



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

Perfect Swap

AUDIT

SECURITY ASSESSMENT

19. March, 2024

FOR



SolidProof_io



@solidproof_io

Introduction	4
Disclaimer	4
Project Overview	5
Summary	5
Social Media	5
Audit Summary	6
File Overview	7
Imported packages	8
Audit Information	9
Vulnerability & Risk Level	9
Auditing Strategy and Techniques Applied	10
Methodology	10
Overall Security	11
Upgradeability	11
Ownership	12
Ownership Privileges	13
Minting tokens	13
Burning tokens	14
Blacklist addresses	15
Fees and Tax	16
Lock User Funds	17
Components	18
Exposed Functions	18
StateVariables	18
Capabilities	19
Inheritance Graph	20
Centralization Privileges	21
Audit Results	22
Critical issues	22
High issues	22



Medium issues	22
Low issues	23
Informational issues	23



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	Perfect Swap
Website	https://perfectswap.io/
About the project	The foundation of PerfectSwap can be traced back to Solidly V2, which serves as the cornerstone of the ve(3,3) model. Solidly V2 introduced a groundbreaking incentive model that addressed the longstanding need for a logical and robust liquidity framework within decentralized exchanges (DEXs). However, even with the promise of ve(3,3) DEXs, there remained significant challenges to overcome regarding efficiency and sustainability.
Chain	Arbitrum
Language	Solidity
Codebase Link	PRFCT 0xA108a69A996d5467444780e2B1b36aD5851de2Ef vePRFCT 0xE000B6f5E6Bbf57A446e89F6CfF296eA589CF754 poolFactory 0xC7ee0B06c2d9c97589bEa593c6E9F6965451Fe93 Voter 0x55b788359aA4f98Ec3805aFa1778Aa85b32A821b router 0xf93F88CC55D27D2c4d72BB0c785e90b5550C3d69 minter 0x5A4c4BF6b71164C2b404A252DEF5FFbb9096Ceda managedRewardsFactory 0x13cAAC57257ffeC41d091d7B3e3d16b7E8D87ebE RewardsDistributor 0x7E1901705CCb2f7bC65576501851939d2964d8c3
Forked Status	<p>The contracts are Forked from VelodromeV2 and Aerodrome Finance contracts which can be found at:</p> <p>https://velodrome.finance/security#contracts</p> <p>https://github.com/aerodrome-finance/contracts/tree/main/contracts</p>
Unit Tests	Not Provided

Telegram	N/A
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.gg/perfectswap
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	19. March 2024	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. 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. This includes internal calculations in the formulae used in the contract.



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/poolFactory.sol	2b8be3a9c062d4ecb6cd181880075a1f08cd9948
contracts/Minter.sol	167cbac5fd9036dd94d64bfdefa3bb5d7403b24a
contracts/RewardsDistributor.sol	60bab1358e211be014f3e2f7b08b252efcfb89bf
contracts/Router.sol	1b180ec32fe31bb89dad3120847d43740f0223bf
contracts/Voter.sol	f97f260016777d661d0b11213b6f8037a4c1272a
contracts/ManagedRewardsFactory.sol	f70fe978e7694d8e382ba9647fc0c55f6a253406
contracts/Prfct.sol	7cc96ba91b17f30220ded9529996c16a56f4c122
contracts/VotingEscrow.sol	52d3bfd19307bf07351c53c7a799692d47ae21dc

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).

Dependency / Import Path	Count
@openzeppelin/contracts/metatx/ERC2771Context.sol	2
@openzeppelin/contracts/proxy/Clones.sol	2
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	4
@openzeppelin/contracts/utils/math/Math.sol	4

Note for Investors: We only audited contracts mentioned in the scope above. Apart from that, all contracts related to the project are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way. Moreover, other libraries or smart contracts used in the code that are not mentioned in the scope were not a part of the audit.

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

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 Directly Upgradeable but the Minter address can be changed in the Rewards Distributor Contract

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 some control over contract's operations

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

Minting Will be Done Automatically in the Smart Contracts



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 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 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 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
8	5	15	3


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
328	5

External	Internal	Private	Pure	View
308	288	5	19	172

StateVariables

Total	 Public
142	104



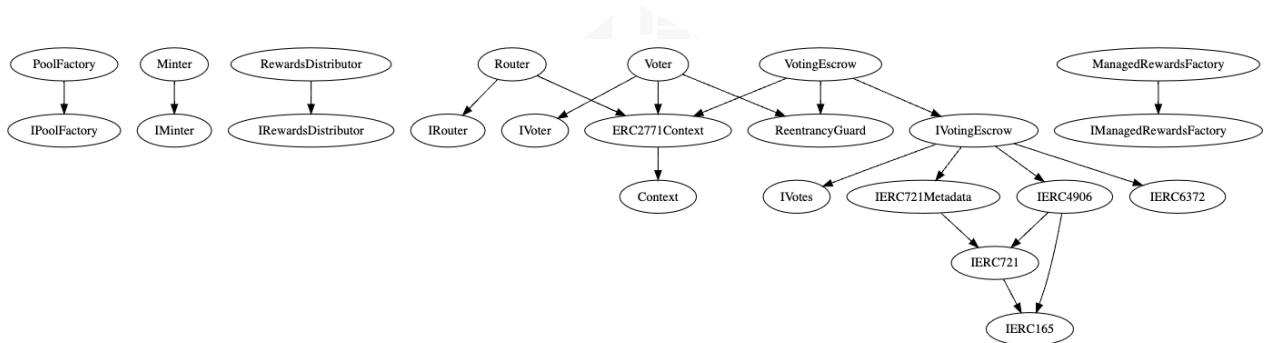
Capabilities

Solidity Versions observed	Transfers ETH	Can Receive Funds	Delegate Call	ECRecover
0.8.19	Yes	Yes	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
Voter	<ul style="list-style-type: none"> • Add/Remove tokens from Whitelist • Kill/Revive gauge • Set Governor, Emergency Council Addresses • Set Maximum Voting Numbers
PoolFactory	<ul style="list-style-type: none"> • Set Voter, Pauser and Fee Manager Address • Pause/Unpause contract • Set Fee
VotingEscrow	<ul style="list-style-type: none"> • Set Allowed Manager and Managed State • Set Team Address

Recommendations

To avoid potential hacking risks, the client should 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

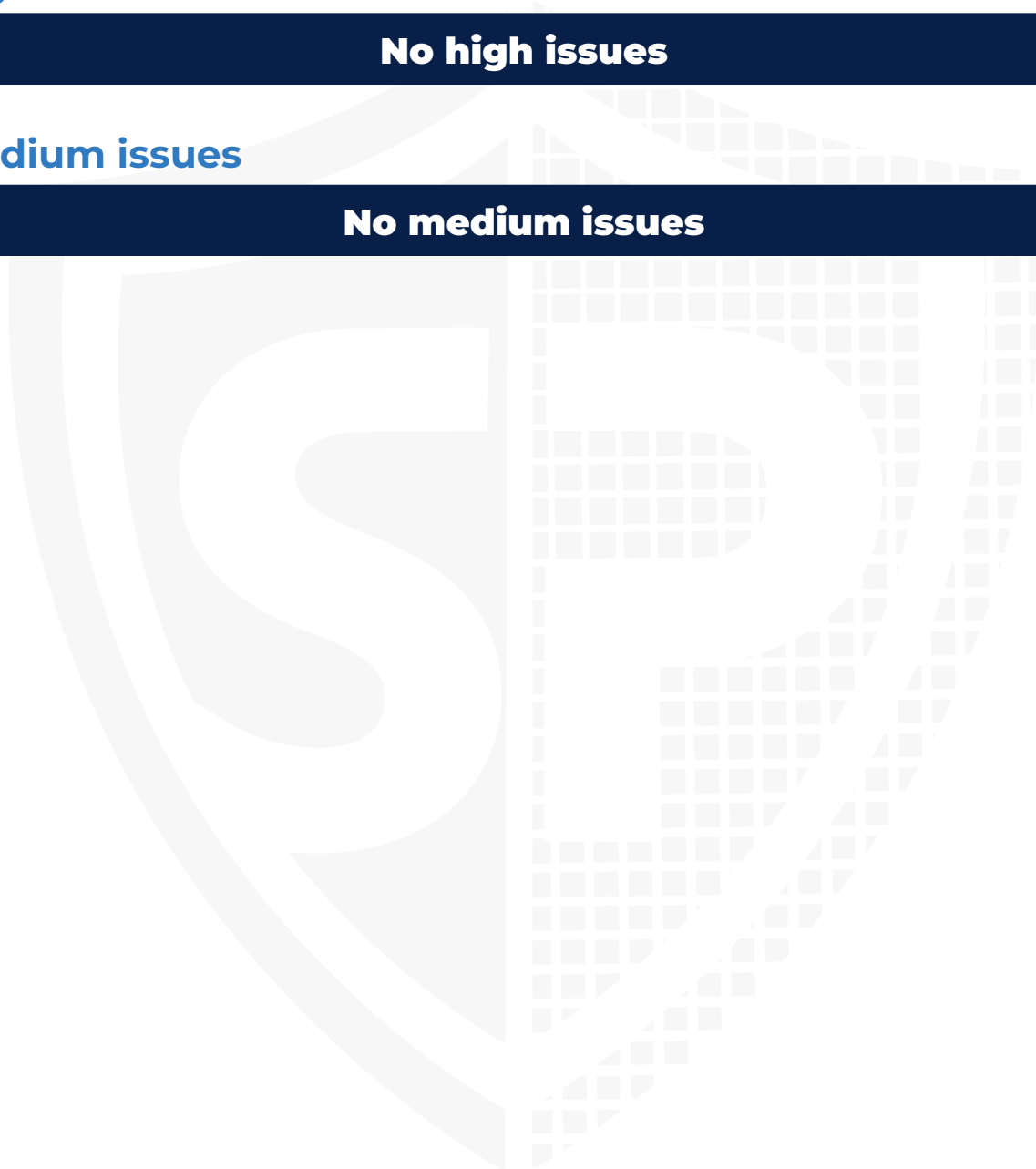
No critical issues

High issues

No high issues

Medium issues

No medium issues



Low issues

#1 | Missing Events

File	Severity	Location	Status
VotingEscrow	Low	L2525, 2542	ACK
Minter	Low	L123, 138	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.

Informational issues

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

File	Severity	Location	Status
VotingEscrow	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.



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

MADE IN GERMANY