Audit Whale

Smart contract code review and security analysis report

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publiclyafter all vulnerabilities are fixed — upon a decision of theCustomer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Asva Finance.
Approved by	Jameson | COO AuditWhale
Type	ERC20 token sale
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Asva-Labs-HQ/Asva-IDO-SC
Commit	d080f4f55218f430aab31f7947b57e2f43fcbf0f
Technical Documentation	YES
JS tests	YES
Website	asva.finance
Timeline	29 OCTOBER 2021 – 07 DECEMBER 2021
Changelog	02 NOVEMBER 2021 – INITIAL AUDIT 07 DECEMBER 2021 - SECOND REVIEW

Table of contents

Introduction

AuditWhale (Consultant) was contracted by Asva Finance (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between October 28th, 2021 – November 2nd, 2021.

Second review conducted on December 7th, 2021.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/Asva-Labs-HQ/Asva-IDO-SC
Commit:
d080f4f55218f430aab31f7947b57e2f43fcbf0f
Technical Documentation: Yes, included
JS tests: Yes, included
Contracts:
AsavaClaimFactory.sol
AsavaPoolFactory.sol
AsvaInvestmentsInfo.sol
Claimer.sol TierIDOPool.sol
interface/IClaimer.sol
interface/ITierIDOPool.sol

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

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency

	▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found 5 low severity issues.

After the second review security engineers found that some contracts were slightly changed. Therefore found 2 low severity issues.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Error in the fallback function declaration

   The fallback function that receives ether should be named fallback, be external payable, and declared without function keyword

   Contracts: Asva.sol

   Function: fallabck

   Recommendation: Fix function’s declaration

   Status: not an issue, source file is not in scope anymore

2. Tautology or contradiction Contracts:

   AsvaInvestmentsInfo.sol Functions:

   validAsvaId, validClaimId

   Recommendation: remove tautological comparisons asvaId >= 0 and claimId >= 0

3. Vulnerability: Mutable function available for everyone

   The function that changes state variables available for call from any address

   Contracts: AsvaInvestmentsInfo.sol

   Function: addPresaleAddress

   Recommendation: Add onlyOwner modifier and transfer ownership to AsavaPoolFactory after creation

   Status: fixed

4. View function could become inaccessible

   View function which iterates through the array of undefined size could be out-of-work in the case when there would be a huge amount of values inside the array.

   Contracts: AsvaInvestmentsInfo.sol

   Function: getInvestors

   Recommendation: Add offset and limit parameters to function getInvestors

   Status: fixed

5. A public function that could be declared external.

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

   Contracts: AsvaInvestmentsInfo.sol

   Functions: validClaimId

   Recommendation: Use the external attribute for functions never called from the contract.

6. Boolean equality

   Boolean constants can be used directly and do not need to be compared to true or false.

   Contracts: TierIDOPool.sol

   Function: addToPoolWhiteList (line #328)

   Recommendation: remove the equality to the boolean constant.

   Status: fixed

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found 5 low severity issues.

After the second review security engineers found that some contracts were slightly changed. Therefore found 2 low severity issues.

Disclaimers

AuditWhale Disclaimer

The smart contracts given for audit have been analyzed 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).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.