SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Piteas

AUDIT

SECURITY ASSESSMENT

08. July, 2023

FOR

✳ piteas

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	Piteas
Website	https://piteas.io
About the project	Piteas serves as a dex aggregator that leverages the latest in swap technology. By integrating multiple DEXs, our platform provides users with access to the vast pools of the most widely used DeFi protocols, amounting to billions of dollars in liquidity. With our advanced routing algorithm, Piteas ensures the best price, lowest slippage, and highest returns for each transaction.
Chain	Pulsechain
Language	Solidity
Codebase Link	https://scan.pulsechain.com/address/0x6BF228eb7F8ad948d37deD07E595EfddfaAF88A6 https://scan.pulsechain.com/address/0x9857686e49C006C8A14c0B722891078165AF848c
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/piteasio
Twitter	https://twitter.com/piteasio
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	05. July 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	08. July 2023	• Reaudit and Router Address Update

Note - This Audit report consists of a security analysis of the **Piteas** smart contracts. 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/interfaces/IPermit2.sol	77cc6083e77b9c18bd62a335237 bd25403de1248
contracts/interfaces/ ISwapManager.sol	ec152c1039443ab8c8fc785211cb 5d78ce996502
contracts/interfaces/ IDaiLikePermit.sol	b351e00bfe3f1571557b887098dc 3213632378ec
contracts/interfaces/IWETH.sol	8e9bed5da17ec2aa0a21446edbc 3624f67428182
contracts/interfaces/ IERC20MetadataUppercase.sol	96f2267ee8de278efc9766a8e785 0d764d8f6113
contracts/EthReceiver.sol	6f991c9f0e1205c1e0790992644a 996842ec3d7d
contracts/errors/Errors.sol	0c39690330176e381332eec2de7 381154254161a
contracts/swapper/interfaces/ IUniswapV3.sol	a72bd49d98bf0f9984219228b7a e73e02b794cb8
contracts/swapper/interfaces/ IUniswapV2.sol	c77b6e07d01e91fa12f1645da688 5ffe352c58a5
contracts/swapper/interfaces/ IBalancer.sol	f050c150ee3cbc48b1453530d6a d58b89af00b7b
contracts/swapper/interfaces/ ISwapper.sol	bc404b2f19d85c5553244d4fc232 6153ae7777ee
contracts/swapper/interfaces/ IWrapper.sol	4ee9033edcddafab71c05f468cfc 2e8ad8f09e90
contracts/swapper/interfaces/ ISolidly.sol	4fd98b31551082a87097dd5bb7f d2e0b8f0fe42d

File	Hash
contracts/swapper/SwapManager.sol	fd96dbdd0c64c71d79554a10a610de5c37e6335c
contracts/swapper/SwapLibrary.sol	8bb8c1ea6e159800a08518ae2fa9f978397f04f5
contracts/libraries/RevertReasonForwarder.sol	da3ebd8df0b8f10b806cc3af5380e8090d1b0a2c
contracts/libraries/ExcessivelySafeCall.sol	94298f31e53b59d626940c6015e65136419f9572
contracts/libraries/RevertReasonParser.sol	c8ea196737a5141c0420f3788cec1837bb2f9e53
contracts/libraries/PitERC20.sol	19c09afefcbdf2c62b2ac7c736dc0ce6339a450f
contracts/libraries/SafeERC20.sol	e73e441460b060cae8a63ce5de50b86f69d44b2c
contracts/libraries/StringUtil.sol	f6f82842d6709163025ab7a3788b1484a964ba3b
contracts/PiteasRouter.sol	c9b56fa40e2b3c23367ac899e68e73bb564895dc

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.

Externally Imported packages
Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
hardhat/console.sol	1

Note for Investors: We only Audited swapper, router, and token contracts for **Piteas**. However, If the project has other contracts (for example, a Presale or a staking contract 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 aspossible.
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	N/A

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	The contracts are not upgradeable directly by the proxy method but it is possible for the owner to change the router address.

Ownership

The ownership is not renounced	❌ The ownership is still present
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: · Centralizations · The owner has significant control over contract's operations
Example	We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.
Comment	N/A

Note - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities then the ownership is automatically considered renounced.

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	There is no minting functionality

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 contract's cost, 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 25%	✅ The owner cannot levy unfair taxes
Description	The owner is not able to set the fees above 25%
Comment	There is no Fee functionality

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 can lock the contract	❌ The owner can lock the contract
Description	The owner is able able to lock the contract by change status function.
Comment	The owner is able to pause/unpause the swap functionality in the router contract

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
3	7	11	1

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
29	7

External	Internal	Private	Pure	View
25	84	6	7	8

StateVariables

Total	🌐 Public
19	0

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.0` `^0.8.18` `>=0.7.6`	————	YES	YES	————

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.
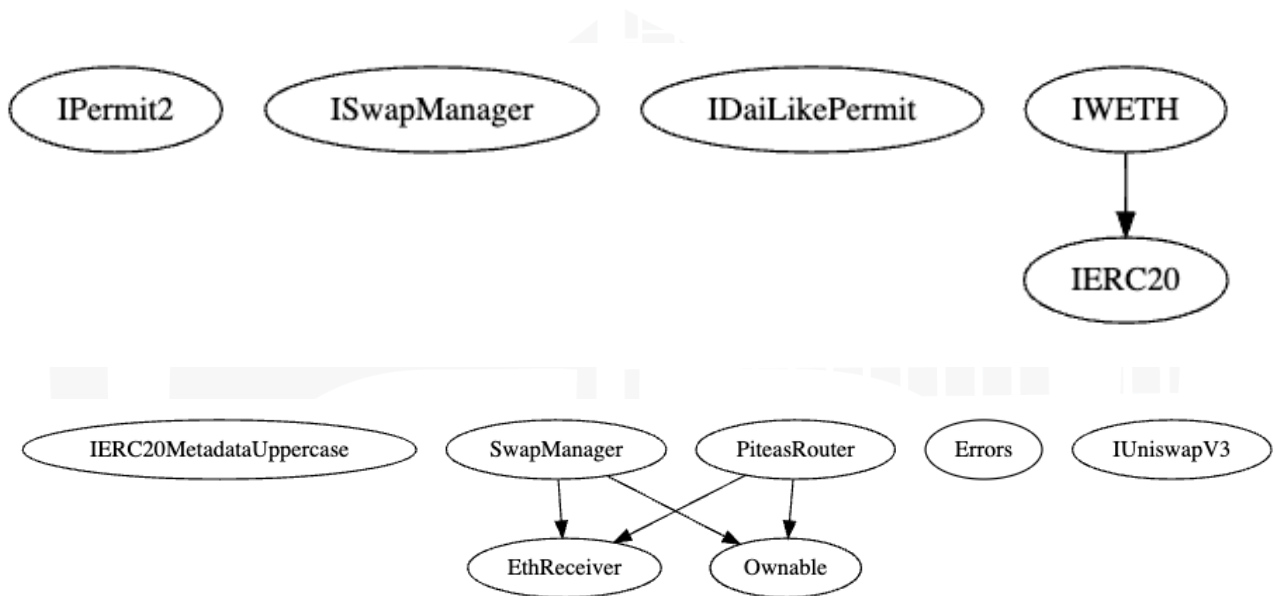
Centralization Privileges

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. PiteasRouter.sol	❖ onlyOwner – Change swap manager address – Withdraw any token from the contract – Pause/Unpause the swap function in the contract
2. SwapManager.sol	❖ onlyOwner – Update Exchange Type – Update Protocol and router Address – Withdraw any token from the contract

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 client 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 can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.

Audit Results

#1 | Missing Zero Address Validation

File	Severity	Location	Status
SwapManager	**Low**	L70, 75, 81	**Fixed**

Description

- Make sure to validate that the address passed in the function parameters is "non-zero".

#2 | NatSpec documentation missing

File	Severity	Location	Status
All	**Informational**	—	**ACK**

Description

- If you started to comment on your code, also comment on all other functions, variables etc.

#3 | Unused State Variable

File	Severity	Location	Status
PiteasRouter	**Informational**	L115	**Fixed**

Description

- Remove the unused state variables or implement them in the contract.

#4 | Floating Pragma

File	Severity	Location	Status
Main	**Informational**	L2	**ACK**

Description

- The current pragma Solidity directive is "^0.8.18". Contracts should be deployed with the same compiler version and flag that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions

#5 | 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.

Solid Proofed

MADE IN GERMANY

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY