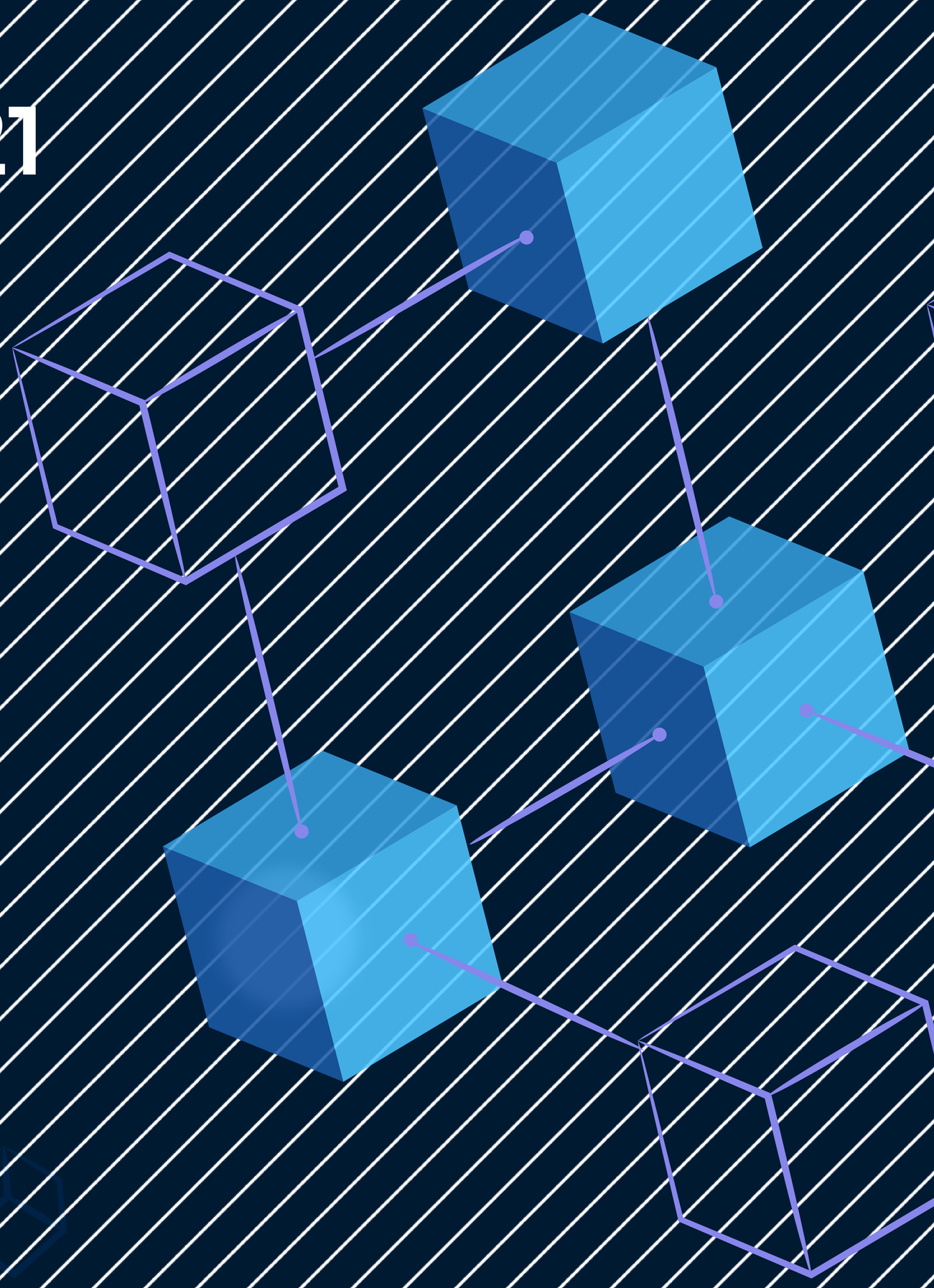
QuillAudits

Audit Report
December, 2021



For

Contents

Scope of the Audit

The scope of this audit was to analyze and document the ASVA Token Token smart contract 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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, 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 BEP-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 vulnerabilities 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 optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

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

Risk-level	Description
High	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	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	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 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	0	0	0
Acknowledged	0	0	0	0
Closed	0	0	0	1

Introduction

During the period of **Dec 16, 2021 to Dec 21, 2021** - QuillAudits Team performed a security audit for ASVA Token smart contracts.

