SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Wagmipad

AUDIT
SECURITY ASSESSMENT

04. February, 2025

FOR

SOLIDProof

Introduction

SolidProof.io is a brand of the officially registered company Future Visions Deutschland. We're mainly focused on Blockchain Security, such as Smart Contract Audits and KYC verification for project teams.

Solidproof.io assesses potential security issues in the smart contracts implementations, reviews for potential inconsistencies between the code base and the whitepaper/documentation, and provides suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should they be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should they be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io does not cover testing or auditing the integration with external contracts 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 analysed, 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 analyse.

Project Overview

Summary

Project Name	Wagmipad
Website	https://wagmipad.org/
About the project	Wagmipad introduces a revolutionary method for conducting token sales, aimed at delivering a secure and trustworthy investment environment. It combats fraud with cutting-edge strategies, directly addressing the challenges of ineffective marketing and investor fears related to scams. Through these pioneering approaches to raising funds, Wagmipad proactively confronts and solves these prevalent issues.
Chain	TBA
Language	Solidity
Codebase Link	Provided as files.
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://x.com/wagmipad
Twitter	https://x.com/wagmipad
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.com/invite/wagmipad
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	11. December 2024	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	28. January 2025	• Reaudit

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes malicious outside manipulation of the contract's functions. 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
LaunchPadVesting.sol	f86882f482cee70b3df8b1c0b228f4ce4bfbb059
RewardsDistributionRecipient.sol	e2a2994354b585391a3b96ae6dd85289183af076
LaunchPad.sol	d70953fde2d7b4a8f22f8b19c7f3071a10192872
LaunchPadSales.sol	93b5a14e14a6ac0e3ddf81afa069daabe66ca96a
StakingRewards.sol	97f5ee2bae6bc55341dc3c5730fd0cc418d67a59
LaunchPadData.sol	ddfe900fe53ca2ff13438988dd5bca83fe8471a1
StakingRewardsFactory.sol	fe843c25768f74cfe8d2d8b78e1adc94dc1c60d0

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) indicate a changed state or potential vulnerability that was not the subject of this scan. Also, the mentioned contracts are not deployed onto mainnet. Hence, any further changes will be considered as out of scope for the audit. Therefore, It is recommended to do the research before investing.

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

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol	2
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1
@openzeppelin/contracts/utils/Pausable.sol	1
@openzeppelin/contracts/utils/ReentrancyGuard.sol	1
@openzeppelin/contracts/utils/math/Math.sol	1

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. Review the specifications, sources, and instructions provided to SolidProof to ensure we understand the smart contract's size, scope, and functionality.
   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 is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., 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.
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

Ownership

Contract ownership is not renounced	❌ The ownership is not renounced
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 refers 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 can 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 without Allowance

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 to burn tokens without any allowances.
Comment	N/A
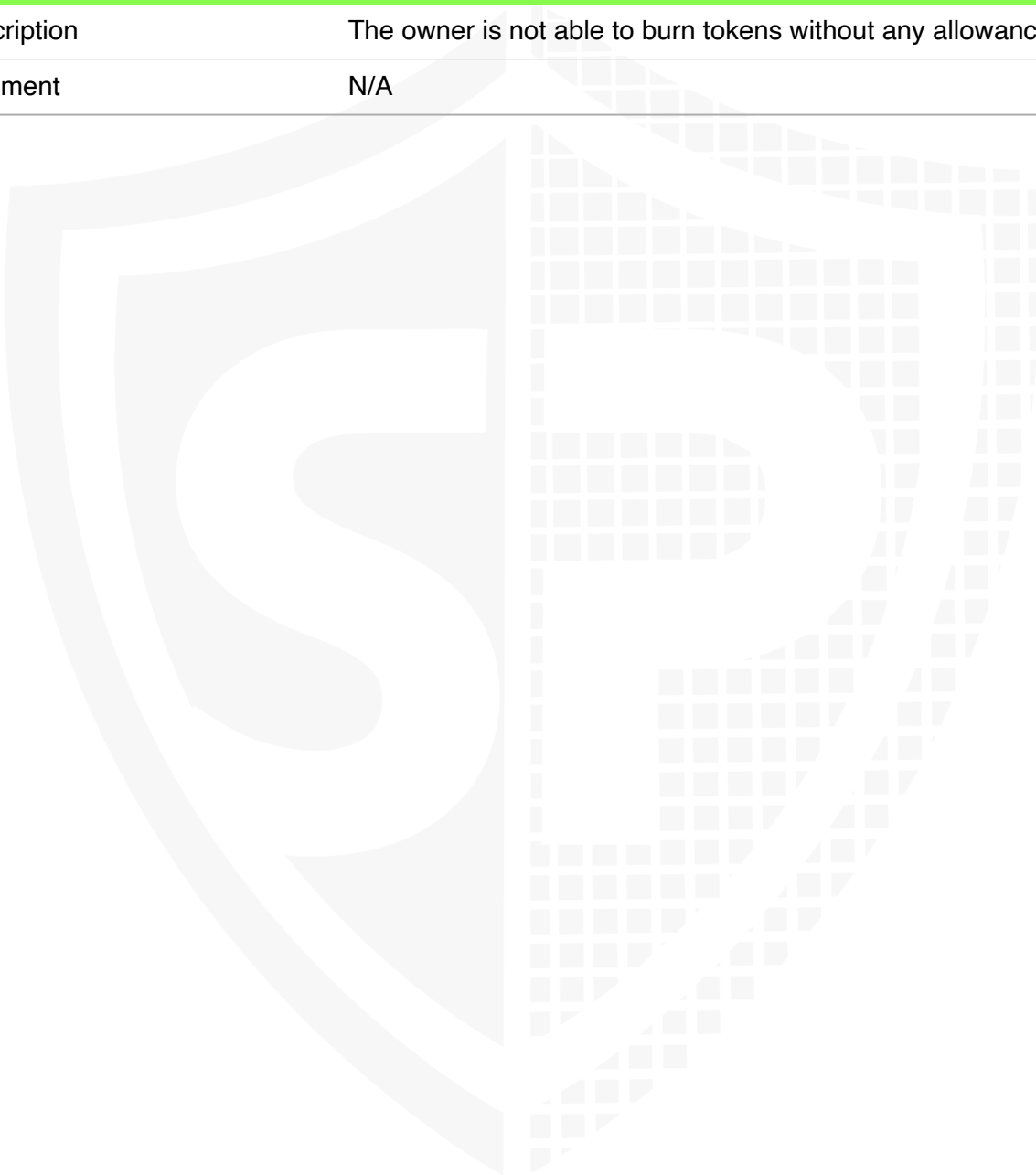
Blacklist addresses

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 to 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 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.

Contract owner can lock the contract	❌ The owner is able to lock the contract
Description	Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.
Example	The contract contains the passable functionality in which the owner of the contract can pause the withdraw functionality from the contract for an indefinite period of time.
Comment	N/A

File, Line/s: L204-206
Codebase: StakingRewards.sol

```solidity
function pause() public onlyOwner {
    _pause();
}
```

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
6	0	0	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
49	0

External	Internal	Private	Pure	View
22	46	3	0	28

StateVariables

Total	🌐 Public
36	33

Capabilities

Solidity Versions observed	Transfers ETH	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
0.8.19 ^0.8.20 ^0.8.19				

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🧨 ECRecover	🌀 New/Create/Create2
yes			yes		yes → NewContract:Staking Rewards

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.
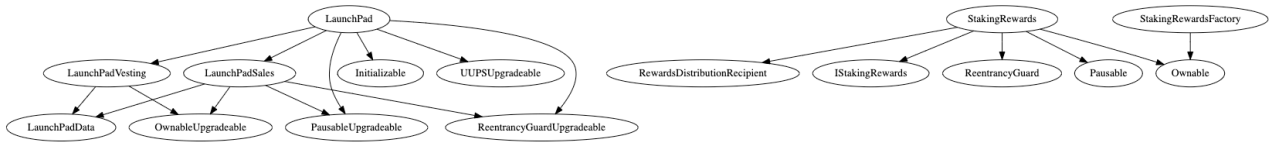
Centralization Privileges

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
LaunchPadSales.sol	• The owner can create a new sale. • The owner can withdraw the tokens from the contract. • The owner can update the project sale end time. • The owner can update the claim start time. • The owner can update the signer address.The owner can update the vesting schedule for the project.The owner can update the required staking per tier value.The owner can update the staking contract address.
LaunchPadVesting.sol	• The owner can update the vesting schedule for the project. • The owner can update the required staking per tier value. • The owner can update the staking contract address.
LaunchPad.sol	• The deployer can initialize the contract.
StakingRewards.sol	• The owner can update the cooling period. • The owner can pause/unpause the staking and unstacking functionalities.
StakingRewardsFactory.sol	• The owner can deploy a new version of the contract.

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

Audit Results

Critical issues

No critical issues

High issues

#1 | The owner can lock claim

File	Severity	Location	Status
StakingRewards.sol	High	L204-206	ACK

Description - The contract includes a pausable functionality that allows the owner to pause the claim and withdrawal processes. However, the absence of a defined limit on the pause duration poses a risk, as it could result in user funds being locked indefinitely. To mitigate this, a maximum locking period should be enforced, ensuring that token claims and withdrawals cannot be paused indefinitely, thus protecting users' access to their funds.

Medium issues

#1 | The owner can withdraw claimable tokens.

File	Severity	Location	Status
LaunchPadSales.sol	Medium	L141-161	ACK

Description - The emergencyWithdraw function in its current implementation allows the owner to withdraw all sale tokens and payment tokens associated with a specific project. However, this could inadvertently claim tokens that are designated as claimable by users, effectively locking the buy and claim functionality for participants. This creates a significant risk of funds being inaccessible to users and disrupts the expected functionality of the contract.

Remediation - To mitigate this, Add a mechanism to validate that tokens eligible for user claims are not affected by the withdrawal. Moreover, Maintain a clear distinction between tokens reserved for users and those under the owner's control.

Low issues

No low issues

24

#1 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	—	ACK

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

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