SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Citadel Swap

AUDIT

SECURITY ASSESSMENT

26. September, 2023

FOR

SolidProof_io

@solidproof_io

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	Citadel Swap
Website	https://www.citadelswap.xyz/
About the project	Citadel is a custom-built infrastructure, composed of a highly efficient and hyper-functional DEX that is designed to support the Base ecosystem by building a flexible and sustainable liquidity strategy, and a Launchpad, built to support new protocols launching on Base by providing the tools necessary for their launch, liquidity creation, and growth.
Chain	Base Scan
Language	Solidity
Codebase Link	https://github.com/citadelswap/contracts/tree/main
Commit	d3d0059
Unit Tests	Not Provided
Forked Stauts	1:1 Fork of the Following: AMM contracts are forked from Camelot DEX — https://docs.camelot.exchange/contracts/amm-v2 Other Contracts are forked from MMF Finance — https://mmfinance.gitbook.io/arbitrum/contract-and-security/contracts

Social Medias

Telegram	https://t.me/citadelswap
Twitter	https://twitter.com/citadelswap
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@CitadelSwap
Discord	https://discord.gg/kfZ3GuvtWy
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	18. September 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	26. September 2023	• Reaudit

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it.

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/TOKENS/esToken.sol	26768fd470880eaebc4f585167e1484030c80c3c
contracts/TOKENS/token.sol	eccd1cfb6cf2caafdd0e8511eabef5492fb043e6
contracts/YIELD FARMING/proxy.sol	7421c7bc85fe2fa08e64851846b2ddd1000015a6
contracts/YIELD FARMING/master.sol	8b1c1b62cafb50a0f6bd7516b46e798768309705
contracts/YIELD FARMING/utils/AddressUpgradeable.sol	c8939d52cd5e15f93e1b14fb3bafcc727630a637
contracts/YIELD FARMING/utils/SafeBEP20.sol	c7feb7cd370ef8321a271d9c50827d474cf30b56
contracts/YIELD FARMING/utils/ContextUpgradeable.sol	0b9573383d939289977f200120392dacd629917f
contracts/YIELD FARMING/utils/Address.sol	58cc6e8fad92ee5b7cef524a1ef94677fe23eafe
contracts/YIELD FARMING/utils/ReentrancyGuard.sol	e87bbb6ad353ea74faace51953364cfd3edd0ede
contracts/YIELD FARMING/library/Math.sol	44f2973c0cb39f3a7e46582fea2b6c238ea796d3
contracts/YIELD FARMING/library/Whitelist.sol	54fd14495a2a50ab87777ff3719975fe149105e8
contracts/YIELD FARMING/library/SafeMath.sol	1a7688732b0260aacdd11cc2c2e4230118229e4e
contracts/YIELD FARMING/library/WhitelistUpgradeable.sol	c7f50d9a9dfd3bd6ee6b40a446aa1946e8891d08
contracts/YIELD FARMING/access/Context.sol	6734a8153c1738efdb6f809c5c30687ad6d2af4c

contracts/YIELD FARMING/access/OwnableUpgradeable.sol	7e38d3d80fa042e58c3415b63973fff3c44fb821
contracts/YIELD FARMING/access/Ownable.sol	cb0a5ddfdf519ef6b42c6d495e8a649e3f20dd2e
contracts/YIELD FARMING/proxy/Initializable.sol	f499d19ef9ec2471f75802c169d5e5a861fb93a4
contracts/DEX/router/router.sol	212744eae7c8efaa7226fe09e11f7a11404eefdc
contracts/DEX/router/libraries/SafeMath.sol	4a7f761df9b656a1f64a99ff7d6fbcdc813f54b6
contracts/DEX/router/libraries/UniswapV2Library.sol	a0842a47968735f09712ebfeeee89788580b08a4
contracts/DEX/factory/libraries/Math.sol	1348a437cf092bb525e65d5b825ce19e5024ca26
contracts/DEX/factory/libraries/IFactory.sol	729f26d7f77771ce86d8c0f7734f0b195a587c58
contracts/DEX/factory/libraries/SafeMath.sol	4a7f761df9b656a1f64a99ff7d6fbcdc813f54b6
contracts/DEX/factory/pair.sol	c292a60c455ea8ccbdaa8f99f9d30f74221777b0
contracts/DEX/factory/factory.sol	4f8e48be5c7ffdcbfda58a5f274efa61ed0eeb60
contracts/DEX/factory/UniswapV2ERC20.sol	44eb8c8203b1db1a9db3cfdd4de8d45981c052d4

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. Moreover, the external interfaces used in the contracts were not a part of the audit.

Imported packages

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

N/A

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and 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

Upgradeability

Contract is an upgradeable	❌ Deployer can update the contract with new functionalities
Description	The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.
Comment	The ‘proxy’ Contract is upgradeable of nature but other contracts can only be deployed once and cannot be updated.

Ownership

The ownership is not renounced	✗ The owner is not renounce
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
Comment	Once the ownership is renounce, the Medium issue and the Minting Red Flag will be fixed

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

Ownership Privileges

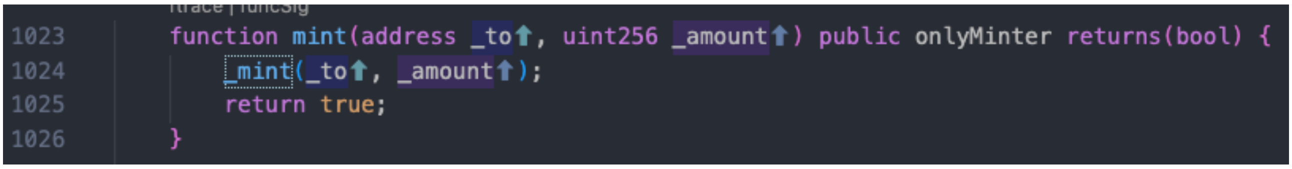
These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

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 can mint new tokens	❌ **The owner able to mint new tokens**
Description	Owners who have the ability to mint new tokens can reward themselves or other stakeholders, who can then sell the newly minted tokens on a cryptocurrency exchange to raise funds. However, there is a risk that the owner may abuse this power, leading to a decrease in trust and credibility in the project or platform. If stakeholders perceive that the owner is using their power to mint new tokens unfairly or without transparency, it can result in decreased demand for the token and a reduction in its value.
Example	If the owner is not transparent and honest about their actions, they may be attempting a rugpull, where they suddenly abandon the project after raising funds, leaving investors with worthless tokens. This can lead to a decrease in the value of existing tokens, potentially rendering them worthless, and causing investors to suffer losses. It is essential for investors to carefully research the project and its developers and exercise caution before investing in any cryptocurrency or DeFi project.
Comment	The owner can add minter addresses and they can mint unlimited tokens. However, when the ownership will be renounced and there are no addresses that can mint token then the minting will not be possible

File, Line/s: esToken.sol and token.sol



```
function mint(address _to, uint256 _amount) public onlyMinter returns(bool) {
    _mint(_to, _amount);
    return true;
}
```

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	N/A

Lock User Funds

In a smart contract, locking refers to restricting access to certain tokens or assets for a specified period. 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
18	15	6	4

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
183	3

External	Internal	Private	Pure	View
92	326	21	49	110

StateVariables

Total	🌐Public
94	54

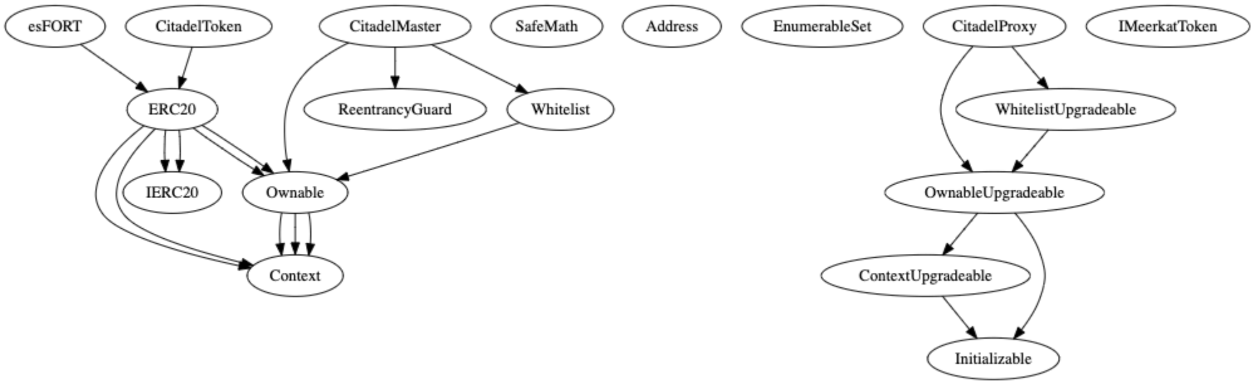
Capabilities

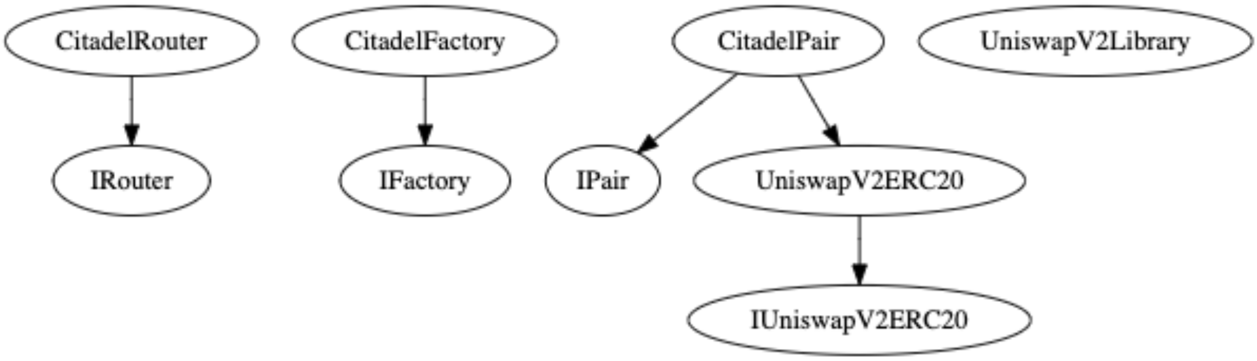
Solidity Versions observed	📤 Transfers ETH	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

`0.6.12`
`>=0.6.12`
`^0.6.0`
`>=0.6.0`
`<0.8.0`
`^0.6.2`
`^0.6.12`
`>=0.4.0`
`>=0.4.24`
`<0.8.0`
`=0.6.6`
`=0.5.16`
`>=0.5.0`
`>=0.5.16`
	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 a single entity controls the contract or if certain participants have special permissions or abilities that others do not.

In the project, some authorities have access to the following functions:

File	Privileges
1. master.sol	• onlyOwner • Add LP to the pool • Set Allocation point for a given pool • Update emission rate to any arbitrary value • Set Nft controller, Gauge Controller, and Meerkat Referral addresses • Set NFT Boost Rate • Enable/Disable Whitelist • Set Proxy Address • Set Comission and Unlock Rate
2. factory.sol	• onlyOwner • Transfer Ownership • Set fee receiver address • Set Stable Coin owner address • Set owner fee share • Set referrer fee share upto 20% maximum
3. pair.sol	• The factory contract owner can set the following • Set pair type as Immutable • Withdraw foreign tokens from the contract's balance • The fee percent owner address can change the fee percent upto 2% • The stable owner of the factory contract can set the stable swap

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 cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

#1 | Owner can mint tokens

File	Severity	Location	Status
token	Medium	L1023	ACK
esToken	Medium	L1019	ACK

Description - The owner can add/remove any wallets from the minter's list, and these minter addresses have the capability to mint tokens till the max supply is reached

Remediation - We recommend limiting the minting so that the owners cannot exploit it for personal gain.

Note for Investors - Once the contract is renounced and there are no minter addresses, the owner will not make the minting possible.

Low issues

#1 | Missing Events

File	Severity	Location	Status
Master	Low	L398—416	ACK
Token	Low	L1032, 1037	ACK
esToken	Low	L1039, 1044	ACK

Description - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes.

#2 | Old Compiler version

File	Severity	Location	Status
All	Low	N/A	ACK

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

Remediation - Use a newer pragma version. At least use the 0.8.18 version.

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
Proxy	Informational	N/A	ACK

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

#2 | Disable initializing

File	Severity	Location	Status
Proxy	Informational	L13	ACK

Description
If the owner updates the contract, a disableInitializer call in the constructor must be implemented. This prevents calling the initialize function again to set the state variables in the contract. This should be implemented only if the contract was deployed before. Otherwise, the owner cannot call the initialize function to set the variables.

Recommendation
If the contract hasn't been deployed, remove the disableInitializer in the constructor. Otherwise, you are not able to initialize the contract. When the contract has a deployed version already, leave it as it is.

#3 | Floating Pragma

File	Severity	Location	Status
Master	Informational	N/A	ACK

Description - The 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.

#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.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY