SAFETIN

AUDIT

Adashe

July 2nd, 2022

TABLE OF CONTENTS

AUDIT SUMMARY

This report was written for Adashe in order to find flaws and vulnerabilities in the Adashe project's source code, as well as any contract dependencies that weren't part of an officially recognized library given they were provided.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Adashe Deployment techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors
- Assessing the codebase to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Through line-by-line manual review of the entire codebase by industry expert

AUDIT OVERVIEW

PROJECT SUMMARY

Project name	Adashe
Description	Adashe is a crypto currency with emphasis on Space, DeFi, and Web 3.0
Platform	BNB Smart-chain
Language	Solidity
Codebase	MD5 Hash | File Name c62574822462d84b54fff01129a030c8 AdasheToken.sol 289df8a8d09c16e824a19e35429269d3 IAggregatorV3Interface.sol 6e3466a4131ec7a54bca6126c4853efa IWETH.sol 5a1517c2dc4f83819a0b20e5806cc817 LPFarming.sol b9a2397e90d5fcca4c18125bffca02ca NoContract.sol b56b9f8056f6fae1714bfdbf3016942a TokenSale.sol 5ab3aae5dd844123c9427eee147717f2 VestingWallet.sol

FINDINGS SUMMARY

Vulnerability	Total	Resolved
Critical	0	0
Major	0	0
Medium	3	3
Minor	4	4
Informational	3	3

EXECUTIVE SUMMARY

Adashe's space-based protocol will eliminate inefficiencies, high costs and restrictions in payments, transacting and record keeping by deploying a constellation of dual-function satellites.

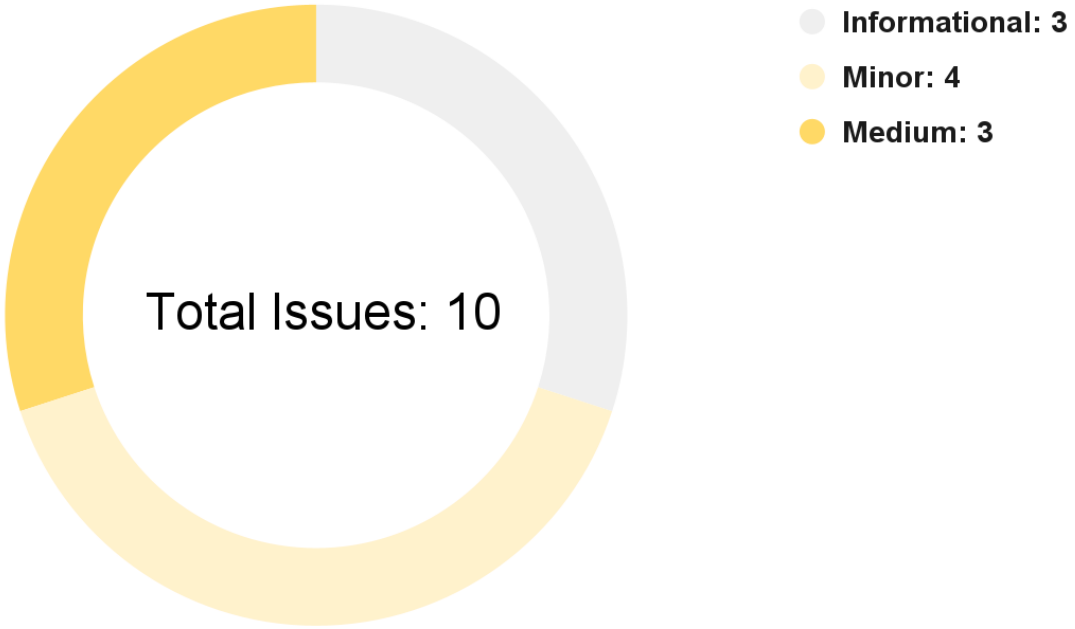
Developed by aerospace inventors, engineers, tech leaders and crypto pioneers, Adashe establishes a standardized industry framework for operating within the space ecosystem. It solves problems such as the lack of dedicated payment networks in the current space economy, and uses blockchain technology to reliably track and store ownership information in space. This is critical for the aerospace industry in addition to the asteroid mining and space tourism sub-sectors.

The aerospace industry is conservative, and the adoption of new technologies can take epochs. Project Adashe offers companies a unique opportunity to participate in the emerging blockchain universe without the traditional security concerns that are associated with payment and data storage systems.

Adashe's core technical team includes Dr. Periklis – a Stanford aeronautics PhD with NASA who helped land the Mars Curiosity rover – in addition to Uzo Mbamalu who is a specialist in foreign currency transactions. Our project's main impact will be felt across the Milky Way and on earth in multiple industries ranging from agriculture to IOT, finance and Web3 architecture.

There have been no critical issues related to the codebase and all findings listed here range from informational to medium. The medium security problem relates to the centralization of privileges, external dependencies and check-effects-interaction patterns.

AUDIT FINDINGS



Code	Title	Severity
CENT-1	Centralization of major privileges	Medium
EXT-1	External protocol dependencies	Medium
RE-1	Check-Effect-Interaction pattern	Medium
BLOC-1	Usage of block.timestamp	Minor
BLOC-2	Usage of block.number	Minor
MSG-1	Missing emit event	Minor
GAS-1	Expensive Functions	Minor

GAS-2	Unoptimised function type	Informational
COMP-1	Unlocked compiler version	Informational
FUNC-1	Redundant function	Informational

CENT-1 | Centralization of major privileges

Description

The onlyOwner modifier of the smart contract(s) gives major privileges over it (whitelisting contracts, transfer raised funds to treasury)*. This can be a problem, in the case of a hack, an attacker who has taken possession of this privileged account could damage the project and the investors.

*This list is not exhaustive but presents the most sensitive points

Recommendation

We recommend at least to use a multi-sig wallet as the owner address, and at best to establish a community governance protocol to avoid such centralization. For more information, see https://solidity-by-example.org/app/multi-sig-wallet/

EXT-1 | Dependence to external protocol

Description

The contract interacts with quickswap based protocols. The scope of the audit would treat these third party entities as black boxes and assume they are fully functional. However in the real world, third parties may be compromised thus leading assets to be lost or stolen. We fully understand that the business logic of the Adashe project is designed to work with quickswap based protocols. This extends to other protocols and interfaces not within the scope of this audit.

Recommendation

We encourage the team to constantly monitor the security level of the entirety of quickswap based protocols interacted with, as the security of the project is highly dependent on the security of these decentralized exchange platforms.

RE-1 | Check-Effect-Interaction pattern

Description

Some functions within Adashe's contracts make external function calls before relevant modifying state variables. This can lead to re-entrancy where the function can be called multiple times before the completion of the first execution. This can be problematic as such multiple invocation of said functions can still succeed even though they should have failed. Functions identified with this issue have been listed below.

- allocateTokensForSale -> TokenSale.sol | Line: 156

Recommendation

We recommend amending this function to have the modification of the availableTokens (TokenSale.sol, line 161) state variable take place **before** calling the safeTransferFrom function (TokenSale.sol, line 160). The TokenSale contract inherits from ReentrancyGuard from OpenZeppelin's ReentrancyGuard.sol so the nonReentrant modifier can also be used.

BLOC-1 | Use of block.timestamp

Description

The use of block.timestamp can be problematic. The timestamp can be partially manipulated by the miner (see https://cryptomarketpool.com/block-timestamp-manipulation-attack/).

Recommendation

We fully understand that the use of block.timestamp within the Adashe Protocol is required for certain functionality such as scheduling sales. Nevertheless, it is still useful to point out this kind of potential security problem.

BLOC-2 | Use of block.number

Description

The use of block.number can be problematic. The timestamp can be partially manipulated by the miner (see https://cryptomarketpool.com/block-timestamp-manipulation-attack/). Since the timestamp of a block cannot be fully trusted, the exact block counting at an exact timestamp cannot be fully trusted.

Recommendation

We fully understand that the use of block.number within the Adashe Protocol is required for epoch functionality. Nevertheless, it is still useful to point out this kind of potential security problem.

MSG-1 | Missing event emits

Description

Some functions within Adashe's contracts modify sensitive variables without emitting an event. This includes functions which modify and add pools . Functions with this issue are listed below:

- add -> LPFarming.sol | Line: 123
- set -> LPFarming.sol | Line: 142

Recommendation

We recommend amending these functions to include event emits to ensure transparency with users.

GAS-1 | Expensive functions

Description

Some functions change state variables within a for loop. This can be incredibly expensive in terms of gas and can lead to gas exhaustion. The scope of this issue has been limited to functions which would be expected to be called often. Functions found with this issue are listed below:

- _massUpdatePools -> LPFarming.sol | Line: 261
- add* -> LPFarming.sol | Line: 123
- set* -> LPFarming.sol | Line: 142

*The add and set function do not contain an expensive for loop but call the _massUpdatePools function which does.

Recommendation

The function _massUpdatePools updates poolInfo for all pools. This function is called by functions add and set every time they are called. Through analysis of the code's logic this seems excessive as it can be expected that multiple pools may be added through the add functions (or set with the set function) within a short timeframe and in each instance the expensive _massUpdatePools function would be called whereas calling it just for the last pool to be added (or set) may be more appropriate. Therefore, we recommend introducing a bool parameter which would allow the decision to call _massUpdatePools to be decided on a per call basis through an if statement.

GAS-2 | Unoptimized function type

Description

Throughout Adashe's contracts some functions are of type public although they are never called within the contract. External functions require significantly less gas to call. Such found functions are listed below:

- release -> VestingWallet.sol | Line: 104
- release -> VestingWallet.sol | Line: 116

Recommendation

We recommend reviewing each of the functions listed above and where possible switch their type from public to external.

COMP-1 | Unlocked compiler version

Description

Adashe's contract does not have locked compiler versions, meaning a range of compiler versions can be used. This can lead to differing bytecodes being produced depending on the compiler version, which can create confusion when debugging, as bugs may be specific to a specific compiler version(s).

Recommendation

To rectify this, we recommend setting the compiler to a single version, the version tested the most to be compatible with the code, an example of this change can be seen below.

```
// Line 17, TokenSale.sol
pragma solidity 0.8.0;
```

FUNC-1 | Redundant function

Description

The interaction of the function renounceOwnership (LPFarming.sol, line 346) within Adashe's contract isn't part of the official OpenZeppelin Ownable.sol contract. This function doesn't provide any functionality to the contract as it does not renounce ownership but reverts with the message "Cannot renounce ownership".

Recommendation

We understand that to maintain functionality within the contract ownership cannot be renounced thus we recommend removing this redundant function from the contract.

Global security warnings

These are safety issues for the whole project. They are not necessarily critical problems but they are inherent in the structure of the project itself. Potential attack vectors for these security problems should be monitored.

CENT-1 | Global SPOF (Single Point Of Failure)

The project's smart contracts often have a problem of centralized privileges. The owner system in particular can be subject to attack. To address this security issue we recommend using a multi-sig wallet, establishing secure project administration protocols and strengthening the security of project administrators.

Compliance with industry standards

The way the contract is developed and its compliance with industry standards are part of the project. In order to increase the optimization of the latter, we recommend refining the code to best fit industry best practices, in particular the use of error messages and library utilization.

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement.

This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without Safetin's prior written consent.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Safetin to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance. This report should not be used in any way

Safetin security assessment to make decisions around investment or involvement with any particular project.

This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Safetin's position is that each company and individual are responsible for their own due diligence and continuous security. Safetin's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or fun.