SOLIDProof

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams. Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Degen Brain Finance
Website	https://dbx.finance/
About the project	DegenBrains Exchange (DBX) is a decentralized cross-chain exchange, built by the Degen Brain Finance (DBF) team. Available first on the Arbitrum blockchain, with expansions planned to Cronos and Base. DBX will offer you Token Swaps, LP Farming, NFT staking, Farm Boosting, Play2Earn Games, and more!
Chain	Arbitrum
Language	Solidity
Codebase Link	https://arbiscan.io/address/0x565C093907E5D8148e5964A6cae76780140c3004 https://arbiscan.io/address/0x41755a88d4ab443d5d8ef73ec64dd3df71fb5559#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	N/A
Twitter	https://twitter.com/DbxFinance
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@degenbrains
Discord	https://discord.com/invite/degenbrains
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	26. June 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	28. June 2023	• Reaudit

Note - This Audit report consists of a security analysis of the **Degen Brain Finance** smart contract. This analysis did not include functional testing (or unit testing) of the contract's logic.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/Router.sol	4d3d60c70fe61438a78f3b79951ea1026825cbd1
contracts/Factory.sol	25da1e5a3c6a216b240b8f6e6182ac19ef2e5837

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).
The code and all the libraries are a fork from uniswap and pancake swap
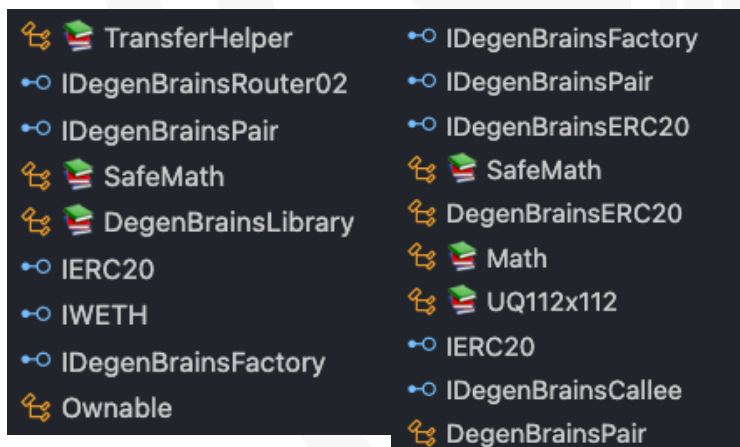
with only name changes and custom UI.

Note for Investors: We only Audited Swap and Router contracts for **Degen Brain Finance**. However, If the project has other contracts (for example, a Presale, NFT, or staking contracts etc), and they were not provided to us in the audit scope, then we cannot comment on its security and are not responsible for it in any way.

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to
      SolidProof to ensure we understand the size, scope, and functionality of the
      smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security
Medium or higher issues

No critical Issues found	✅ Contract is safe to deploy
Description	The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.
Comment	The code is 1:1 forked from uniswap/pancake swap with only differences are the project names in the file. The project owners confirmed this by the following comments: *"Uniswap yes, Custom UI, Forked Uniswap/PCS, No edits but our name"*

Upgradeability

Contract is not an upgradeable	✅ Deployer cannot update the contract with new functionalities
Description	The contract is not an upgradeable contract. The deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