The code for the audit was taken from following the official link: https://github.com/Asva-Labs-HQ/Asva-Token-V2

V	Date	Commit ID/Contract address
1	Dec 16	4945214f0752579a2838f5ea9c0c27958fa61dff
2	Dec 21	0x63433410a4c468b3fc8bfe375827a7f48c28ea83

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

1. **Using old Solidity version**

   Throughout the contract, Solidity 0.8.0 is used and is deployable. However, at the time of writing there are 2 known bugs in version 0.8.0. Consider upgrading your contracts to use the latest version of Solidity, 0.8.9.

Status: Fixed in version 02

Functional Tests

Address: 0x3b092D39Df38DC4Bd76C8835a0486dd8247925a7

Function name	Input	Output	TX	Status
balanceOf	0x1dd64394E29c5988f04A8E074D0DBACd4D614729	90000000000000000000000000	N/A	Passed
transfer	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","90000000000000000000000000"	true	0xce9922d24e66c09800b2d5547ab4adfb48d86de42308a319dd8f221eb1978bc4	Passed
approve	"0x153b057d5d7262dC92099B59c975255ecE66784F","90000000000000000000000000"	true	0x13f18fde6b111dbd8d56664052f79e9d19dbdea3142bc4512f6a170fa7e98061	Passed
allowance	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d7262dC92099B59c975255ecE66784F"	90000000000000000000000000	N/A	Passed
decreaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","90000000000000000000000000"	true	N0xda4cb2e31340d944ebb982b6fa55a87d8dadba15437ad5df1c8d2dfb7d5a32f7	Passed
increaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","90000000000000000000000000"	true	0xe86c763e4f5463522c7f51e7aefb909393f1908553457baeada95086c67fc70c	Passed
transferFrom	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d7262dC92099B59c97525	true	0x7294f274ee1b4f8cb1ae85b9976b99f1fe878e44103a3c0	Passed

	5ecE66784F",900000000000000000000000		138af996c76fb8c56	
balanceOf	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	0	N/A	Passed
balanceOf	0x153b057d5d7262dC92099B59c975255ecE66784F	900000000000000000000000	N/A	Passed
approve	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","100000000000000000000000"	true	0x43ad50ce6f3254aebadc84690114247545 1ac3365c2c45 1de4f9b9e6fc150dee	Passed
burnFrom	"0x153b057d5d7262dC92099B59c975255ecE66784F","100000000000000000000000"	true	0x3ea59b95687f931b5f69fdb809037495de0e86291879574ca7f39285c2f1791e	Passed
balanceOf	0x153b057d5d7262dC92099B59c975255ecE66784F	800000000000000000000000	N/A	Passed
burn	20000000000000000000000000	True	0x708bcc1fb35632c929d7bd9ab816d8d34bf2017c8777b88983425af21647dced	Passed
balanceOf	0x153b057d5d7262dC92099B59c975255ecE66784F	600000000000000000000000	N/A	Passed

Automated Tests

Slither

```
INFO:Detectors:
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (ASVA.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#150-164)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#178-181)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#197-205)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#20-22)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#35-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SOLHINT LINTER

```
ASVA.sol
  3:1  error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement                          compiler-version
  9:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility

✖ 2 problems (1 error, 1 warning)
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the ASVA Token platform. We performed our audit according to the procedure described above.

The audit showed an informational severity issue. In the end, the issue was Fixed by the Auditee.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the ASVA Token platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the ASVA Token Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
December, 2021

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com