



**SOLID**Proof  
*Bring trust into your projects*

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

MADE IN GERMANY

# Defi Checkbook

## AUDIT

SECURITY ASSESSMENT

**30. April, 2024**

FOR



THE DEFI CHECKBOOK



[SolidProof.io](https://solidproof.io)



[@solidproof\\_io](https://t.me/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
Blacklist addresses	13
Fees and Tax	14
Lock User Funds	15
Components	16
Exposed Functions	16
StateVariables	16
Capabilities	17
Inheritance Graph	18
Centralization Privileges	19
Audit Results	21
Critical issues	21
High issues	22
Medium issues	23
Low issues	24





## Introduction

[SolidProof.io](https://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](https://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

<b>Project Name</b>	<b>The DeFi Checkbook</b>
<b>Website</b>	<a href="https://thedeficheckbook.com/">https://thedeficheckbook.com/</a>
<b>About the project</b>	The DeFi Checkbook enables mass transactions significantly reduce the gas fees and administrative overhead associated with performing numerous individual transactions, thereby offering an efficient, cost-effective solution for various blockchain-based distributions and financial processes.
<b>Chain</b>	BSC and Polygon
<b>Language</b>	Solidity
<b>Codebase Link</b>	Provided as Files
<b>Commit</b>	N/A
<b>Unit Tests</b>	Not Provided

## Social Medias

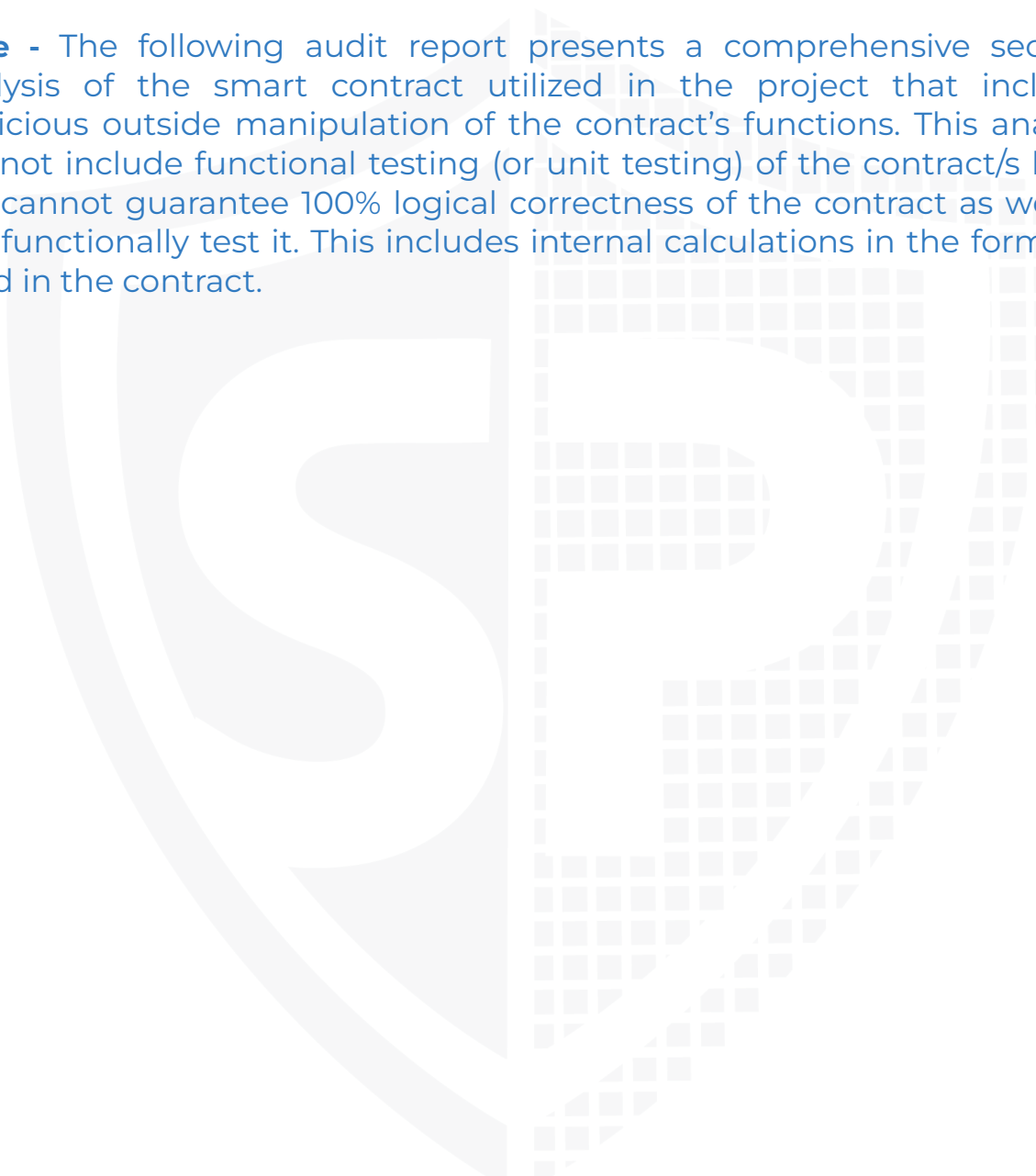
<b>Telegram</b>	N/A
<b>Twitter</b>	N/A
<b>Facebook</b>	N/A
<b>Instagram</b>	N/A
<b>Github</b>	N/A
<b>Reddit</b>	N/A
<b>Medium</b>	N/A
<b>Discord</b>	N/A
<b>Youtube</b>	N/A
<b>TikTok</b>	N/A
<b>LinkedIn</b>	N/A



## Audit Summary

Version	Delivery Date	Changelog
v1.0	23. April 2024	<ul style="list-style-type: none"> <li>• Layout Project</li> <li>• Automated- /Manual-Security Testing</li> <li>• Summary</li> </ul>
v1.1	30. April 2024	<ul style="list-style-type: none"> <li>• Reaudit</li> </ul>

**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
contracts/ICheckBook.sol	bf0fcedc1846e555099873482f2182eca8d26bbc
contracts/ IStableSwapRouter.sol	0bce4ebe48eefcde1d06ec3b7176591dc470e53b
contracts/ ICheckBookFactory.sol	099787ddf6cbbedbc347ee06aaa4ec9c1b1c28c86
contracts/TransferHelper.sol	ef3a5b6cdebd70202fc9f638e739eb775ff4eefa
contracts/ GlobalCheckbook.sol	f54cb685eb10e955feefbd85f03c9b205d8e3913
contracts/Checkbook.sol	88ac1e5aaef227b61ce258ba4a8d66f26710adb7
contracts/ CheckbookFactory.sol	8fe189b2fff5c5b2ee01931a979fc22e2cb1e40d
contracts/ContactBook.sol	1653abac4e79996a3a8430895cca9e5337d2ec08
contracts/Clones.sol	d8ecadb4465c685e1c8dd1c8a72ff8be079e9e14
contracts/ IUniswapV2Router02.sol	e5dabc6c5547b0dc0324e37e1326b0155ace84b2
contracts/IContactBook.sol	f9bc7b1659fdfaac313e8d1144cfd171a359cd01
contracts/IERC20.sol	f55c1f18fbcc8ed90e9d353197f93aaa1c6420c6

*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.*



## 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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 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.*

### 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.*

<b>Contract owner can set fees greater than 25%</b> <span style="float: right;"><b>✗ The owner able to set unfair fee</b></span>	
Description	<p>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.</p> <p>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.</p>
Example	<p>Our assumption is that the owner can adjust the referral and flat fees up to 100%. If the fee is set to 100%, it implies that the full amount of tokens you intend to send 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 fee.</p>
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 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

 <b>Contracts</b>	 <b>Libraries</b>	 <b>Interfaces</b>	 <b>Abstract</b>
8	1	8	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.

 <b>Public</b>	 <b>Payable</b>
175	14

<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>
165	115	0	7	61




## StateVariables

<b>Total</b>	 <b>Public</b>
53	35





## Capabilities

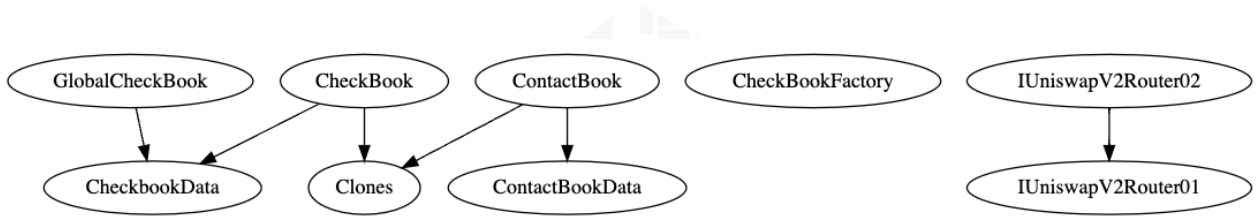
Solidity Versions observed	Transfers ETH	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
^0.8.19 ^0.8.0 0.8.19	Yes	Yes		





## 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
<b>Checkbook.sol</b>	<ul style="list-style-type: none"> <li>• Write Checks</li> <li>• Activate/Deactivate checkbooks</li> <li>• Set Cotntract Book Address and name</li> <li>• Update image URL</li> </ul>
<b>CheckbookFactory.sol</b>	<ul style="list-style-type: none"> <li>• Add/Remove referrer addresses</li> <li>• Update check book implementation address</li> <li>• Update Checkbook Amount and Referrer Fee</li> <li>• Set Router Address</li> <li>• Set Payment Token</li> <li>• Withdraw all type of tokens from the contract balance including the checkbook amount that is deposited by users</li> </ul>
<b>GlobalCheckbool.sol</b>	<ul style="list-style-type: none"> <li>• Withdraw tokens from the contract balance</li> <li>• Set Owner, and checkbook factory addresses</li> <li>• Set Name, Locked Status and Image URL for the checkbook</li> </ul>

### 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**





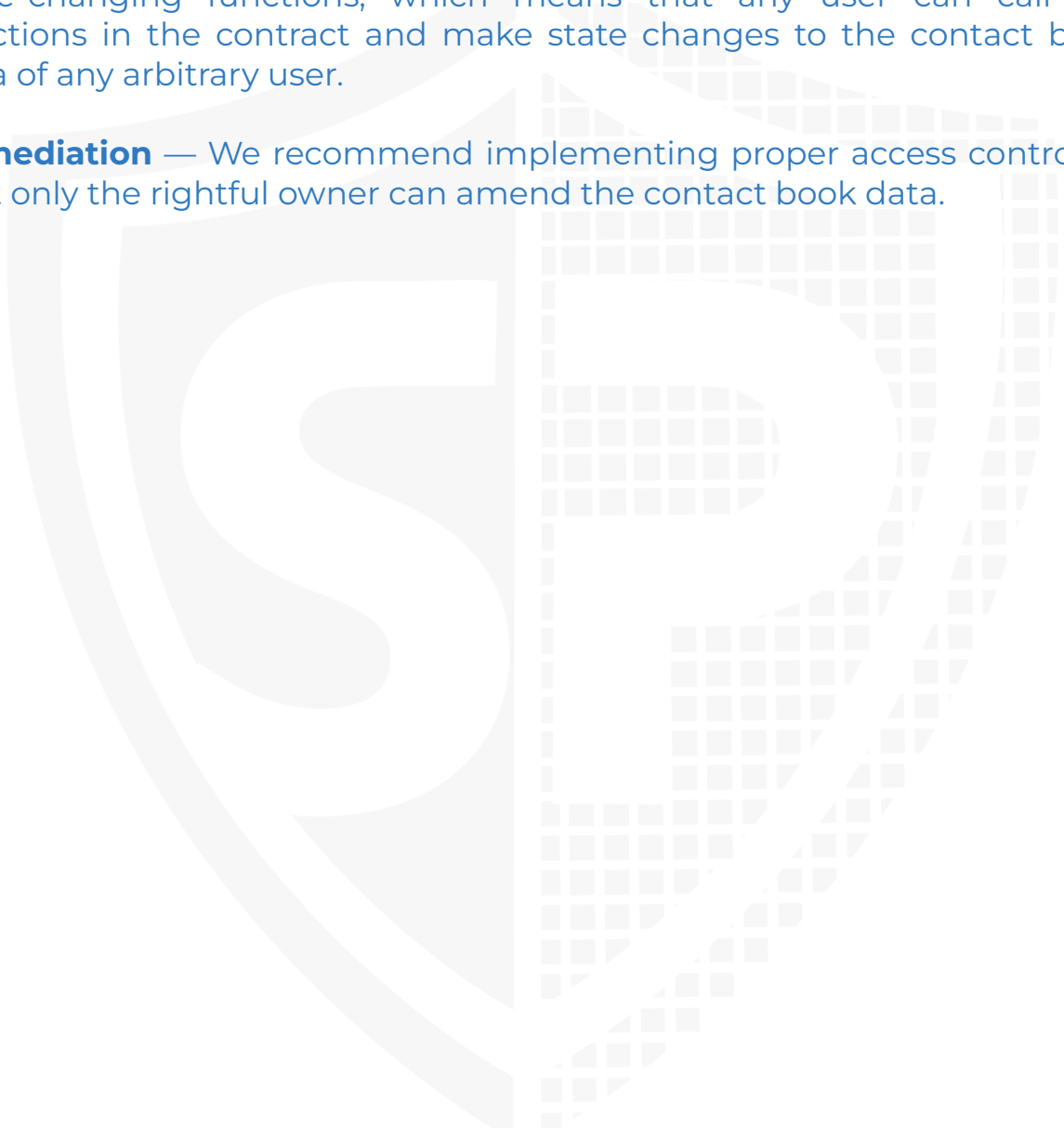
# High issues

## #1 | Missing Access Control

File	Severity	Location	Status
ContactBook	High	All Functions	Fixed

**Description** — The contract does not implement access controls on the state-changing functions, which means that any user can call the functions in the contract and make state changes to the contact book data of any arbitrary user.

**Remediation** — We recommend implementing proper access control so that only the rightful owner can amend the contact book data.





## Medium issues

**No medium issues**





## Low issues

### #1 | Missing Zero Address Validation

File	Severity	Location	Status
GlobalCheckbook	Low	L218, 222	Open
CheckbookFactory	Low	L291, 296	Open
Checkbook	Low	L330	Open

**Description** — Validate that the address passed in the function parameters is “non-zero”; otherwise, the code's intended behaviour will be affected.

### #2 | Missing Events

File	Severity	Location	Status
GlobalCheckbook	Low	All Ownership Functions	Open
CheckbookFactory	Low	L301—321	Open
Checkbook	Low	L326, 330	Open

**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 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	N/A	Open

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

### #2 | Floating Pragma

File	Severity	Location	Status
All	Informational	N/A	Open

**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.

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

File	Severity	Location	Status
All	Informational	N/A	Open

**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