

BLOCKSTARS TECHNOLOGY

Smart Contract Security Audit



PROJECT: TURBO
ANALYSIS TYPE: DEEP
VERSION: 1
START DATE: 04th MAY 2023



REPORT CONTENTS

1. Introduction	3
2. Project Context	3
3. Audit Scope	4
4. Definitions	5
5. Analytic Statistics	6
6. Manual Audit Process	7
7. Audit Findings in Detail	8
9. Conclusion	9
10. Appendix	10

DECLARATION

This is a BLOCKSTARS TECHNOLOGY smart contract security audit created for the Turbo contract.



Results and documentation within this report contain confidential information regarding the clients' contracts.

Analysis results of the smart contract is supplied containing lists of vulnerabilities and malicious code which could be used to malform the project. **[TODO]** something about how the contract is public on etherscan and could be shared, therefor no need for confidentiality OR legally are we to keep confidentiality because we are exposing possible risk?

1. Introduction

This security audit report contains value information regarding the Turbo contract. Analysis of security vulnerabilities, smart contract best practices, and possible attack vectors are identified through use of standardised automated tests using static, dynamic, and symbolic analysis tools. We outlined our systematic approach to evaluate potential security issues within the core smart contract of this project and provide an audit summary with remedies for mitigating the vulnerability findings.

2. Project Context

This section explains the client's project in detail with scope.

ITEM	DESCRIPTION
Website	https://sites.google.com/view/turbotoad/home
Source	Smart contract program
Language	Solidity
Blockchain	Ethereum (ETH)
Project Location	https://etherscan.io/address/0xA35923162C49cF95e6BF26623385eb431ad920D3
Audit type	DEEP
Analysis Methods	Static and Dynamic automation tests, and Manual functional review.
Audit Team	Pura, Faizin, Marcus
Approved By	Nilanga (<i>Team lead</i>)
Timeline	From: 4 th MAY 2023 To: 5 th MAY 2023
Change logs	V1

3. Audit Scope

Scope of this project is to identify any identifiable vulnerabilities to improve the coding practices required for the given smart contract.

The following security audit steps were conducted:

- Automated testing using analysis tools which include static, dynamic, and Symbolic analysis methods.
- Manual functional reviews by our smart contract auditors.

Audit method:	BASIC	STANDARD	DEEP
----------------------	--------------	-----------------	-------------

ITEM	DESCRIPTION
Repository	https://etherscan.io/address/0xa35923162c49cf95e6bf26623385eb431ad920d3#code
Address	0xA35923162C49cF95e6BF26623385eb431ad920D3
Technical Documentation	Business logics provided: FALSE

4. Definitions

4.1. Security Severity

SEVERITY	DESCRIPTION
High	High level vulnerabilities may or may not be difficult to exploit, however they contain significant impact potential on the smart contract's execution due to lack of security access control (Example: Public access to crucial data and manipulation of functions).
Medium	Medium vulnerabilities may not lead to loss of assets or data, but it is important to fix these issues which may be used to exploit the project.
Low	Low level vulnerabilities are related to out-dated, unused code snippets. They may not have a significant impact on contract execution but is recommended to execute any fixes to them.
Informational	Does not contain vulnerabilities, but requires best practices, code standards and documentary code. It is recommended to execute a solution for these findings.

5. Analytic Statistics

Security Issues	#	Description	Type	Turbo
Programming Issues	1	Unchecked return value from low-level external calls	SWC-104	P
	2	Floating pragma is set	SWC-103	P
	3	Error Handling and logging are implemented	Custom	F
	4	State variable should not be used without being initialized	Custom	P
	5	Is inheritance used properly	SWC-125	P
	6	External components used insecurely	Custom	P
	7	Functions that loop over unbounded data structures	Custom	P
	8	Msg.value should not be used in a loop	Custom	P
Code Specifications, Best Practices	9	Use of the "constant" state mutability modifier is deprecated.	SWC-111	P
	10	Use of the "throw" state mutability modifier is deprecated.	SWC-111	P
	11	Strict equalities should not render the function to be unusable	Custom	P
	12	Use of best Practices	Custom	F
	13	Business logic is implemented as per the documents provided	Custom	
Optimise Gas	14	Message call with hardcoded gas amount	SWC-134	P
	15	Check for gas usage and minimize gas consumption.	Custom	P
Risk to Attacks	16	Code contains suicidal, greedy, and prodigal instructions	SWC-106	P
	17	Contract is Haltable	Custom	P
	18	Adopt checks-effects-interactions patterns for any transactions of value	Custom	P
	19	Reduce and remove unnecessary code to reduce attack surface area.	Custom	P
	20	Timestamps should not be used to execute critical functions.	SWC-116	P
	21	Sensitive data in normal form should not be stored on-chain	Custom	P
	22	Vulnerable to Integer overflow and under-flow	SWC-101	P

