STAYSAFU
AUDIT

May 1st, 2023

XAI

TABLE OF CONTENTS

I. SUMMARY

II. OVERVIEW

III. FINDINGS

CENT-1 : Centralization of major privileges

CENT-2 : Centralization of initial token distribution

EXT-1 : Dependence to external protocol

MAN-1 : Bad management of the blocking time of the last transfer

LACK-1 : Lack of events to track important actions

IV. GLOBAL SECURITY WARNINGS

V. DISCLAIMER

AUDIT SUMMARY

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

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and XAI 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

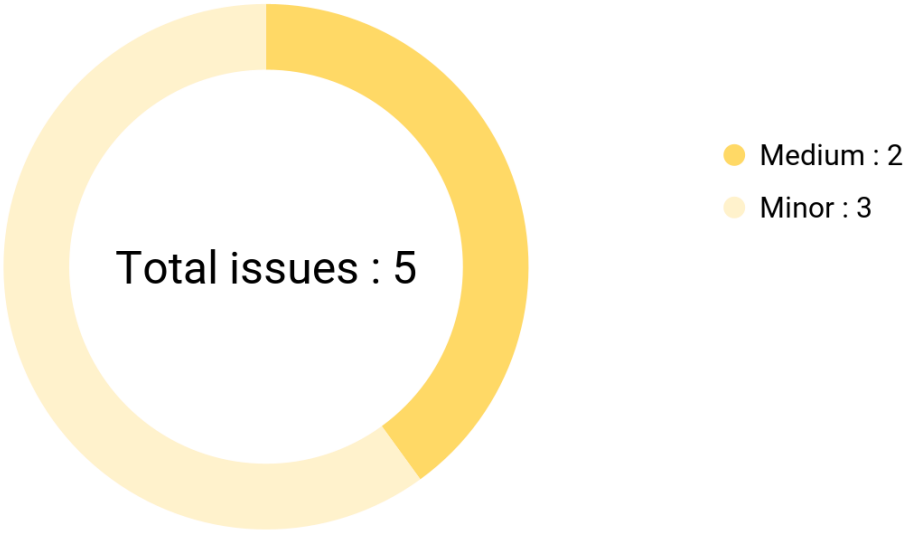
Project name	XAI
Description	XAI is an application that uses artificial intelligence and machine learning algorithms to create unique digital assets that can be sold as NFTs.
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0xd16eaaba33a0822f5cbe4e0c63ca51d3c3fbb08b#code

FINDINGS SUMMARY

Vulnerability	Total
Critical	0
Major	0
Medium	2
Minor	3
Informational	0

EXECUTIVE SUMMARY

XAI is an application that uses artificial intelligence and machine learning algorithms to create unique digital assets that can be sold as NFTs. NFTs are blockchain-based tokens that represent ownership of digital assets, such as artwork, music, videos, or any other type of creative content. AI can have a significant impact on the crypto industry in several ways, including trading, fraud detection, security, portfolio management, and DApps.



AUDIT FINDINGS

Code	Title	Severity
CENT-1	Centralization of major privileges	Medium
CENT-2	Centralization of initial token distribution	Medium
EXT-1	Dependence to external protocol	Minor
MAN-1	Bad management of the blocking time of the last transfer	Minor
LACK-1	Lack of events to track important actions	Minor

CENT-1 | Centralization of major privileges

Description

The onlyOwner modifier of the smart contract(s) gives major privileges over it (setUniswapPairAddress, addToWhiteList, setTimeout…)*. 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/

CENT-2 | Centralization of initial token distribution

Description

A constructor (line 506) within the contract mints the initial token supply to the deployer address (msg.sender). This initially centralizes token supply to the deployer address.

Recommendation

We recommend decentralising tokens as soon as possible, matching the project's intentions. Examples of this are burning tokens or adding tokens to a liquidity pool (locked). We also recommend being fully transparent with the community about token distribution.

EXT-1 | Dependence to external protocol

Description

The contract interacts with Uniswap 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 XAI project is designed to work with Uniswap 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 Uniswap protocols interacted with, as the security of the project is highly dependent on the security of these decentralized exchange platforms.

MAN-1 | Bad management of the blocking time of the last transfer

Description

Currently, the contract updates lastTransfer[recipient] every time a transfer is made from uniswapPairAddress. This means that a malicious actor could reset his blocking time simply by sending a small amount of tokens from uniswapPairAddress to his address. This bypasses the blocking mechanism that is supposed to prevent "front-running" bots.

Recommendation

Change the update logic of lastTransfer so that it is only activated when the user transfers to uniswapPairAddress. This will prevent a user from resetting their block timeout simply by making a small transfer from uniswapPairAddress. The logic of _checkTimeout can be modified as follows:

```
function _checkTimeout(address sender, address recipient) internal {

if (sender == uniswapPairAddress || _timeoutWhitelist[sender]) { lastTransfer[recipient] = block.timestamp; } if (!_timeoutWhitelist[sender] && recipient == uniswapPairAddress) { require(block.timestamp >= lastTransfer[sender] + timeout, "lock for prevent front runner bot."); } }
```

With this change, lastTransfer[recipient] is updated whenever a transfer is made from uniswapPairAddress or if the sender is whitelisted. Thus, attempting to reset the blocking time by making a small transfer from uniswapPairAddress would not affect the blocking time of a non-whitelisted user trying to make a transfer to uniswapPairAddress.

LACK-1 | Lack of events to track important actions

Description

Events in smart contracts are important tools for monitoring actions that occur in the contract. In your contract, actions such as adding or removing addresses from the whitelist, changing the timeframe and transferring tokens are important, but there are no events that are triggered when these actions occur.

For example, when the addToWhitelist function is called, it changes the state of the contract by adding a new address to the whitelist, but no event is emitted to record this action. This means that it is more difficult to monitor changes to the whitelist, as developers or users have to manually inspect the contract state.

Recommendation

Add events for each important action in the contract. For example, you can define a WhitelistUpdated event that is issued whenever an address is added or removed from the whitelist. Here is how you can do it:

```
event WhitelistUpdated(address indexed account, bool isWhitelisted);


function addToWhitelist(address _address) external onlyOwner {
    require(!_timeoutWhitelist[_address], "Address already set in white list");
    _timeoutWhitelist[_address] = true;
    emit WhitelistUpdated(_address, true);
}


function removeFromWhitelist(address _address) external onlyOwner {
    require(_timeoutWhitelist[_address], "Address isn't in white list");
    _timeoutWhitelist[_address] = false;
    emit WhitelistUpdated(_address, false);
}
```

Also, you can add events for setTimeout, setUniswapPairAddress and transfer functions. This makes it easier to track actions on the contract and improves transparency for users.

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 contract has 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.

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.

The Company only to the extent permitted under the terms shall use this report provided in connection with the Services set forth in the Agreement 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 StaySAFU'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 StaySAFU 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 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.

StaySAFU's position is that each company and individual are responsible for their own due diligence and continuous security. StaySAFU'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.