th, 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 TYON token smart contract is an ecosystem token in BSC blockchain networks.
- The token follows the ERC20 standard, which will make it compatible with all the platforms that support the ERC20 standard.
- This contract has functionality like enabling and disabling trading fees, calculating ecosystem fees and tax fees, setting tax percentage values, setting LP addresses, setting ecosystem fee percentages, and setting the buy and sell amount.
- The Tyrion token smart contract inherits AccessControlUpgradeable, SafeERC20Upgradeable, PausableUpgradeable, OwnableUpgradeable and Initializable standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope.

Audit scope

Name	Code Review and Security Analysis Report for Tyrion Token Smart Contract
Platform	**BSC / Solidity**
File	TYON.sol
File MD5 Hash	08F32F0C3B15EC5D0F58FB5314B35D57
Updated File MD5 Hash	FBF4B1DBB3BF957F95CF9C9C67276F9D
Audit Date	April 5th, 2023
Revised Audit Date	April 8th, 2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation

Tokenomics:
• Name: TYON
• symbol: TYON
• Decimals: 9
• Total Supply: 500 Million
  ○ 40% for GrowthX wallet
  ○ 7% for TyrionShield Wallet
	YES, This is valid.

Other Specification: • Open Zeppelin standard code is used. • initializer modifier is used to prevent initializing a token twice. • Minting _totalSupply values into owner and _growthXWallet account.	**YES, This is valid.**

Tax Cut Per Wallet Specification:
• Buy / Sell Taxing: 2.5%
  ○ Ecosystem Fee: 1%
  ■ 0.25% of GrowthX tax
  ■ 0.25% of TyrionShield tax
  ■ 0.25% of FundMe tax
  ■ 0.25% of Ecosystem tax
  ○ Tax Fee: 1.5% to holders
• Transfer Taxing: 0.5%
  ○ Ecosystem Fee: 0.5%
  ○ Tax Fee: 0
• Maximum Tax Fee: 10%
• Maximum Ecosystem Fee: 10%
• Initial Sale Phase: 1
• Maximum Transaction Amount: 5 Million TYON
• Maximum Transaction Amount percentage: 4% of the
	YES, This is valid. Owner wallet's private key must be handled very securely. Because if that is compromised, then it will create problems.

total supply • Minimum Buy/Sell Amount: 500 TYON	
<u>Owner has control over following functions:</u> • Set the maximum transaction percentage. • Set a minimum Buy and Sell amount. • Set current phases. • Removes an account from the excluded list. • Add a new account to the excluded list. • Current owner can transfer ownership of the contract to a new account. • Deleting ownership will leave the contract without an owner, removing any owner-only functionality. • A role of admin can assign different role addresses. • Withdraw all ETH trapped in the smart contract. • Withdraw ERC20 tokens trapped in the smart contract.	**YES, This is valid. Owner wallet's private key must be handled very securely. Because if that is compromised, then it will create problems.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts do 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, 1 high, 3 medium and 1 low and a very low level issue.
We confirm that 1 high issue and 3 medium issues are fixed/Acknowledged in the revised smart contract code.

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	Moderate
	Features claimed	Moderate
	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**

Code Quality

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

The libraries in the Tyrion Token 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 Tyrion Token .

The Tyrion Token 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 not well commented on in the smart contracts. Ethereum's NatSpec commenting style is used, which is a good thing.

Documentation

We were given a Tyrion Token smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not **well** commented. but 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://www.tyrion.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 contract infrastructure that 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

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__AccessControl_init	internal	access only Initializing	No Issue
3	__AccessControl_init_unchained	internal	access only Initializing	No Issue
4	onlyRole	modifier	Passed	No Issue
5	supportsInterface	read	Passed	No Issue
6	hasRole	read	Passed	No Issue
7	_checkRole	internal	Passed	No Issue
8	_checkRole	internal	Passed	No Issue
9	getRoleAdmin	read	Passed	No Issue
10	grantRole	write	access only Role	No Issue
11	revokeRole	write	access only Role	No Issue
12	renounceRole	write	Passed	No Issue
13	_setupRole	internal	Passed	No Issue
14	_setRoleAdmin	internal	Passed	No Issue
15	_grantRole	internal	Passed	No Issue
16	_revokeRole	internal	Passed	No Issue
17	__Ownable_init	internal	access only Initializing	No Issue
18	__Ownable_init_unchained	internal	access only Initializing	No Issue
19	onlyOwner	modifier	Passed	No Issue
20	owner	read	Passed	No Issue
21	_checkOwner	internal	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue
24	_transferOwnership	internal	Passed	No Issue
25	__Pausable_init	internal	access only Initializing	No Issue
26	__Pausable_init_unchained	internal	access only Initializing	No Issue
27	whenNotPaused	modifier	Passed	No Issue
28	whenPaused	modifier	Passed	No Issue
29	paused	read	Passed	No Issue
30	_requireNotPaused	internal	Passed	No Issue
31	_requirePaused	internal	Passed	No Issue
32	_pause	internal	Passed	No Issue
33	_unpause	internal	Passed	No Issue
34	constructor	write	Passed	No Issue
35	initialize	write	Passed	No Issue
36	__TYON_V1_init_unchained	internal	access only Initializing	No Issue

37	receive	external	Passed	No Issue
38	setTaxFeePercent	external	access only Role	No Issue
39	setEcosystemFeePercent	external	access only Role	No Issue
40	setMaxTxPercent	external	access only Owner	No Issue
41	setMinBuySellAmount	external	access only Owner	No Issue
42	setCurrentPhase	external	access only Owner	No Issue
43	setBadge	external	access only Role	No Issue
44	withdrawToken	external	Owner can drain contract's coin and tokens	Refer to audit findings
45	withdraw	external	Owner can drain contract's coin and tokens	Refer to audit findings
46	pause	external	access only Owner	No Issue
47	unpause	external	access only Owner	No Issue
48	deliver	external	Passed	No Issue
49	name	external	Passed	No Issue
50	symbol	external	Passed	No Issue
51	decimals	external	Passed	No Issue
52	salePhase	external	Passed	No Issue
53	totalSupply	external	Passed	No Issue
54	totalFees	external	Passed	No Issue
55	balanceOf	external	Passed	No Issue
56	setLPAddress	write	access only Owner	No Issue
57	removeLPAddress	write	access only Owner	No Issue
58	transfer	write	Tax fees validation, Tax fees on transfers	Acknowledged
59	approve	write	Passed	No Issue
60	transferFrom	write	Passed	No Issue
61	increaseAllowance	write	Passed	No Issue
62	decreaseAllowance	write	Passed	No Issue
63	excludeFromReward	write	access only Owner	No Issue
64	includeInReward	write	access only Owner	No Issue
65	excludeFromFee	write	access only Owner	No Issue
66	includeInFee	write	access only Owner	No Issue
67	reflectionFromToken	read	Passed	No Issue
68	tokenFromReflection	read	Passed	No Issue
69	getUserBadge	read	Passed	No Issue
70	isExcludedFromReward	read	Passed	No Issue
71	isExcludedFromFee	read	Passed	No Issue
72	allowance	read	Passed	No Issue
73	_distributeTax	internal	Passed	No Issue
74	removeAllFee	internal	Passed	No Issue
75	enableTradingFee	internal	Passed	No Issue
76	disableTradingFee	internal	Passed	No Issue
77	restoreAllFee	internal	Passed	No Issue
78	_approve	internal	Passed	No Issue
79	_transfer	internal	Passed	No Issue

80	_tokenTransfer	internal	Passed	No Issue
81	_transferStandard	internal	Passed	No Issue
82	_transferBothExcluded	internal	Passed	No Issue
83	_transferToExcluded	internal	Passed	No Issue
84	_transferFromExcluded	internal	Passed	No Issue
85	_balanceOf	internal	Passed	No Issue
86	_reflectFee	internal	Passed	No Issue
87	_getValues	internal	Passed	No Issue
88	_getTValues	internal	Passed	No Issue
89	_getRate	internal	Passed	No Issue
90	_getCurrentSupply	internal	Passed	No Issue
91	calculateTaxFee	internal	Passed	No Issue
92	calculateEcosystemFee	internal	Passed	No Issue
93	_getRValues	internal	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.

High Severity

(1) Initialize function is not working:

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() initializer {}

/**
 * @dev initialize the token contract. Minting _totalSupply into owner and growthX account.
 * setting msg sender as DEFAULT_ADMIN_ROLE, MINTER_ROLE, BURNER_ROLE.
 * Note:initializer modifier is used to prevent initialize token twice.
 */
function initialize(
    address _growthX,
    address _tyrionShield,
    address _fundMe,
    address _ecosystemGrowth,
    address _growthXWallet,
    address _tyrionShieldWallet
) public initializer {
    __Context_init_unchained();
      Ownable_init_unchained();
```

Constructor and initialize both use the initializer modifier. Constructor doesn't have any code written but when the modifier is called the "_initialized" variable is set to 1 which prevents initialization function execution.

Resolution: We advise removing the initializer modifier from the constructor.

Status: This is fixed in the revised smart contract code.

Medium

(1) The value of 30% presale is missing:
The value of 47% of the total supply has been allocated to two wallets. And a 30% presale is required. However, 22% of the total supply is also available for presale.

Resolution: We suggest correcting the supply for presale.

Status: This is fixed in the revised smart contract code.

(2) Tax fees validation:
The maximum limit for tax fees is 50%, individually. So, tax fee + ecosystem fee can be 100%, as they can both be set at 50%. Owners can stop the sales by setting high tax fees.

Resolution: We suggest correcting the limits for fees.

Status: This is fixed in the revised smart contract code.

(3) Tax fees on transfers:
On changing transfer fees, the change does not reflect. Transfer of the token does not use the latest updated _transferEcosystemFee and _transferTaxfee.

Resolution: We suggest correcting the transfer fee logic.

Status: This is acknowledged in the revised smart contract code.

Low

(1) Owner can drain contract's coin and tokens:

```
/**
 * @dev function to withdraw ERC20 tokens trapped on smartcontract.
 * @param amount amount of token reuired to withdraw.
 * @param token address of the ERC20 token smartcontract
    - the caller must be the owner of the contract
 */

function withdrawToken(
    uint256 amount,
    address token
) external virtual onlyOwner {
    IERC20Upgradeable ERC20Token = IERC20Upgradeable(token);
    ERC20Token.safeTransfer(owner(), amount);
}

/**
 * @dev function to withdraw all ETH trapped on smartcontract to
   owner's account.
 */

function withdraw() external virtual onlyOwner {
    (bool sent, ) = owner().call{value: address(this).balance}("");
    require(sent, "Failed to send Ether");
}
```

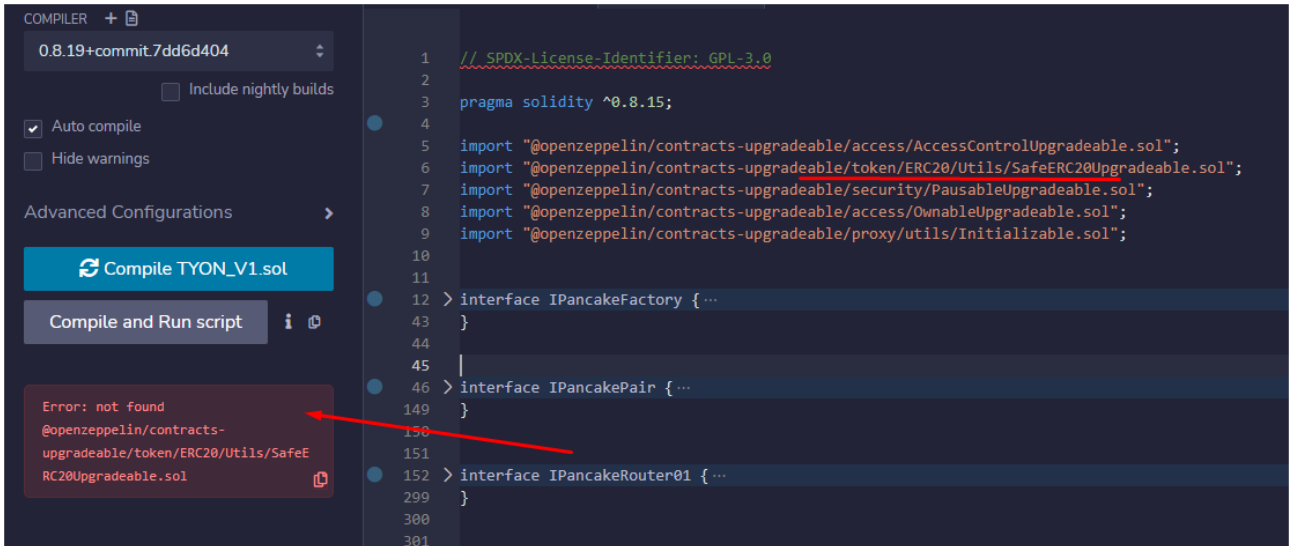
By using "withdraw" and "withdrawToken" functions, the owner can drain all the coins and tokens from the contract.

Resolution: We suggest confirming this feature is required.

Status: We got confirmation from the Tyrion team that this is a required feature, and will be executed based on business logic.

Very Low / Informational / Best practices:

(1) Wrong openzeppelin path:



The openzeppelin path for SafeERC20Upgradeable is wrong.

Resolution: Replace "Utils" with "utils".

Status: This is fixed in the revised smart contract code.

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:

OwnableUpgradeable.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.
- _checkOwner: Check address is current owner address or not.

AccessControlUpgradeable.sol

- grantRole: A role of admin can assign different role addresses.
- revokeRole: A role of admin can remove role address.
- renounceRole: A role of admin can renounce roles for self.

TYON_V1.sol

- setMaxTxPercent: Percentage for maximum transaction amount can be set by the owner.
- setMinBuySellAmount: Minimum Buy and Sell amount can be set by the owner.
- setCurrentPhase: Current phase can be set by the owner.
- withdrawToken:The owner can withdraw ERC20 tokens trapped in the smart contract.
- withdraw:The owner can withdraw all ETH trapped in the smart contract to the owner's account.
- pause: The owner can set trigger pause status true.
- unpause: The owner can set trigger pause status false
- setLPAddress: LP address can be set by the owner.
- removeLPAddress: Lp address can be removed. by the owner.
- excludeFromReward: The owner can add a new account to the excluded list.
- includeInReward: Removes an account from the excluded list by the owner.
- excludeFromFee: The owner can set the excluded account address status is true.

- includeInFee: The owner can set the excluded account address status is false.

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 file. And we have used all possible tests based on given objects. We have observed 1 high severity issue, 3 medium severity issues, 1 low severity issue and 1 Informational issue in the code. We confirm that 1 high issue and 3 medium issues are fixed / acknowledged in the revised smart contract code. So, **it's good to go 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 smart 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).

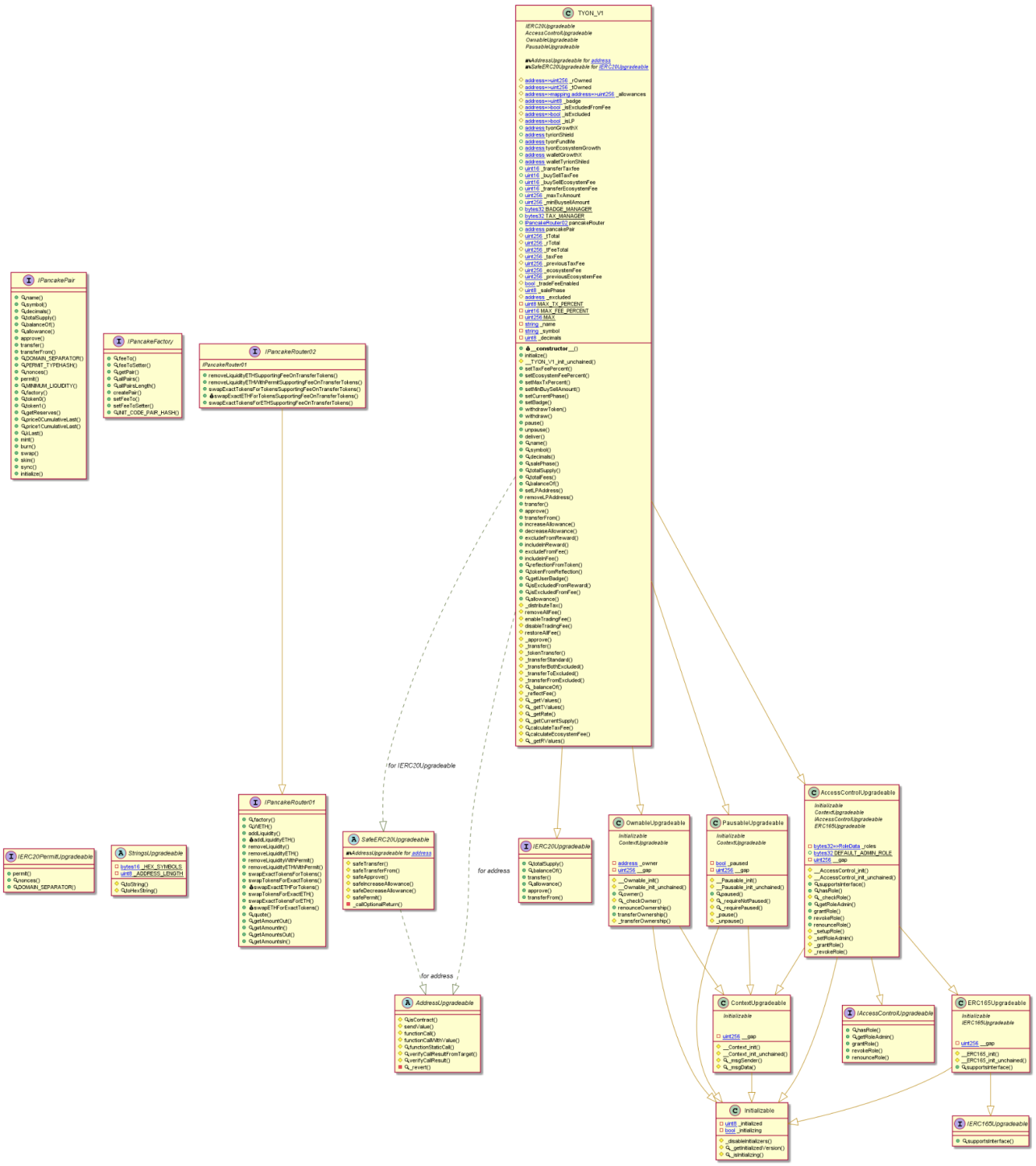
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - Tyrion Token



Slither Results Log

Slither Log >> TYON.sol

```
TYON_V1.allowance(address,address).owner (TYON.sol#1406) shadows:
        - OwnableUpgradeable.owner() (TYON.sol#720-722) (function)
TYON_V1._approve(address,address,uint256).owner (TYON.sol#1479) shadows:
        - OwnableUpgradeable.owner() (TYON.sol#720-722) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

TYON_V1.setMaxTxPercent(uint256) (TYON.sol#1135-1138) should emit an event for:
        - _maxTxAmount = (_tTotal * (maxTxPercent)) / (10 ** 2) (TYON.sol#1137)
TYON_V1.setMinBuySellAmount(uint256) (TYON.sol#1140-1142) should emit an event for:
        - _minBuysellAmount = minToken * 10 ** 9 (TYON.sol#1141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

AddressUpgradeable._revert(bytes,string) (TYON.sol#426-435) uses assembly
        - INLINE ASM (TYON.sol#428-431)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

TYON_V1.excludeFromFee(address) (TYON.sol#1337-1343) compares to a boolean constant:
        -require(bool,string)(_isExcludedFromFee[account] == false,account already excluded) (TYON.sol#1338-1341)
TYON_V1.includeInFee(address) (TYON.sol#1345-1351) compares to a boolean constant:
        -require(bool,string)(_isExcludedFromFee[account] == true,account already included) (TYON.sol#1346-1349)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

TYON_V1.includeInReward(address) (TYON.sol#1324-1335) has costly operations inside a loop:
        - delete _tOwned[account] (TYON.sol#1329)
TYON_V1.includeInReward(address) (TYON.sol#1324-1335) has costly operations inside a loop:
        - delete _isExcluded[account] (TYON.sol#1330)
TYON_V1.includeInReward(address) (TYON.sol#1324-1335) has costly operations inside a loop:
        - _excluded.pop() (TYON.sol#1331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (TYON.sol#366-372) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (TYON.sol#385-387) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (TYON.sol#389-396) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (TYON.sol#347-352) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (TYON.sol#414-424) is never used and should be removed
ContextUpgradeable.__Context_init() (TYON.sol#686-687) is never used and should be removed
ContextUpgradeable._msgData() (TYON.sol#695-697) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (TYON.sol#797-798) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (TYON.sol#800-801) is never used and should be removed
Initializable._disableInitializers() (TYON.sol#669-675) is never used and should be removed
Initializable._getInitializedVersion() (TYON.sol#677-679) is never used and should be removed
Initializable._isInitializing() (TYON.sol#681-683) is never used and should be removed
OwnableUpgradeable.__Ownable_init() (TYON.sol#707-709) is never used and should be removed
PausableUpgradeable.__Pausable_init() (TYON.sol#753-755) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (TYON.sol#494-504) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (TYON.sol#515-526) is never used and should be r
emoved
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (TYON.sol#506-513) is never used and should be r
emoved
SafeERC20Upgradeable.safePermit(IERC20PermitUpgradeable,address,address,uint256,uint256,uint8,bytes32,bytes32) (TYON.sol#528-5
42) is never used and should be removed
SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (TYON.sol#485-492) is never used and should b
e removed
StringsUpgradeable.toHexString(uint256) (TYON.sol#577-588) is never used and should be removed
StringsUpgradeable.toString(uint256) (TYON.sol#557-575) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.13 (TYON.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in AddressUpgradeable.sendValue(address,uint256) (TYON.sol#347-352):
        - (success) = recipient.call{value: amount}() (TYON.sol#350)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (TYON.sol#374-383):
        - (success,returndata) = target.call{value: value}(data) (TYON.sol#381)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (TYON.sol#389-396):
        - (success,returndata) = target.staticcall(data) (TYON.sol#394)
Low level call in TYON_V1.withdraw() (TYON.sol#1180-1183):
        - (sent) = owner().call{value: address(this).balance}() (TYON.sol#1181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IPancakeRouter01.WETH() (TYON.sol#8) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (TYON.sol#235) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (TYON.sol#237) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (TYON.sol#268) is not in mixedCase
Function IPancakeFactory.INIT_CODE_PAIR_HASH() (TYON.sol#337) is not in mixedCase
Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (TYON.sol#472) is not in mixedCase
Function ContextUpgradeable.__Context_init() (TYON.sol#686-687) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (TYON.sol#689-690) is not in mixedCase
Variable ContextUpgradeable.__gap (TYON.sol#699) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (TYON.sol#707-709) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (TYON.sol#711-713) is not in mixedCase
Variable OwnableUpgradeable.__gap (TYON.sol#743) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (TYON.sol#753-755) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (TYON.sol#757-759) is not in mixedCase
Variable PausableUpgradeable.__gap (TYON.sol#793) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (TYON.sol#797-798) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (TYON.sol#800-801) is not in mixedCase
Variable ERC165Upgradeable.__gap (TYON.sol#806) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init() (TYON.sol#810-811) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (TYON.sol#813-814) is not in mixedCase
Variable AccessControlUpgradeable.__gap (TYON.sol#898) is not in mixedCase
Contract TYON_V1 (TYON.sol#900-1712) is not in CapWords
Parameter TYON_V1.initialize(address,address,address,address,address,address)._growthX (TYON.sol#976) is not in mixedCase
Parameter TYON_V1.initialize(address,address,address,address,address,address)._tyrionShield (TYON.sol#977) is not in mixedCase
Parameter TYON_V1.initialize(address,address,address,address,address,address)._fundMe (TYON.sol#978) is not in mixedCase
Parameter TYON_V1.initialize(address,address,address,address,address,address)._ecosystemGrowth (TYON.sol#979) is not in mixedC
```

```
Parameter TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address)._fundMe (TYON.sol#1001) is not in
mixedCase
Parameter TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address)._ecosystemGrowth (TYON.sol#1002) i
s not in mixedCase
Parameter TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address)._growthXWallet (TYON.sol#1003) is
not in mixedCase
Parameter TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address)._tyrionShieldWallet (TYON.sol#1004
) is not in mixedCase
Parameter TYON_V1.getUserBadge(address)._address (TYON.sol#1379) is not in mixedCase
Parameter TYON_V1.calculateTaxFee(uint256)._amount (TYON.sol#1690) is not in mixedCase
Parameter TYON_V1.calculateEcosystemFee(uint256)._amount (TYON.sol#1695) is not in mixedCase
Variable TYON_V1._rOwned (TYON.sol#909) is not in mixedCase
Variable TYON_V1._tOwned (TYON.sol#910) is not in mixedCase
Variable TYON_V1._allowances (TYON.sol#911) is not in mixedCase
Variable TYON_V1._badge (TYON.sol#912) is not in mixedCase
Variable TYON_V1._isExcludedFromFee (TYON.sol#913) is not in mixedCase
Variable TYON_V1._isExcluded (TYON.sol#914) is not in mixedCase
Variable TYON_V1._isLP (TYON.sol#915) is not in mixedCase
Variable TYON_V1._transferTaxfee (TYON.sol#928) is not in mixedCase
Variable TYON_V1._buySellTaxFee (TYON.sol#929) is not in mixedCase
Variable TYON_V1._buySellEcosystemFee (TYON.sol#930) is not in mixedCase
Variable TYON_V1._transferEcosystemFee (TYON.sol#931) is not in mixedCase
Variable TYON_V1._maxTxAmount (TYON.sol#933) is not in mixedCase
Variable TYON_V1._minBuysellAmount (TYON.sol#934) is not in mixedCase
Variable TYON_V1._tTotal (TYON.sol#943) is not in mixedCase
Variable TYON_V1._rTotal (TYON.sol#944) is not in mixedCase
Variable TYON_V1._tFeeTotal (TYON.sol#945) is not in mixedCase
Variable TYON_V1._taxFee (TYON.sol#947) is not in mixedCase
Variable TYON_V1._previousTaxFee (TYON.sol#948) is not in mixedCase
Variable TYON_V1._ecosystemFee (TYON.sol#949) is not in mixedCase
Variable TYON_V1._previousEcosystemFee (TYON.sol#950) is not in mixedCase
Variable TYON_V1._tradeFeeEnabled (TYON.sol#951) is not in mixedCase
Variable TYON_V1._salePhase (TYON.sol#953) is not in mixedCase
Variable TYON_V1._excluded (TYON.sol#954) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable TYON_V1._getValues(uint256).rTransferAmount (TYON.sol#1652) is too similar to TYON_V1._getValues(uint256).tTransferAm
ount (TYON.sol#1649)
Variable TYON_V1.reflectionFromToken(uint256,bool).rTransferAmount (TYON.sol#1362) is too similar to TYON_V1._getValues(uint25
6).tTransferAmount (TYON.sol#1649)
Variable TYON_V1._getValues(uint256).rTransferAmount (TYON.sol#1652) is too similar to TYON_V1._getTValues(uint256).tTransferA
mount (TYON.sol#1666)
Variable TYON_V1._transferFromExcluded(address,address,uint256).rTransferAmount (TYON.sol#1616) is too similar to TYON_V1._tra
nsferToExcluded(address,address,uint256).tTransferAmount (TYON.sol#1597)
Variable TYON_V1.reflectionFromToken(uint256,bool).rTransferAmount (TYON.sol#1362) is too similar to TYON_V1._getTValues(uint2
56).tTransferAmount (TYON.sol#1666)
Variable TYON_V1._transferFromExcluded(address,address,uint256).rTransferAmount (TYON.sol#1616) is too similar to TYON_V1._tra
nsferBothExcluded(address,address,uint256).tTransferAmount (TYON.sol#1575)
Variable TYON_V1._transferFromExcluded(address,address,uint256).rTransferAmount (TYON.sol#1616) is too similar to TYON_V1._get
Values(uint256).tTransferAmount (TYON.sol#1649)
Variable TYON_V1._transferToExcluded(address,address,uint256).rTransferAmount (TYON.sol#1595) is too similar to TYON_V1._trans
ferBothExcluded(address,address,uint256).tTransferAmount (TYON.sol#1575)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address) (TYON.sol#998-1088) uses literals with too m
any digits:
        - _tTotal = 500000000 * 10 ** 9 (TYON.sol#1010)
TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address) (TYON.sol#998-1088) uses literals with too m
any digits:
        - _maxTxAmount = 5000000 * 10 ** 9 (TYON.sol#1032)
TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address) (TYON.sol#998-1088) uses literals with too m
any digits:
        - transfer(_tyrionShieldWallet,35000000 * 10 ** 9) (TYON.sol#1086)
TYON_V1.__TYON_V1_init_unchained(address,address,address,address,address,address) (TYON.sol#998-1088) uses literals with too m
any digits:
        - transfer(_growthXWallet,200000000 * 10 ** 9) (TYON.sol#1087)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TYON_V1.pancakePair (TYON.sol#941) should be constant
TYON_V1.pancakeRouter (TYON.sol#940) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
TYON.sol analyzed (18 contracts with 84 detectors), 156 result(s) found
```

Solidity Static Analysis

TYON.sol

Security

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.

more

Pos: 295:24:

Gas & Economy

Gas costs:

Gas requirement of function TYON_V1.includeInReward 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: 438:4:

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: 440:8:

Email: audit@EtherAuthority.io

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: 792:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 16:4:

Miscellaneous

Constant/View/Pure functions:

TYON_V1.reflectionFromToken(uint256,bool) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 467:4:

Similar variable names:

TYON_V1._getRValues(uint256,uint256,uint256,uint256) : Variables have very similar names "rTaxCut" and "tTaxCut". Note: Modifiers are currently not considered by this static analysis.

Pos: 823:54:

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: 635:12:

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.
more
Pos: 444:16:

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: 811:15:

Solhint Linter

TYON.sol

```
TYON.sol:3:1: Error: Compiler version ^0.8.15 does not satisfy the r
semver requirement
TYON.sol:14:1: Error: Contract has 34 states declarations but allowed
no more than 15
TYON.sol:14:1: Error: Contract name must be in CamelCase
TYON.sol:82:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
TYON.sol:82:31: Error: Code contains empty blocks
TYON.sol:112:5: Error: Function name must be in mixedCase
TYON.sol:207:32: Error: Code contains empty blocks
TYON.sol:285:9: Error: Variable name must be in mixedCase
TYON.sol:295:25: Error: Avoid using low level calls.
```

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