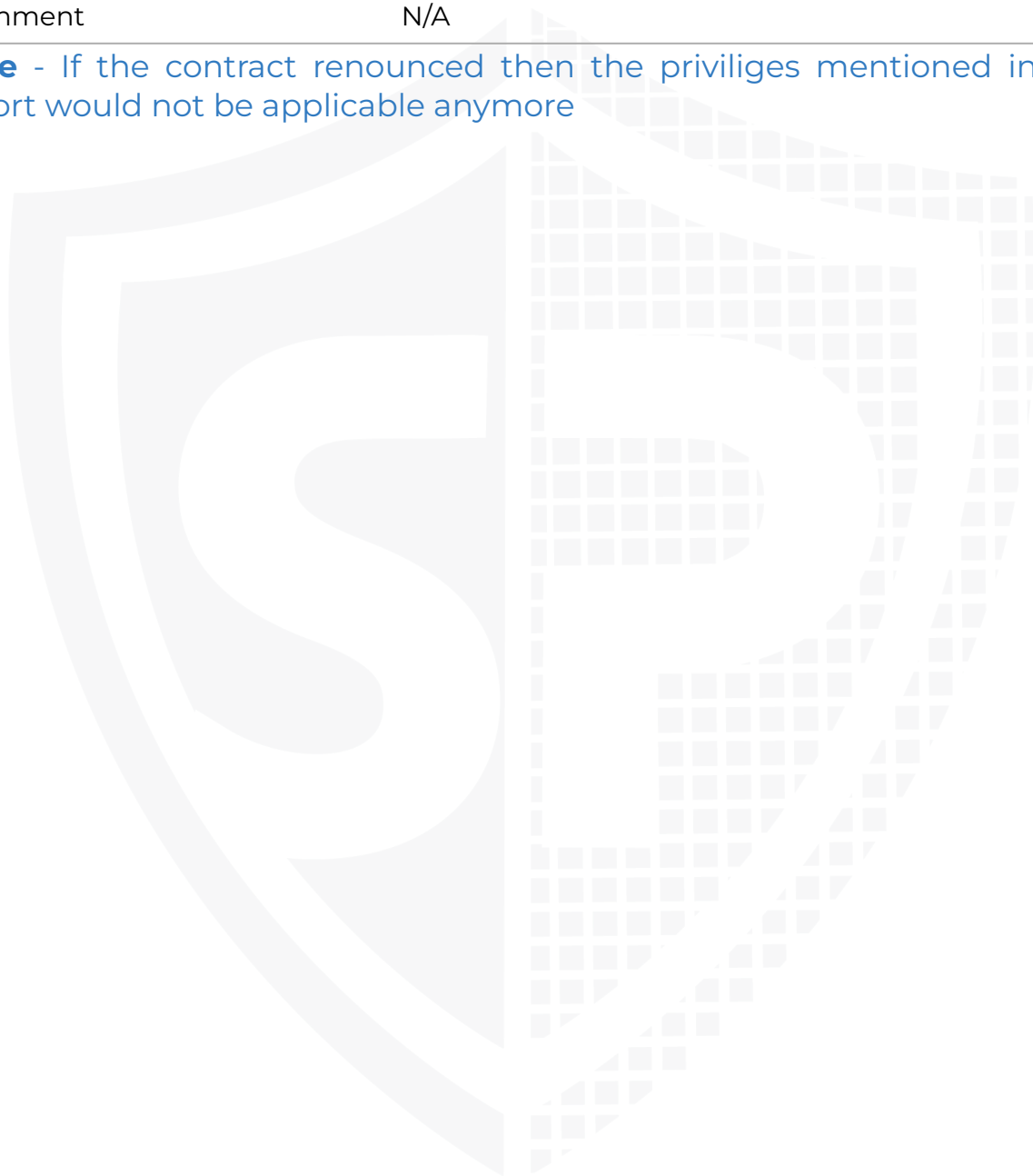| The ownership is renounced	✅ The owner is renounced
Description	The owner renounced the ownership that means the contract's owner will no longer have any control or authority over the contract's operations.
Comment	N/A

Note - If the contract renounced then the priviliges mentioned in the report would not be applicable anymore

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss.
We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or
blockchain network. This process is typically performed by the project's owner or
designated authority, who has the ability to add new tokens to the network's total
supply.

Contract owner cannot mint new tokens	✅ The owner cannot mint new tokens
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ The owner cannot burn tokens
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses	✅ The owner cannot blacklist addresses
Description	The owner is not able blacklist addresses to lock funds.
Comment	N/A

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

Contract owner cannot set fees more than 10%	✅ The owner cannot blacklist addresses
Description	The owner is not able to set the fees above 10%
Comment	N/A

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Owner cannot lock the contract	✅ The owner cannot lock the contract
Description	The owner is not able to lock the contract by any functions or updating any variables.
Comment	N/A

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
5	6	10	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
177	10

External	Internal	Private	Pure	View
160	133	5	38	61

StateVariables

Total	🌐 Public
31	24

Capabilities

Solidity Versions observed	💰 Can Receive Funds	🖥️ Uses Assembly
=0.6.6 =0.5.16	yes	yes (2 asm blocks)

📤 Transfers ETH	🧮 Uses Hash Functions	🧾 ECRecover	🌀 New/Create/Create2
yes	yes	yes	yes → AssemblyCall:Name:create2

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.
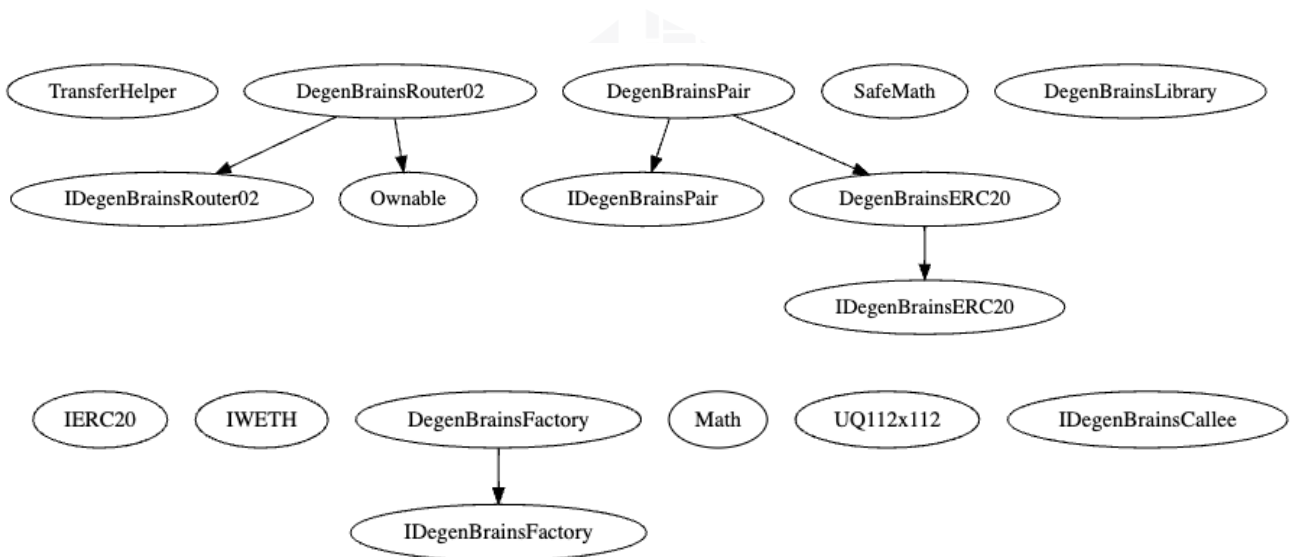
Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
1. Factory.sol	‣ The fee to setter address can set the swap fees up to 10%
2. Router.sol	‣ N/A

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the project owners can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Results

#1 | Old Compiler version

File	Severity	Location	Status
All	**Low**	L2	**ACK**

Description

- The contracts use outdated compiler versions, which are not recommended for deployment as they may be susceptible to known vulnerabilities.

#2 | Missing Zero Address Validation

File	Severity	Location	Status
Router.sol	**Low**	L402	**ACK**

Description

- Make sure to validate that the address passed in the constructor parameters is "non-zero" because these addresses cannot be modified again.

#3 | Missing Zero Address Validation

File	Severity	Location	Status
Factory.sol	**Low**	L301	**ACK**

Description

- Make sure to validate that the address passed in the initializer parameters is "non-zero" because these addresseses cannot be modified again.

#4 | Contract doesn't import npm packages from source (like OpenZeppelin etc.)

File	Severity	Location	Status
All	**Informational**	N/A	**ACK**

Description

- We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

MADE IN GERMANY

Solid Proofed

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY