KaJ Labs Foundation

KYC & AUDIT.

KaJ Labs Foundation specializing in blockchain technology solutions, Audits, KYC / Doxx.

CERTIFICATE OF COMPLIANCE

Smart Contract Audit by KaJ Labs

Atua AI Token

Audit Passed

09/17/2024

Table of Contents

Audit Summary

This report has been prepared for Atua AI Token on the ETH and BSC networks. KaJ Labs provides both client-centered and user- centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Ensuring contract logic meets the specifications and intentions of the client without exposing the user’s funds to risk.

- Testing the smart contracts against both common and uncommon attack vectors.

- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.

- Assessing the codebase to ensure compliance with current best practices and industry standards.

- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.

- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Parameter	Result
Address	0x36b2269FD151208a4bfc3DEA503E0a6F2485fA78 0x791A5c2261823dBF69b27B63E851B7745532Cfa2
Contract Name	BurnableTeamToken
Token Tracker	TUA
Decimals	18
Supply	4,999,999,999.999999
Platform	ETH and BSC
Compiler	v0.6.12+commit.27d51765
Optimization	Yes with 200 runs
Other Settings:	default evmVersion
Language	Solidity
Codebase	https://bscscan.com/token/0x36b2269fd151208a4bfc3dea503e0a6f2485fa78#code https://etherscan.io/token/0x791a5c2261823dbf69b27b63e851b7745532cfa2#code

Main Contract Assessed

Token Tracker	Contract	Live
TUA	0x36b2269FD151208a4bfc3DEA503E0a6F2485fA78 0x791A5c2261823dBF69b27B63E851B7745532Cfa2	Yes

Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
❖ Unencrypted Private Data On-Chain	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Code With No Effects	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Message call with hardcoded gas amount	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Hash Collisions With Multiple Variable Length Arguments	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Unexpected Ether balance	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Presence of unused variables	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Right-To-Left-Override control character (U+202E)	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Typographical Error	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ DoS With Block Gas Limit	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Arbitrary Jump with Function Type Variable	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Insufficient Gas Griefing	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Incorrect Inheritance Order	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Write to Arbitrary Storage Location	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Requirement Violation	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Missing Protection against Signature Replay Attacks	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Weak Sources of Randomness from Chain Attributes	✓ Complete	✓ Complete	✓ **Low / No Risk**

Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
❖ Authorization through tx.origin	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Delegatecall to Untrusted Callee	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Use of Deprecated Solidity Functions	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Assert Violation	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Reentrancy	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Unprotected SELFDESTRUCT Instruction	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Unprotected Ether Withdrawal	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Unchecked Call Return Value	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Outdated Compiler Version	✓ Complete	✓ Complete	✓ **Low Issues**
❖ Integer Overflow and Underflow	✓ Complete	✓ Complete	✓ **Low / No Risk**
❖ Function Default Visibility	✓ Complete	✓ Complete	✓ **Low / No Risk**

Contract Ownership

The contract ownership of Atua AI isn’t currently renounced. The owner has the power to call burn function and there isn’t renounced function the other write functions will be like the investors so no need to renounced the ownership

01

The current owner is the address

0x15C60dE480Ec1887fC220BD3377d4dD0d13DE947

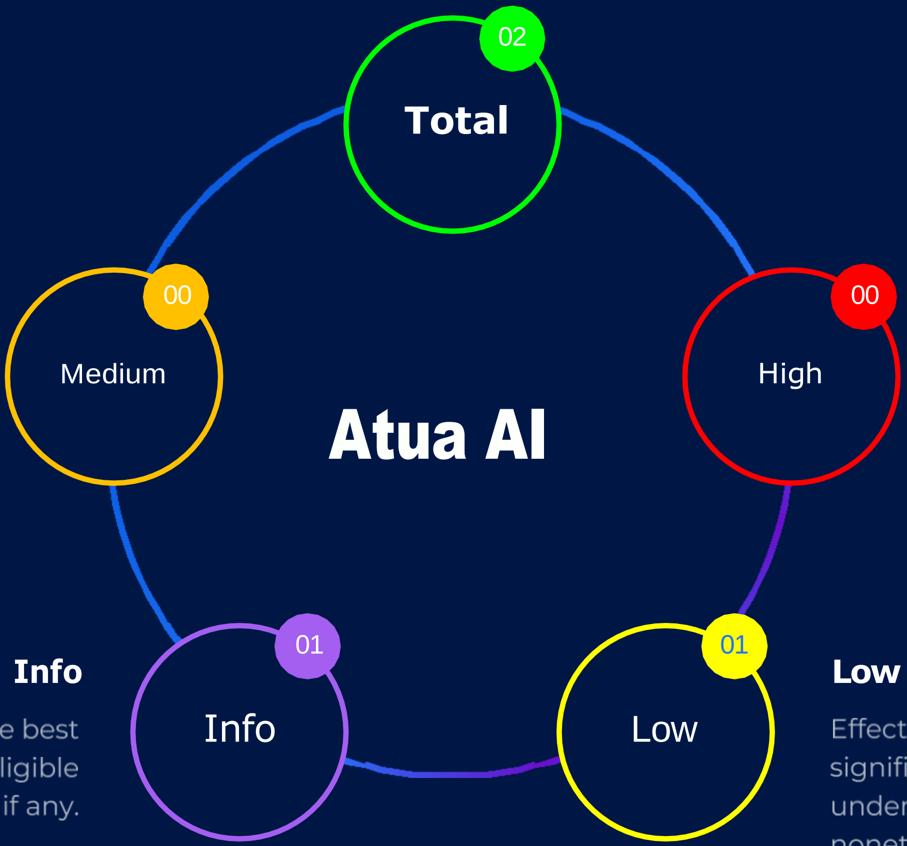
which can be viewed from: HERE

Technical Findings Summary

Classification of Issues

Total

What you should pay attention to



02 Total

00 Medium

00 High

Atua AI

01 Info

01 Low

Medium

Bugs or issues with 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.

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

Info

Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

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.

Findings

Pragam version not fixed

ID	Severity	Contract	Issue
01	Low	BurnableTeamToken	**The complier**

It is a good practice to lock the solidity version for a live deployment (use 0.8.2 instead of >=0.6.0 <0.8.0). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors. And avoid Solidity compiler Bugs check here

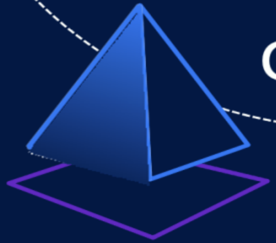
https://sepolia.etherscan.io/solcbuginfo

Statue:

Acknowledged.

Findings

Outdate Complier



ID	Severity	Contract	Issue
01	Informational	BurnableTeamToken	**The complier**

Description

The compiler being used was released 3 years ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

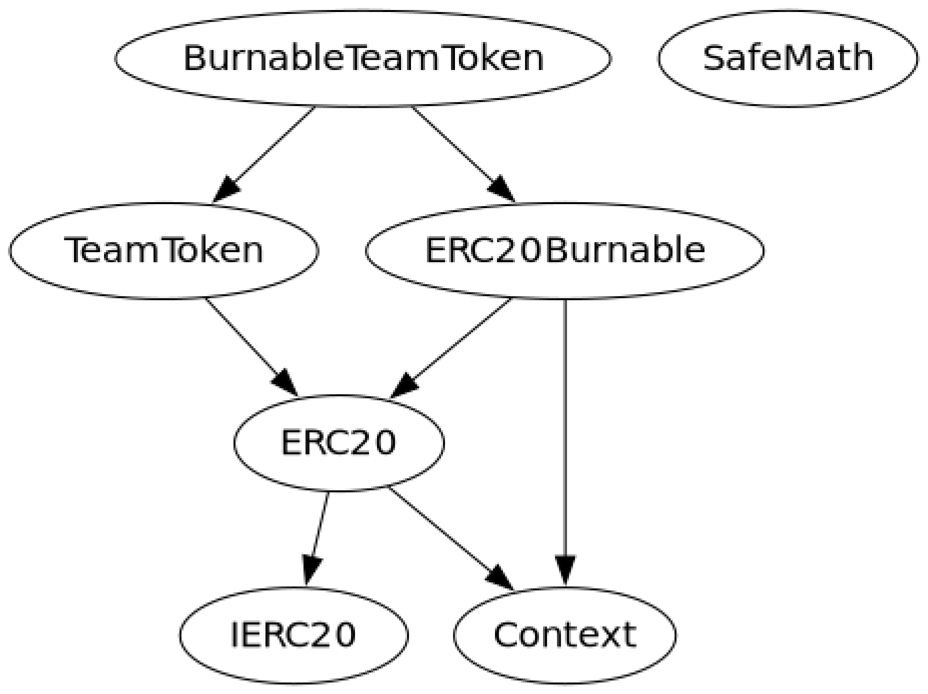
Statue:

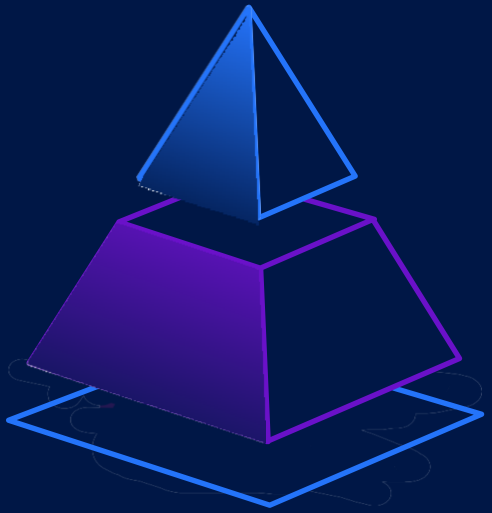
Acknowledged.

Privileged Functions (only Owner & Others)

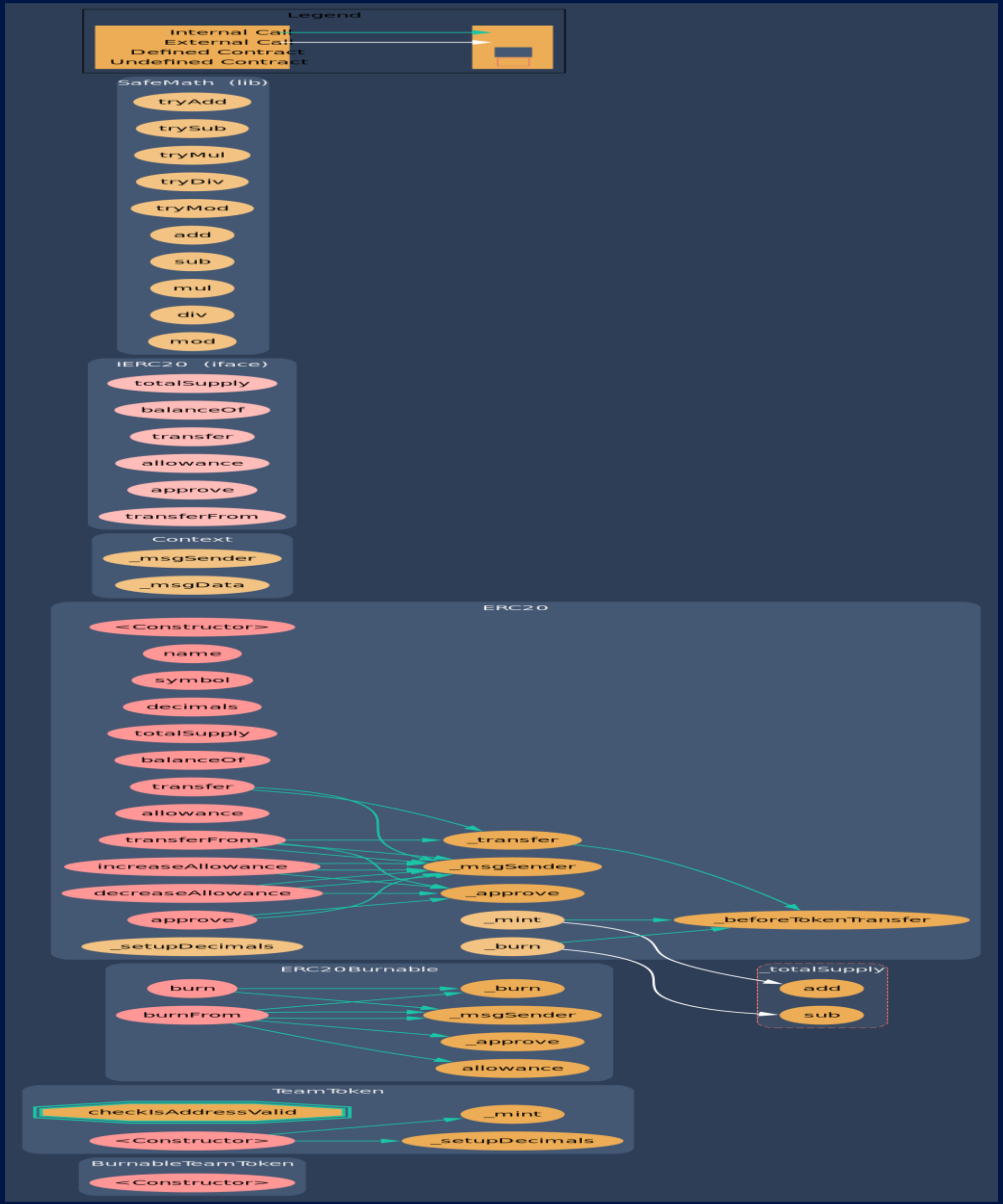
Function Name	Parameters	Visibility
✓ approve	▪ address	▪ **write/public**
✓ burn	▪ uint256	▪ **write/public**
✓ burnFrom	▪ uint256	▪ **write/public**
✓ transfer	▪ address and uint256	▪ **write/public**
✓ transferFrom	▪ address and uint256	▪ **write/public**
✓ decreaseAllowance	▪ address and uint256	▪ **write/public**
✓ increaseAllowance	▪ address and uint256	▪ **write/public**
✓ allowance	▪ address and uint256	▪ **read/public**
✓ name	▪ string	▪ **read/public**
✓ symbol	▪ string	▪ **read/public**
✓ balanceOf	▪ address	▪ **read/public**
✓ totalSupply	▪ uint256	▪ **read/public**
✓ descimal	▪ uint8	▪ **read/public**

Inheritance graph

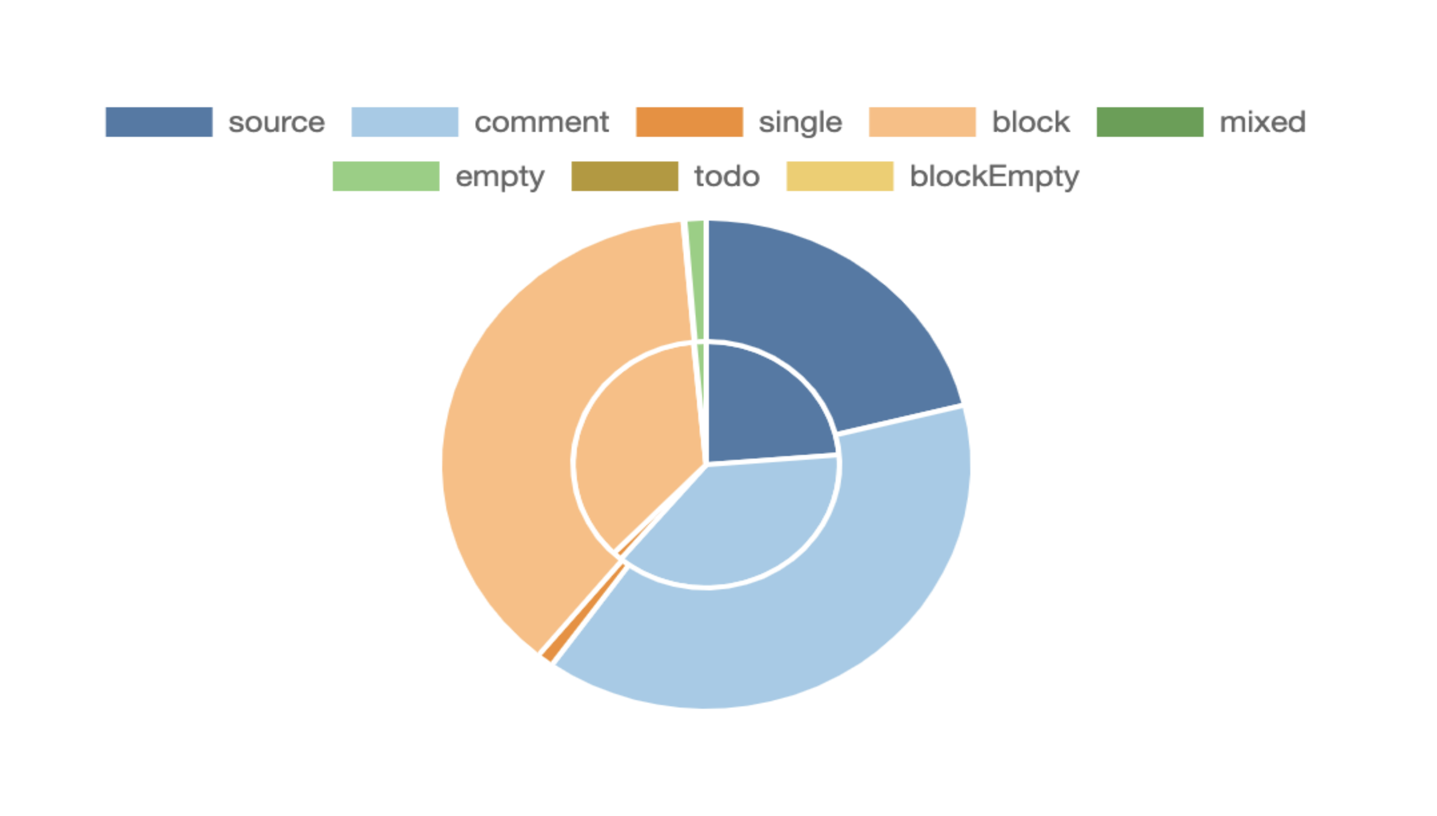




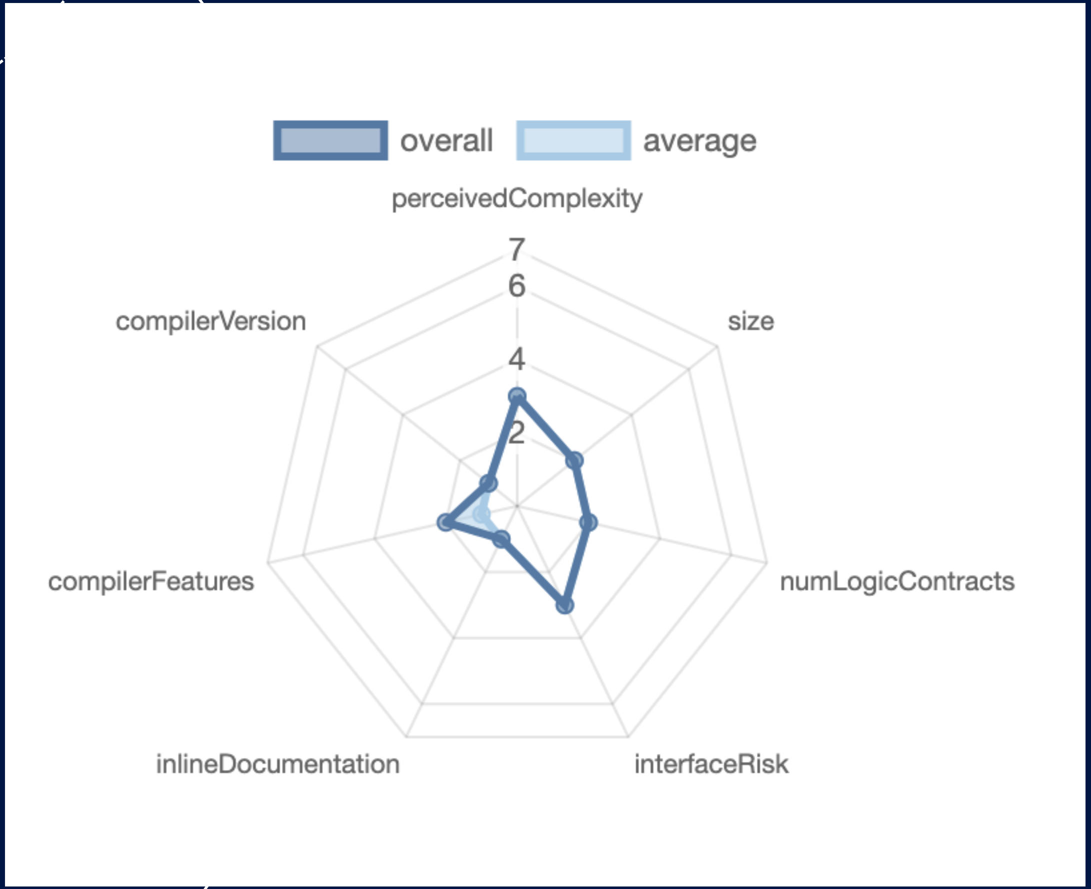
Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_setupDecimals
_transfer
_msgSender
_approve
_mint
_burn
_beforeTokenTransfer
_totalSupply
add
sub
ERC20Burnable
burn
burnFrom
_burn
_msgSender
_approve
allowance
TeamToken
checkIsAddressValid
<Constructor>
_mint
_setupDecimals
BurnableTeamToken
<Constructor>

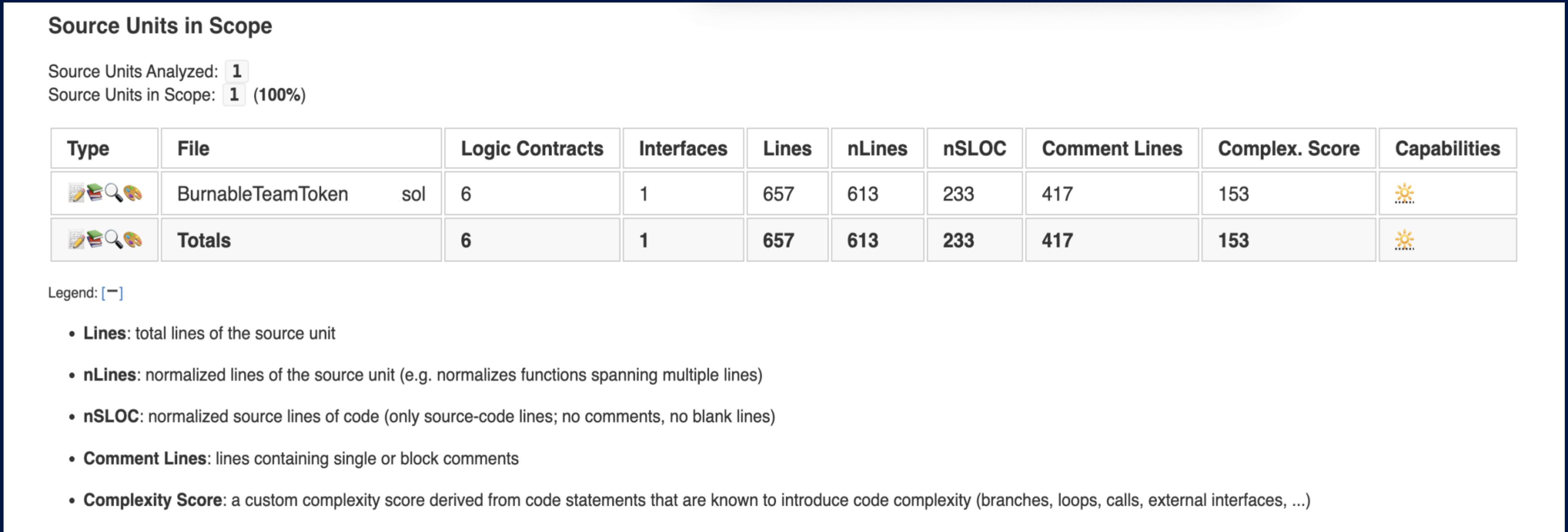
Source Lines



Risk Levels


overall
average
perceivedComplexity
size
numLogicContracts
interfaceRisk
inlineDocumentation
compilerFeatures
compilerVersion
7
6
4
2

Source unites in scope



Source Units in Scope

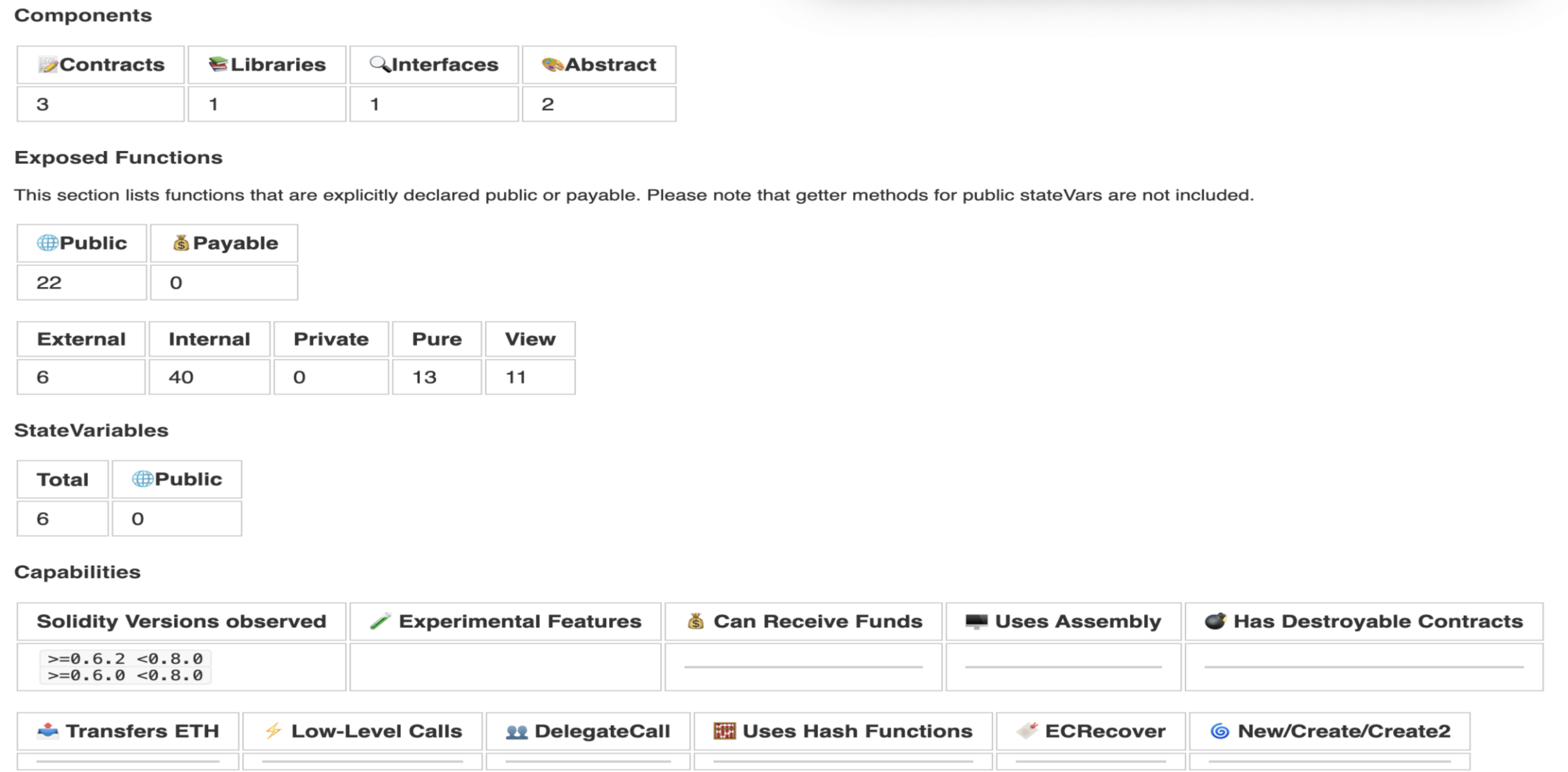
Source Units Analyzed: **1**
Source Units in Scope: **1** (**100%**)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	BurnableTeamToken sol	6	1	657	613	233	417	153	☀
📝📚🔍🎨	**Totals**	**6**	**1**	**657**	**613**	**233**	**417**	**153**	☀

Legend: [−]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities



Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
3	1	1	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
22	0

External	Internal	Private	Pure	View
6	40	0	13	11

StateVariables

Total	🌐Public
6	0

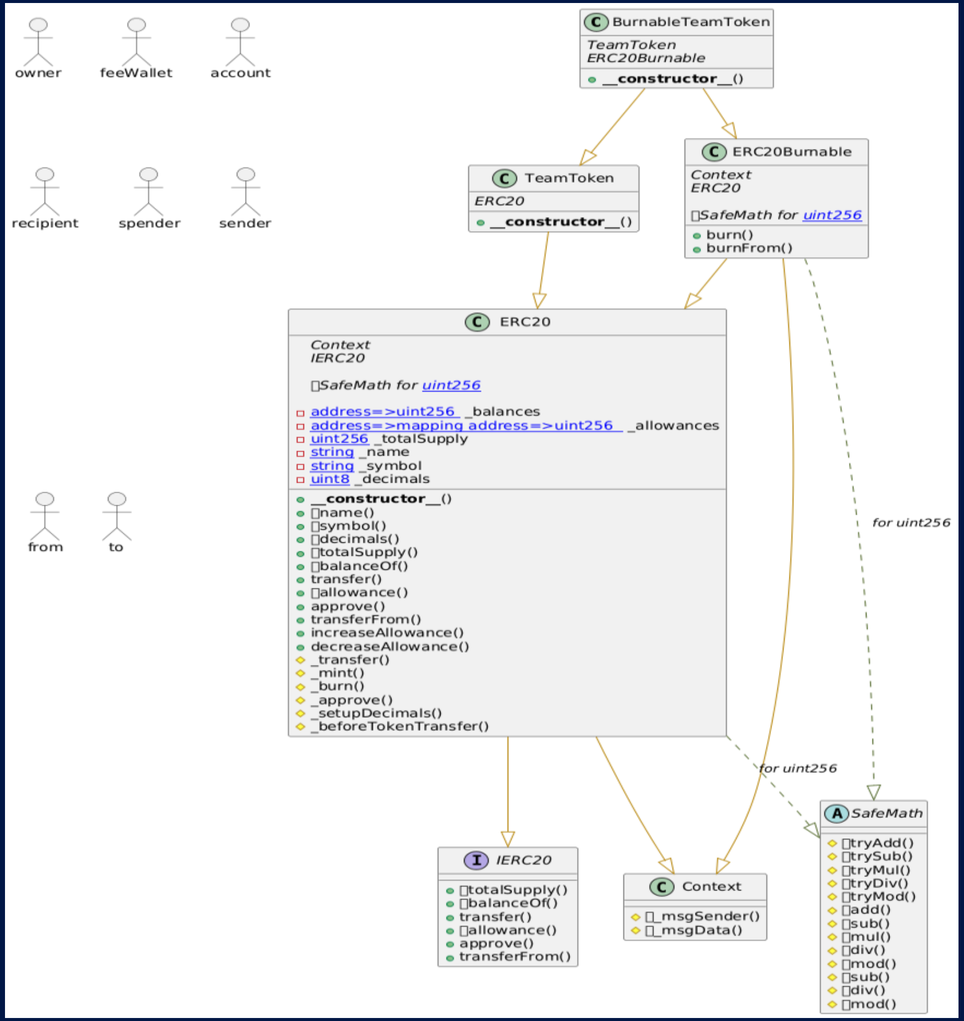
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

`>=0.6.2 <0.8.0`
`>=0.6.0 <0.8.0`
		—	—	—

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
—	—	—	—	—	—

Unified Modeling Language (UML)



Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract 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.

Security state of the reviewed contract is “Well Secured”.

✓ No volatile code.

✓ No high severity issues were found.

Disclaimer

KaJ Labs 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 advocation 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.

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 KaJ Labs is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will KaJ Labs 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.

The assessment services provided by KaJ Labs is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where- is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties