SAFETIN AUDIT

AGORA BANK

June 3rd, 2022

TABLE OF CONTENTS

AUDIT SUMMARY

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

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Agora Bank 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	Agora Bank
Description	AgoraBank is a decentralized crypto bank owned by its users. AgoraBank modernizes the banking system by gathering institutional and cryptocurrency services within a single application.
Platform	BNB Chain
Language	Solidity
Codebase	0xa58034453A116D6d33D02e7DA245F933520f957a (SWAP CONTRACT) 0x3903664601Fa6795899eeD287B9F6fbb6795851B (NFT CONTRACT)

FINDINGS SUMMARY

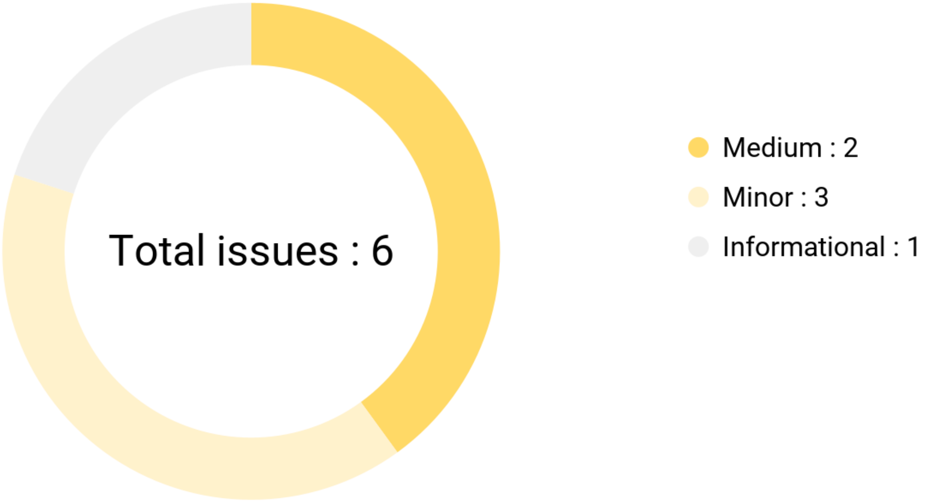
Vulnerability	Total	Resolved
● Critical	0	0
● Major	0	0
● Medium	2	0
● Minor	3	3
● Informational	1	1

EXECUTIVE SUMMARY

Agora Bank is a decentralized bank owned by its users. Technically, it aims to be a multichain DEX (Decentralized EXchange). As a DAO, the Agora Bank project is also composed of a token, the AGO, which represents the participation of project members in new features, marketing budget, and management of reserves for future development. The project is also accompanied by an NFT, to pre-order AGO tokens during a presale. However, this audit is about the swap contract (NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol) and the NFT contract (NEXT_PUBLIC_AGORA_NFT_CONTRACT.sol).

There have been no major or critical issues related to the codebase and all findings listed here are minor or informational. The major security problem is the centralization of privileges.

AUDIT FINDINGS



Code	Title	Severity
CENT-1	Centralization of major privileges	● Medium
COMP-1	Unfixed version of compiler	● Minor
THRE-1a	Missing threshold for minor func	● Minor
UINT-1	Unoptimized uint size	● Informational

UINT-1 | Unoptimized uint size

Description

Some variables in the contract are of type uint, but not of the right size. In order to optimize gas costs when deploying and using the contract, we invite you to assign the right size uint to each variable.

2 errors of this type have been found in NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol.

Recommendation

```
//Edited code containing appropriate uint sizes
//l 22  :Since percentage is under 10 000, uint16 is
enough
uint16 percentage;
//l 161 : Since percentage is under 10 000, uint16 is
enough (fee  = fees[i].percentage)
uint16 fee = 0;
```

COMP-1 | Unfixed version of compiler

Description

Both smart contracts do 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).

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

1 error of this type has been found in

NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol

1 error of this type has been found in

NEXT_PUBLIC_AGORA_NFT_CONTRACT.sol

Recommendation

We recommend fixing the compiler version to the most recent one :

```
//Edited code containing fixed compiler version in
NEXT_PUBLIC_AGORA_NFT_CONTRACT.sol
//l2
pragma solidity 0.8.0;
-----------------------
//Edited code containing fixed compiler version in
NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol
```

```
//l2
pragma solidity 0.8.0;
```

THRE-1 | Missing threshold for fees setting function

Description

The fee setting function does not have a safety threshold. Even though this function is protected by the onlyOwner modifier, it is important to add a threshold to prevent an attacker from setting fees to 100%

1 error of this type has been found in NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol.

Recommendation

We recommend adding a threshold to the concerned function. We leave it to you to decide which threshold best fits the logic of the project :

```
//Edited code containing safety thresholds
//l 220
function setFees(uint256 level, Fees memory fee) external
onlyOwner {
    require(fee.percentage < 10000, "Fees cannot be equal
to 100%");
    fees[level] = fee;
}
```

CENT-1 | Centralization of major privileges

Description

The onlyOwner modifier of both smart contracts gives major privileges over it (for NEXT_PUBLIC_AGORA_NFT_CONTRACT.sol the ability to pause the NFT transactions and to change the price of each token; for NEXT_PUBLIC_AGORA_SWAP_CONTRACT.sol, the ability to set the address that receives the funds, as well as the address that receives the fees)*. 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.

*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/

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 onlyOwner 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.

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.