Key: F = Fail P = Pass

6. Manual Audit Process

6.1. Functions Overview of Turbo

#	Function	Type	Observation	Status
1	Constructor()	Constructor, Inherits ERC20	<ul style="list-style-type: none"> Inherits the ERC20 contract. Sets token name to Turbo and symbol as "TURBO". Calls the _Mint function to create and assign the initial supply of Turbo tokens to the msg.sender. Initial supply defined by INITIAL_SUPPLY constant. 	SAFE
2	distributeTokens(address distributionWallet)	External onlyOwner	<ul style="list-style-type: none"> Allows the contract owner to transfer the entire token supply to another wallet address. <p>Warnings:</p> <ul style="list-style-type: none"> Need to be careful when able to send all tokens to any address. If the owner renounces their ownership before transferring the ownership, then this function can never be accessed indefinitely. Additionally, it requires owner approval of transfer, burn and allowance. <p>Special Note: The contract was created on 29th April, 2023 by 0x315e6e028060e9b870e10ec5cab711ae2d566795, and soon after its creation on the same day, the owner has executed this function distributeTokens() and transferred all the Turbo tokens (69,000,000,000) to the following distribution Wallet address, before it renounced its ownership. 0x211Daf53D67c9bEF19336864C10675B99a699DA4 (Refer the transaction in Appendix)</p>	SAFE

7.1. INFORMATIONAL

1. Issue: Use of excessive digits

Function name: State variable declaration

```
INITIAL_SUPPLY = 69000000000 * 10 ** 18
```

Description:

It uses literals with too many digits and may be difficult to read and review.

Resolution:

Consider using scientific notation.

2. Issue: Inefficient Token Transfer

Function name: distributeTokens()

```
function distributeTokens(address distributionWallet)
external onlyOwner {
    uint256 supply = balanceOf(msg.sender);
    require(supply == INITIAL_SUPPLY, "Tokens already
distributed");

    _transfer(msg.sender, distributionWallet, supply);
}
```

Description:

This function is written to transfer all the tokens balance from the owner at a time. There is no way to transfer a specific amount of tokens to different distribution wallet addresses.

Resolution:

Instead of transferring all tokens by one transaction, the function could be modified to transfer only the specified number of tokens, while checking the current balance of the owner's wallet. Requires checks for when the balance is less than the total supply tokens.

9. Conclusion

The auditing team at BLOCKSTARS TECHNOLOGY was tasked with 1 smart contract from Turbo for a **DEEP** audit evaluation. The team has completed automation testing, manual review of codes and test cases review on the Turbo contract. The audit used static, dynamic, and symbolic analysis tools for reviewing each function within the contract.

The Turbo contract does not contain suicidal instructions; hence it is not vulnerable. It does not contain Call/Suicide; hence it is not prodigal, nor does it have lock vulnerabilities found because the contract cannot receive Ether.

Based on this analysis, the Turbo contract has identified 1 vulnerability and 3 informational that should be addressed:

TURBO 0 HIGH 0 MEDIUM 0 LOW 2 INFORMATIONAL

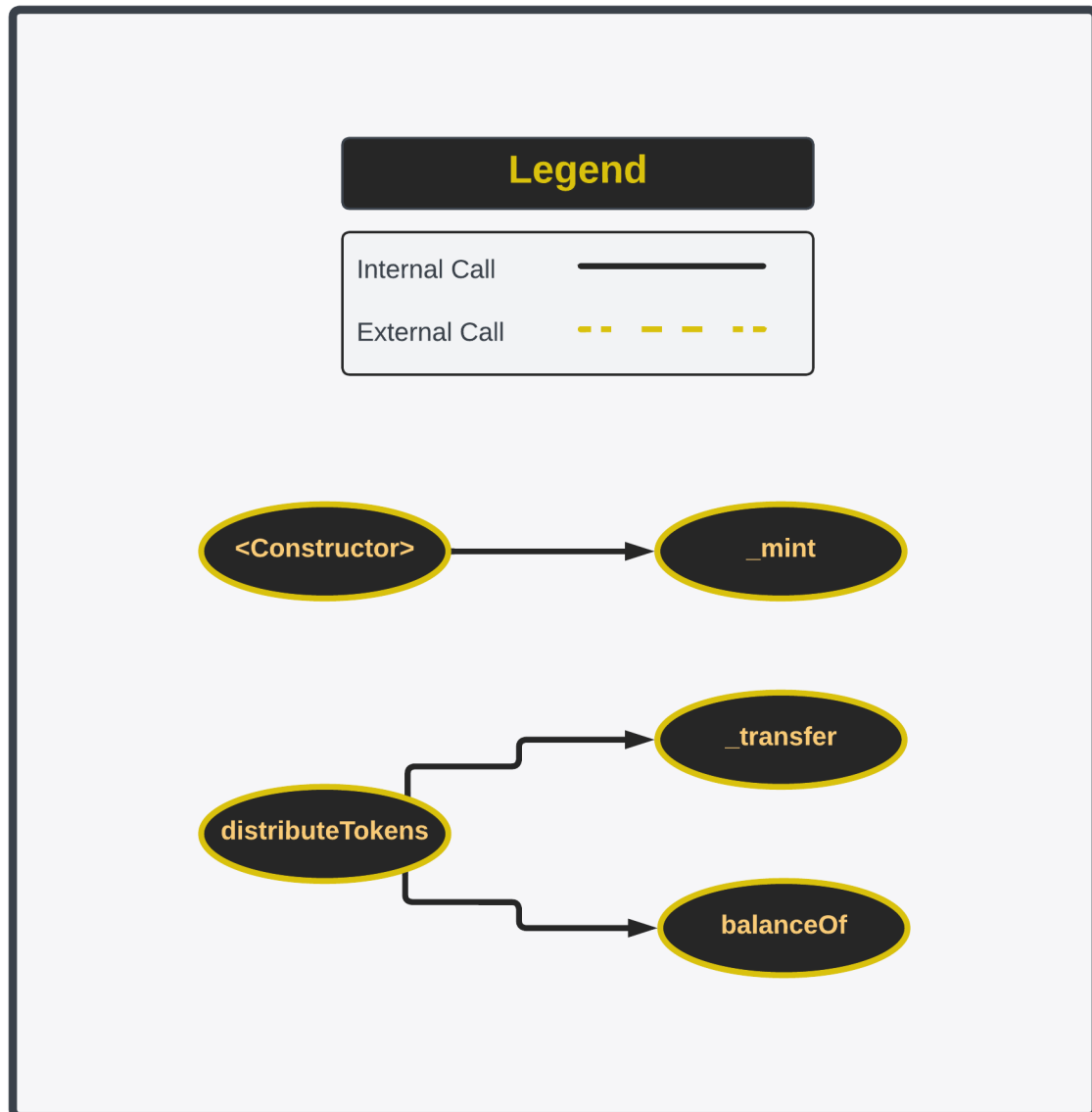
The Turbo contract in its current state is **SAFE** from potential attacks. Since it has distributed all the Turbo tokens, to other distributed wallet, before it renounced the ownership, there will be no issues of locking the tokens in the contract. Currently the contract renounced ownership, and all owner rights functions are not accessible.

It is strongly advised not to underestimate the potential impact of low-severity vulnerabilities, as intentionally leaving them unaddressed could leave the entire project vulnerable to exploitation.


In order to maintain the security of the project's contract/s, all identified vulnerabilities, regardless of their severity, should be promptly addressed and remediated. Neglecting to do so could result in significant harm to the project's integrity and reputation, and potentially even result in major losses. Therefore, it is imperative that regular security audits are conducted to identify and address any potential vulnerabilities from this point onward.

10.1. Functional Flow Chart

10.1.1. Turbo



10.2. Transaction hash of distributeTokens(address distributionWallet) function

Transaction Hash:	0x35ab4a97f92569578a2e6beec36a4c87b667f984c6e184b421df23baf6f55275
Status:	Success
Block:	17149234 71412 Block Confirmations
Timestamp:	10 days 55 mins ago (Apr-29-2023 03:59:23 AM +UTC) Confirmed within 30 secs
Sponsored:	
From:	0x315E6e028060e9b870e10Ec5Cab711AE2D566795
Interacted With (To):	0xA35923162C49cF95e6BF26623385eb431ad920D3
ERC-20 Tokens Transferred:	From 0x315E6e...2D566795 To 0x211Daf...9a699DA4 For 69,000,000,000 Turbo... (TURBO...)
Value:	0 ETH (\$0.00)
Transaction Fee:	0.00167542157004864 ETH \$3.10
Gas Price:	33.98972592 Gwei (0.00000003398972592 ETH)