



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

MagicFoxFi

Audit

Security Assessment
15. April, 2023

For



MAGICFOX



SolidProof_io



@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Links	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	17
Source Units in Scope	19
Critical issues	20
High issues	20
Medium issues	20
Low issues	20
Informational issues	20
Audit Comments	21
SWC Attacks	22

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	12. April 2023 - 14. April 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Arbitrum, Binance Smart Chain and Polygon

Website

<https://www.magicfox.fi/>

Telegram

<https://t.me/magicfoxfi>

Twitter

<https://twitter.com/magicfoxfi>

Discord

<https://go.magicfox.fi/join-discord>



Description

Swap, Optimizer & business-to-business solution. Invest and build passive income.

Project Engagement

During the 12th of April 2023, **MagicFoxFi Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository.

Logo



Contract Links

v1.0

<https://github.com/magicfoxfi/contracts/tree/main/swap>

Commit: [5a0dba2](#)

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. 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 evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
 - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
 - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Router.sol

```
./libraries/Math.sol  
./interfaces/IERC20.sol  
./interfaces/IPair.sol  
./interfaces/IPairFactory.sol  
./interfaces/IRouter.sol  
./interfaces/IWETH.sol
```

Pair.sol

```
./libraries/Math.sol  
./interfaces/IERC20.sol  
./interfaces/IPair.sol  
./interfaces/IPairCallee.sol  
./PairFactory.sol  
./PairFees.sol
```


Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

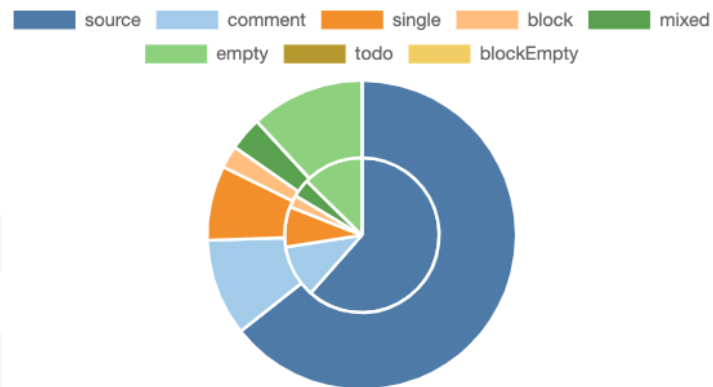
File Name	SHA-1 Hash
contracts/interfaces/IPair.sol	05f5e2ed078a44a178b81efb91a2ef5fa0b41b95
contracts/interfaces/IMinter.sol	e046cb34d960eab81f4e0fa3d949bc87d8a7cbe2
contracts/interfaces/IRouter.sol	f7549cc63d7a9f5825fa0339ccd1d5dd8d887b15
contracts/interfaces/ IPairCallee.sol	91e71b132ac89f61a0704b775e5f4c2996a30e70
contracts/interfaces/ IRewardsDistributor.sol	38673e379e9a8242a3b611899f14323e3c943daa
contracts/interfaces/IWETH.sol	6bca1338ef4ec79bad9e52ce7a9659f3d09e9de3
contracts/interfaces/ IPairFactory.sol	9a5df04dbfff7d804eb2bb866eb3a848627dea56
contracts/interfaces/IERC20.sol	871201f0e54485d806ecd22897a1c004237e1509
contracts/Router.sol	726246d64257e33d892a550abb3243397e6f002f
contracts/PairFees.sol	e562c32e4195a602dec4acf593a932acb163e62e
contracts/libraries/Base64.sol	8047d95ed49bdcc5962b759575092beb30887727

contracts/libraries/Math.sol	2ed659578b56f6a7ec9e6bada577018c60e426b7
contracts/libraries/ SignedSafeMath.sol	8b0cd9feef40dde8dcb671d8fb3ad58aeeb03420
contracts/PairFactory.sol	0d7d0719f75f310f931b59873dd1aa49c5ab7eb9
contracts/Pair.sol	d08316961137ab474ee6042d11b88e0c7362a51d

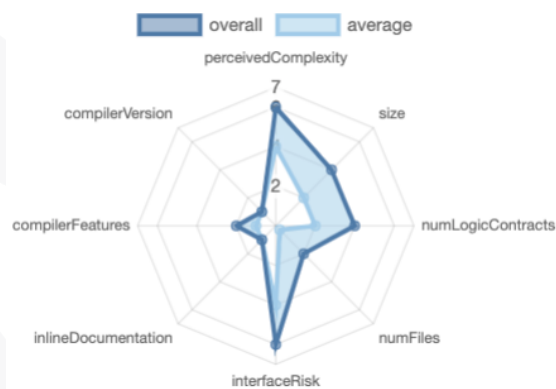


Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
4	3	8	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
104	4







External	Internal	Private	Pure	View
93	98	0	17	51



StateVariables

Total	 Public
61	43

Capabilities

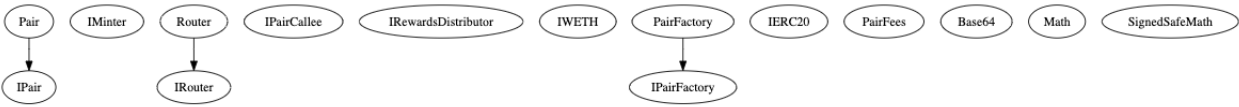
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.13		yes	yes (1 asm blocks)	

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
yes			yes	yes	yes → NewContract:PairFees

 TryCatch	 Unchecked
	yes

Inheritance Graph

v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Overall checkup (Smart Contract Security)



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

Modifiers and public functions

v1.0

PairFactory

```
setPause  
setFeeManager  
acceptFeeManager  
setOwnerFee  
setOwnerFeeAddress  
setPartner  
setFee  
setDegenFee  
createPair
```

Note:

- General fork from THENA - <https://github.com/ThenafiBNB/THENA-Contracts>
- Contracts inside are the same as the THENA contracts and Differences between MagicFoxFi and THENA contracts are the following:
 - PairFactory
 - PairFactory has an extra 'degen fee' functionality, and the stakingNFT fee is replaced by owner fee. Referral fee is replaced by Partner fee.
 - Pauser is replaced by the Fee manager address that can pause the contract
 - Dibs address is replaced by partner, and lp address
 - Pair
 - Pair contract has an extra degen functionality that will take affect at the time of getting the fee while swapping. The fee manager address can set the degen value to either true or false
 - Staking fee is replaced by owner fee
 - Referral fee is replaced by partner fee
 - PairFees
 - Staking fee is replaced by owner fee
 - Router

- The router contract is identical as of THENA

Ownership/Authority Privileges

- [PairFactory.sol](#)
 - The fee manager address can pause/unpause the contract
 - Set a new fee manager address
 - Set Owner fee percentage and address
 - Set partner address and fee. Max partner fee can be set up to 50% so beware about it
 - Set degen fee, but not more than 1%

Please check if an `OnlyOwner` or similar restrictive modifier has been forgotten.

Source Units in Scope v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/IPair.sol	—————	1	25	5	3	1	35
contracts/interfaces/IMinter.sol	—————	1	9	5	3	1	9
contracts/interfaces/IRouter.sol	—————	1	6	5	3	1	3
contracts/interfaces/IPairCallee.sol	—————	1	6	5	3	1	3
contracts/interfaces/IRewardsDistributor.sol	—————	1	9	5	3	1	9
contracts/interfaces/IWETH.sol	—————	1	8	5	3	1	10
contracts/interfaces/IPairFactory.sol	—————	1	11	5	3	1	13
contracts/interfaces/IERC20.sol	—————	1	16	5	3	1	17
contracts/Router.sol	1	—————	412	320	267	19	300
contracts/PairFees.sol	1	—————	61	61	45	6	30
contracts/libraries/Base64.sol	1	—————	63	63	41	8	167
contracts/libraries/Math.sol	1	—————	35	35	33	1	12
contracts/libraries/SignedSafeMath.sol	1	—————	93	93	33	46	11
contracts/PairFactory.sol	1	—————	142	142	116	16	95
contracts/Pair.sol	1	—————	591	591	458	78	371
Totals	7	8	1487	1345	1017	182	1085

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	PairFactory.sol	Missing Zero Address Validation (missing-zero-check)	62	Check that the address is not zero
#2	PairFactory.sol	Missing Events Arithmetic	57-106	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	PairFactory.sol	State variables that could be declared constant (constable-states)	20	Add the `constant` attributes to state variables that never change
#3	PairFees.sol	Error message is missing	23, 48, 36	Provide an error message for require statement
#4	PairFactory.sol	Error message is missing	58, 109	Provide an error message for require statement

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

15. April 2023:

- This project consists of the following forks
 - THENA
- Read whole report and modifiers section for more information
- The low issues that exist in the THENA codebase still exist in the forked code.
- We cannot make sure that the logic and the contracts are 100% correct because the MagicFoxFi Team didn't provide us a white paper.
- We recommend using a multisig wallet for the owner address to prevent any risk of the loss of private key
- Do your own research here

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

*Solid
Proofed*

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


MADE IN GERMANY