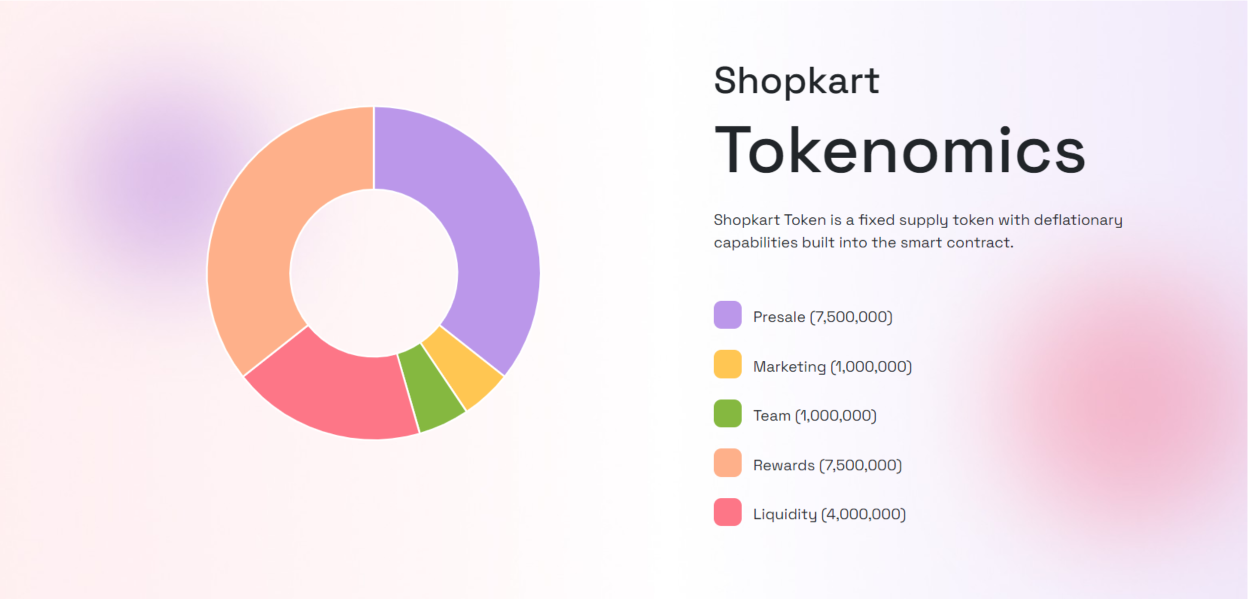
Overview

ShopToken by ShopKart

ShopKart is an e-commerce marketplace where users can order products from eBay, Amazon, Aliexpress and ShopKart's own marketplace, offering retailers and consumers a great experience and an opportunity to put crypto to use.

Name:	**ShopToken**
Symbol:	**SHOP**
Decimals:	**18**
Total Supply:	**21 Million**

Contract: ShopToken.sol



Shopkart

Tokenomics

Shopkart Token is a fixed supply token with deflationary capabilities built into the smart contract.

Scope of Audit

The scope of this audit was to analyse **ShopToken.sol** smart contract's codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return boolean
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of ShopToken smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the Smart contract is structured in a way that will not result in future problems.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimisation of code to reduce gas consumption.

Tools and Platforms used for Audit

Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
OPEN	0	1	3	0
CLOSED	0	0	0	0

Issues Found

High Severity Issues

None.

Medium Severity Issues

- **Unused Burning Functionality**:

  The contract implements functionality for the burning of tokens, with internal functions **_burn** and **_burnFrom** but are not accompanied with respective public/external functions.

Low Severity Issues

- [#L5] Old Compiler. Use the latest solidity compiler to avoid bugs introduced in the older versions.

- [#L538-544] function **_mint() :** Add check for **amount > 0**

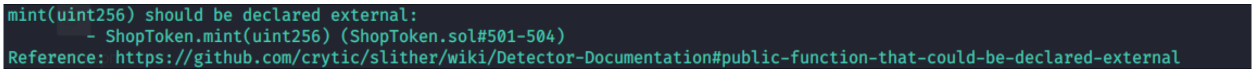
- **approve() race**
  The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender is able to spend their entire approval amount twice.

  Reference:
  - https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
  - https://medium.com/mycrypto/bad-actors-abusing-erc20-approval-to-steal-your-tokens-c0407b7f7c7c
  - https://eips.ethereum.org/EIPS/eip-20

Informational

None

Gas Optimization- public functions that are never called by the contract should be declared external to save gas.



```
mint(uint256) should be declared external:
        - ShopToken.mint(uint256) (ShopToken.sol#501-504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Automated Tests

Slither

Slither didn’t detect any high severity issues.

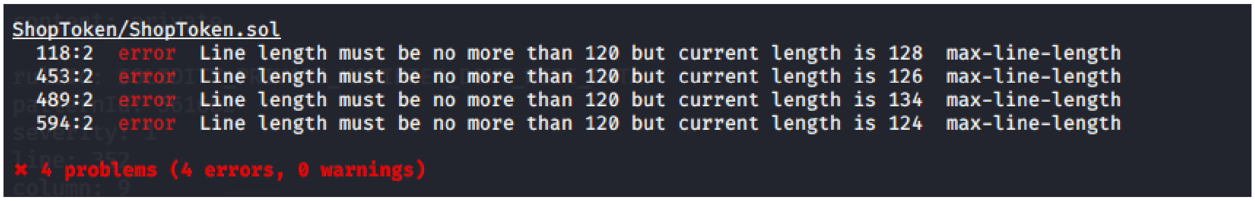
Mythril

Mythril didn’t detect any high severity issues.

Smartcheck

Smartcheck didn’t detect any high severity issues.

Solhint



```
ShopToken/ShopToken.sol
  118:2  error  Line length must be no more than 120 but current length is 128  max-line-length
  453:2  error  Line length must be no more than 120 but current length is 126  max-line-length
  489:2  error  Line length must be no more than 120 but current length is 134  max-line-length
  594:2  error  Line length must be no more than 120 but current length is 124  max-line-length

✖ 4 problems (4 errors, 0 warnings)
```

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides a security audit, please don’t consider this report as investment advice.

Summary:

Some issues of medium and low severity have been reported during the audit. Some suggestions have also been made to improve the code quality and gas optimisation. There were NO critical or major issues found that can break the intended behaviour.