



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

Gecko

AUDIT

SECURITY ASSESSMENT

26. February, 2024

FOR



SolidProof_io



@solidproof_io

Introduction	4
Disclaimer	4
Project Overview	5
Summary	5
Social Medias	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	23
Critical issues	23
High issues	23



Medium issues	24
Low issues	25
Informational issues	26





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	Gecko Inu Staking
Website	https://geckoinu.vip/
About the project	<p>Gecko Inu, the heart of AVAX's community tokens! A meme coin that's all about friendship and fun.</p> <p>Dive into the vibrant web3.0 world with Gecko Inu and experience a community-driven adventure.</p> <p>Gecko Inu not just a meme coin; it's a celebration of togetherness and laughter in the Avalanche ecosystem. Life's truly richer with a sprinkle of Gecko Inu spirit!</p>
Chain	AVAX Chain
Language	Solidity
Codebase Link	Prived as Files
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/geckoinuavax
Twitter	https://twitter.com/GeckoInuAvax
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	26. February 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/stakingPool.sol	e22ef8e33781c8a144292a38069d963fd3998cc1

Please note: Files with a hash value different than those 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.

Imported packages

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

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/ReentrancyGuard.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 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

N/A



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

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	N/A
---------	-----



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 can set fees greater than 25%

✗ The owner able to set unfair tax

Description

For example, a decentralized exchange (DEX) smart contract may charge a fee for each trade executed on the platform. This fee can be set by the owner of the contract and may be a percentage of the trade value or a flat fee.

In other cases, the owner of the smart contract may set fees for accessing or using certain features of the contract. For instance, a subscription-based service smart contract may charge a monthly or yearly fee for access to premium features.

It's important to note that the fees set by the owner of a smart contract may not be the same as the gas fees required to execute the contract on the blockchain. Gas fees are generally set by the network and vary based on factors such as network congestion and the complexity of the transaction. The fees set by the contract owner, on the other hand, are independent of gas fees and are typically charged in addition to gas fees.

Overall, fees set by the owner of a smart contract can provide an additional source of revenue for the contract's owner and can help to ensure the sustainability of the contract over time.

Example

Our assumption is that the owner can adjust the fees up to 100%. If the unstaking fee is set to 100%, it implies that the full amount of tokens you intend to withdraw will be sent to the address specified as the recipient in the contract. This implies that the recipient will never have the intended amount of tokens in their wallet as it has all been used up in paying for the unstaking fee.

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. 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

The owner cannot lock user funds, but it is possible to pause the “deposit” function.


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
1	0	0	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
33	0

External	Internal	Private	Pure	View
32	36	1	0	15

StateVariables

Total	 Public
24	23



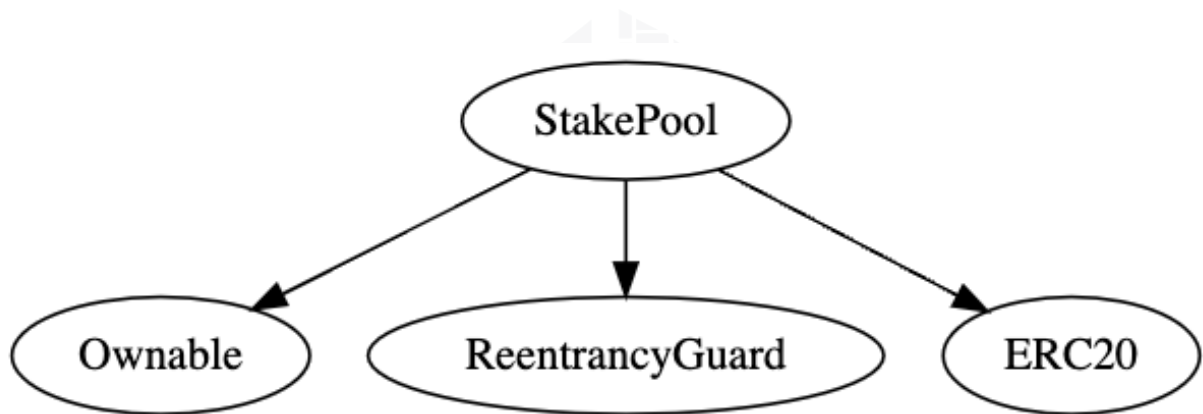
Capabilities

Solidity Versions observed	Transfers ETH	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
^0.8.1	Yes	No	No	No



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
Main	<ul style="list-style-type: none"> • Set Lock Duration • Withdraw all Rewards from the contract balance at any time. • Recover Wrong Tokens sent to the contract • Owner can drain staked tokens on their wallet which is not recommended • Stop Rewards • Update Fee Period • Set Staking Fees to any arbitrary value • Set Fee Collector Address and Pool Limits per user • Set Pool cap to any arbitrary value including zero which may result in the locking of a pool in which the user desires to deposit funds. • Update Staking and Unstaking Blocks • Add/Remove reward token

Recommendations

To avoid potential hacking risks, the client should carefully manage the private key of the privileged account. 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

No high issues



Medium issues

#1 | Unfair Tax Practice

File	Severity	Location	Status
Main	Medium	L481	ACK

Description - If the unstaking fee is active, the owner can set it to an arbitrary value, which can be very high. If done so, then all the withdrawable funds of the user will be credited to the owner's controlled wallet.

Remediation - We recommend putting an upper limit on the unstaking fees.

#2 | Owner can drain tokens

File	Severity	Location	Status
Main	Medium	L455	ACK

Description - If the contract is initialized with the "isRemovable" setting then the owner can drain the contract's complete balance, including the staked tokens, by calling this function.

Remediation - Make sure that taking out staked tokens is impossible. However, a fail-safe functionality can be implemented for safety where the staked tokens are returned to their respective investors.

Low issues

#1 | Missing Events

File	Severity	Location	Status
Main	Low	L200, 455, 470—485	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 | Redundant Check

File	Severity	Location	Status
Main	Low	L363	ACK

Description - The check inside this function is redundant because the “_amount” value is fetched from the contract’s mapping and not from the user.

Remediation - We recommend removing this check.

#3 | Inefficient implementation of element removal

File	Severity	Location	Status
Main	Low	L264	ACK

Description - The use of “delete” to remove the stakers is not recommended as it leaves empty slots in the storage, and when there are a very high number of stakers, there is a good chance of the storage being overflowed.

Remediation - We recommend using “pop” to remove the stakers.

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
Main	Informational	N/A	ACK

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

#2 | Floating Pragma

File	Severity	Location	Status
Main	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.

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