SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Magic Bag

AUDIT

SECURITY ASSESSMENT

28. June, 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	MagicBag
Website	https://magicbag.io/
About the project	A Unique And Innovative Ethereum Token With Constantly Evolving Tokenomics.
Chain	Ethereum
Language	Solidity
Codebase Link	https://etherscan.io/token/0x8029d6984f700220472fc26269165764809d01a4#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

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

Audit Summary

Version	Delivery Date	Changelog
v1.0	27. June 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	28. June 2023	• Reaudit

File Overview

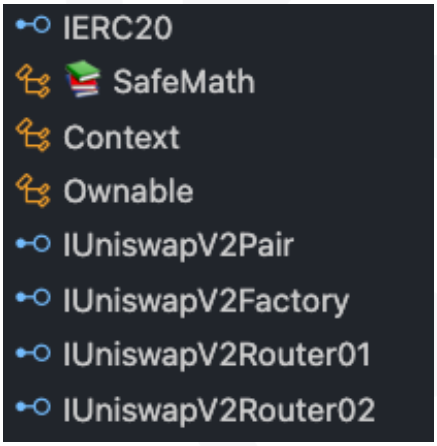
The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/MagicBag.sol	25df8e2ff2a6e24b1985dc20888a5ca36a15e2c4

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).



Note for Investors: We only Audited a token contract for **MagicBag**. However, If the project has other contracts (for example, a Presale contract, staking contract etc), and they were not provided to us in the audit scope, then 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

Medium or higher issues

No critical Issues found	✅ **Contract is safe to deploy**
Description	The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.
Comment	N/A

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	The owner is able to drain out liquidity by using the uniswap interfaces

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 can blacklist addresses	❌ **The owner able to burn tokens**
Description	If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value. Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders. Therefore, it is important for projects and platforms to be transparent about their blacklisting practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its blacklisting practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.
Example	The owner can manually add the wallets and contracts to the bots list, and all the addresses in that list will not be able to trade tokens.
Comment	N/A

Line/s: 502, 794
Codebase: Main

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 15%	✅ **The owner cannot blacklist addresses**
Description	The owner is not able to set the fees above 15%
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 user funds	❌ **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	An example of locking here is by blacklisting the addresses. This causes that the blacklisted address is not able to transfer, buy or sell anymore. Moreover, the owner also have the privilege to start/stop the trading at any time which will also result in the lock of user funds. Another example of locking the tokens in the contract is by setting the maximum wallet and transaction limit to zero.
Comment	N/A

Line/s: 502, 794, 707, 713
Codebase: Main

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	1	5	2

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
113	6

External	Internal	Private	Pure	View
66	96	10	23	40

StateVariables

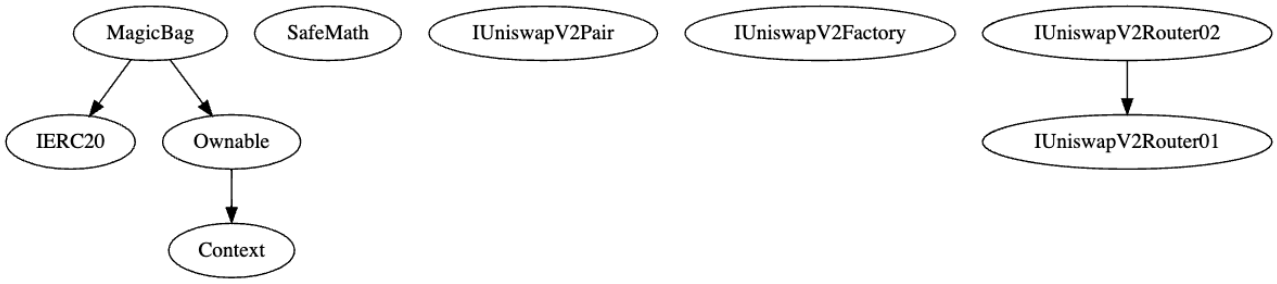
Total	🌐 Public
34	22

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.9`	----	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 the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
Main	‣ Include/Exclude wallets/contracts from reflection and TAX. Addresses excluded from reflections will not be able to claim rewards. ‣ Add/Remove wallets/contracts from the blacklist (bots list) ‣ Set treasury and LP pair address ‣ Set max wallet max transaction limit to any arbitrary value including zero and lock the transfers in the contract ‣ Manually Swap treasury and Reflection from the contract's balance ‣ Withdraw all types of tokens from the contract balance, including the native ones ‣ Pause/Unpause the following functionalities but not all at once - Burn Status - AutoLp Status - ReflectionStatus - Treasury Status

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

#1 | Enable/Disable Trading

File	Severity	Location	Status
Main	Medium	L707, 713	ACK

Description - The owner of the contract is able to enable/disable the trading functionality at any time. If done so, then no user will be able to transfer the funds. However, this functionality can also be abused to manipulate the price of the token.

Remediation - Make sure to make the enable trading variable changeable only once or remove it.

#2 | Missing Timelock

File	Severity	Location	Status
Main	Medium	L707, 713	Fixed

Description - The contract misses a timelock in the claim reflection function. This means that the claim function can be called recursively.

Remediation - We recommend putting a timelock so that the claim function cannot be called by an external contract recursively and only legitimate users will be able to claim reflections.

#3 | Owner can drain tokens

File	Severity	Location	Status
Main	Medium	L538	Open

Description - The owner of the contract is able to drain the complete balance of the contract by calling this function. Moreover, the owner is also able to drain the liquidity by using the uniswap interfaces.

Remediation - Make sure that it is not possible to take out native tokens and draining the liquidity manually is impossible.

#4 | Missing Zero Address Validation

File	Severity	Location	Status
Main	Low	L815, 778	ACK

Description

- Make sure to validate that the address passed in the function parameters is “non-zero”.

#5 | Missing Events

File	Severity	Location	Status
Main	Low	L707, 713, 762 — 815	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 in the contract.

#6 | Weak Randomization

File	Severity	Location	Status
Main	Low	L927	ACK

Description

- The contract uses a weak on-chain randomization which is not recommended as it can be easily predictable as it is happening on the chain which is visible to the public.

#7 | Missng “isContract” check

File	Severity	Location	Status
Main	Low	L687	Fixed

Description

- The contract doesn’t have any checks to verify whether the claim function is being called by an EOA or a contract.

Remediation - We recommend putting a check to verify that the callee of the claim function must be an EOA

#8 | Unused Return Values

File	Severity	Location	Status
Main	Informational	L1002	ACK

Description

- Ensure that all the return values of the function calls are used.

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