



# SOLIDProof

*Bring trust into your projects*

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

MADE IN GERMANY

## Gyrowin

# AUDIT

SECURITY ASSESSMENT

**19. September, 2023**

FOR



**SolidProof\_io**



**@solidproof\_io**



Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Imported packages	6
Components	7
Exposed Functions	7
Capabilities	8
Inheritance Graph	9
Audit Information	10
Vulnerability & Risk Level	10
Auditing Strategy and Techniques Applied	11
Methodology	11
Overall Security	12
Upgradeability	12
Ownership	13
Ownership Privileges	14
Minting tokens	14
Burning tokens	15
Blacklist addresses	16
Fees and Tax	17
Lock User Funds	18
Centralization Privileges	19
Audit Results	20

## 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

<b>Project Name</b>	<b>Gyrowin</b>
<b>Website</b>	<a href="https://www.gyro.win/">https://www.gyro.win/</a>
<b>About the project</b>	Gyrowin governance forum is a place where you can discuss about the new proposals with gyrowin community before it can be submitted for the on-chain voting.
<b>Chain</b>	Binance Smart chain(bsc)
<b>Language</b>	Solidity
<b>Codebase</b>	<a href="https://bscscan.com/address/0x7770Bb54655A84ea8159e81931bEA33936E39777#code">https://bscscan.com/address/0x7770Bb54655A84ea8159e81931bEA33936E39777#code</a>
<b>Commit</b>	N/A
<b>Unit Tests</b>	Not Provided

## Social Medias

<b>Telegram</b>	<a href="https://t.me/gyrowin">https://t.me/gyrowin</a>
<b>Twitter</b>	<a href="https://twitter.com/Gyrowin">https://twitter.com/Gyrowin</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>	<a href="https://discord.com/invite/gyrowin">https://discord.com/invite/gyrowin</a>
<b>YouTube</b>	N/A
<b>TikTok</b>	N/A
<b>LinkedIn</b>	N/A



## Audit Summary

Version	Delivery Date	Change Log
v1.0	19. August 2023	<ul style="list-style-type: none"> <li>· Layout Project</li> <li>· Automated/ Manual-Security Testing</li> <li>· Summary</li> </ul>
v1.2	19. September 2023	<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. 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 it was not functionally tested by us.





## File Overview

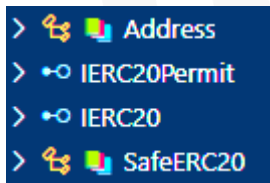
The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/Interfaces/IBEP20.sol	3105e2d77d31c9b0d11c40ac40e8258891e247dd
contracts/GyroWin.sol	bf035834dfd6e6637512c104d03cd2ff89065b3d

*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.*



**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.





## 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 needed within 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>
1	2	3	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>
61	2











<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>
49	90	4	3	30

## StateVariables

<b>Total</b>	 <b>Public</b>
34	12



## Capabilities

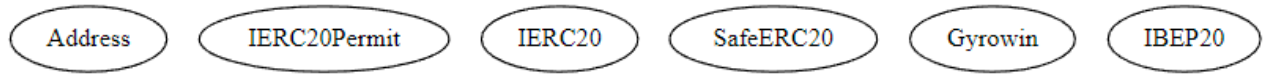
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
<div>&gt;=0.4.0</div> <div>^0.8.1</div> <div>^0.8.0</div> <div>=0.8.19</div>	<div>-----</div>	<div>yes</div>	<div>yes</div> <div>(3 asm blocks)</div>	<div>-----</div>	
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECREcover	 New/Create/Create2
<div>yes</div>		<div>yes</div>	<div>yes</div>	<div>yes</div>	





## 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.*



## Audit Information

### Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the 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. 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 upgradable



**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 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	N/A
Comment	N/A

**Note** – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

## 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 can mint new tokens**

**✗ The owner can mint new tokens.**

Description

The money plant wallet address of the contract can mint any new tokens which will be referred to as reward tokens and the amount of reward token will always be less than the burn amount, also this functionality is going to be used through an external contract which will handle all the minting and burning of tokens during the staking period.

Comment

N/A

**File/Line(s): L800-830**  
**codebase: GyroWin.sol**

```
function moneyPlant(uint256 burnAmount!, uint256 rewardAmount!) external onlyMoneyPlantCA() returns (bool) {
    require(balance[_treasury] >= burnAmount!, "GW: burn amount exceeds treasury balance");
    require(rewardAmount! <= burnAmount!, "GW: reward amount exceeds burn amount");

    totalSupply -= burnAmount!;
    balance[_treasury] -= burnAmount!;
    circulatingSupply -= burnAmount!;

    // burn game rewards
    emit Transfer(_treasury, deadAddress, burnAmount!);

    // money plant extra rewards
    if (rewardAmount! != 0) {
        circulatingSupply += rewardAmount!;
        balance[_treasury] += rewardAmount!;

        emit Transfer(deadAddress, _treasury, rewardAmount!);

        // totalSupply should be equal and less than 5% of max. total supply with the current run
        if (totalSupply > maxTotalSupply * 95 / 100) {
            revert("GW: require more burn");
        }

        // totalSupply should be equal to or greater than 20% of max. total supply
        if (totalSupply < maxTotalSupply * 80 / 100) {
            revert("GW: total supply should be equal to or greater than 4 billion");
        }
    }

    return true;
}
```

## 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 can burn tokens

✗ The owner can burn tokens.

#### Description

The operator of the contract can burn any amount of tokens which will be referred to as burn tokens, also this functionality is going to be used through an external contract which will handle all the minting and burning of tokens during the staking period.

#### Comment

The amount of circulating supply can be increased or decreased by an external contract.

File/Line(s): L800-830  
codebase: GyroWin.sol

```
function moneyPlant(uint256 burnAmount!, uint256 rewardAmount!) external onlyMoneyPlantCA() returns (bool) {
    require(balance[_treasury] >= burnAmount!, "GW: burn amount exceeds treasury balance");
    require(rewardAmount! <= burnAmount!, "GW: reward amount exceeds burn amount");

    totalSupply -= burnAmount!;
    balance[_treasury] -= burnAmount!;
    circulatingSupply -= burnAmount!;

    // burn game rewards
    emit Transfer(_treasury, deadAddress, burnAmount!);

    // money plant extra rewards
    if (rewardAmount! != 0) {
        circulatingSupply += rewardAmount!;
        balance[_treasury] += rewardAmount!;

        emit Transfer(deadAddress, _treasury, rewardAmount!);

        // totalSupply should be equal and less than 5% of max. total supply with the current run
        if (totalSupply > maxTotalSupply * 95 / 100) {
            revert("GW: require more burn");
        }

        // totalSupply should be equal to or greater than 20% of max. total supply
        if (totalSupply < maxTotalSupply * 80 / 100) {
            revert("GW: total supply should be equal to or greater than 4 billion");
        }
    }

    return true;
}
```



## 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 set fees more than 25%**

Description

The owner is not able to set the fees above 25%.

Comment

The amount of fees cannot be set to more than 1%.

## 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 token 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 user funds.**

**✗ The owner can lock user funds.**

Description

The operator of this contract can lock the tokens from the transfer of tokens and the locked tokens will be subtracted from the circulating supply.

Comment

The lock function can decrease the circulating supply numbers info that we are storing in the token contract variable. The contract contains external contracts such as vesting contract, reserve contract, and freezing contract that actually locks the tokens and calls the token contract to reduce the circulating supply numbers.

**File/Line(s): L1412-1430**  
**codebase: GyroWin.sol**

```
function initializeLock(address contractAddr!, uint256 amount!) external onlyOperatorCA() {
    require(
        swapPair[contractAddr!] == true ||
        contractAddr! == vestingCA ||
        contractAddr! == reserveCA ||
        contractAddr! == loyaltyCA ||
        contractAddr! == freezeLockCA,
        "GW: invalid contract");
    require(amount! <= balance[address(this)], "GW: amount exceeds balance");

    if (!vestingLock && swapPair[contractAddr!] == false) {
        transferToken(address(this), contractAddr!, amount!);
    }

    // subtract locked token from circulatingSupply
    circulatingSupply -= amount!;

    emit LockInfo(contractAddr!, amount!, block.timestamp);
}
```

## 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
GyroWin.sol	<ul style="list-style-type: none"> <li>➤ The money plant contract can mint the reward tokens which will be less than the number of tokens burned during the staking of tokens.</li> <li>➤ The owner can transfer ownership.</li> <li>➤ The owner can change the operator's address. Also, the operator of this contract will always remain as an external contract. The operator will be a contract address only.</li> <li>➤ The owner can update the money plant contract address.</li> <li>➤ The owner can update the treasury account.</li> <li>➤ The owner can update the reserve contract address.</li> <li>➤ The owner can update the loyalty contract address.</li> <li>➤ The owner can update the freeze contract address.</li> <li>➤ The owner can update the vesting contract address.</li> <li>➤ The owner can change the swap pair address.</li> <li>➤ The owner can whitelist addresses from gas fees.</li> <li>➤ The owner can set the gas limit to not less than 3000000000 Wei.</li> <li>➤ The owner can whitelist addresses from fees.</li> <li>➤ The owner can set fees of not more than 1%.</li> <li>➤ The owner can enable trading only once.</li> <li>➤ The owner can claim the stuck tokens. Also, the owner can claim the contract's own tokens.</li> <li>➤ The operator can lock and unlock the total amount of tokens of a user for vesting.</li> </ul>

## 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 Result

### #1 | Operator can mint and burn the tokens from external address.

File	Severity	Location	Status
GyroWin.sol	Medium	L800-830	ACK

**Description** - It is recommended that the burning of the tokens should not be done without any allowance as of now, the operator can burn any number of tokens from other accounts, which is not recommended. Also, The operator can mint the tokens after the initial deployment which will always be less than the amount of tokens burned.

### #2 | Transfer of tokens without enabling trade.

File	Severity	Location	Status
GyroWin.sol	Medium	1062-1100	ACK

**Description** – The trading needs to be enabled by the owner in order for regular users to transfer tokens. On the contrary, the owner can authorize addresses manually and those addresses will be able to trade tokens. This functionality can be exploited in the following way, For example, there is a presale and the wallets used for the presale can be authorized by the owner. All the tokens obtained can be consolidated into a final wallet address and facilitate trading and selling of the acquired tokens, the last wallet address can be authorized.

### #3| Operator can lock tokens for unlimited period.

File	Severity	Location	Status
GyroWin.sol	Medium	L1412-1430	ACK

**Description** – The operator has the ability to lock the circulating supply of the tokens. Also, the contract contains external contracts for vesting, loyalty, and reserve that can lock the tokens and increase or decrease the circulating supply.

### #4 | Missing zero and dead address check.

File	Severity	Location	Status
GyroWin.sol	Low	L1219-1225, L1229-1235, L1239-1245, L1331-1336	ACK

**Description** - The address can be set to a zero or dead address, which is not recommended. Add a 'require' check so that we it will throw a revert.

## #5 | Floating pragma solidity version.

File	Severity	Location	Status
GyroWin.sol	Low	L2	ACK

**Description** - Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

## #6 | Missing events.

File	Severity	Location	Status
GyroWin.sol	Low	L831-834, L843-855, L863-875, L700, L710, L716	Fixed

**Description** – It is recommended to emit all the critical parameter changes.

**Alleviation** – events added in the functions.

## #7 | Missing length check.

File	Severity	Location	Status
GyroWin.sol	Low	L843-854, L863-875	Fixed

**Description** – Add a 'require' that the account length and the length of amount array must be the same. Otherwise, the operation will fail.

**Alleviation** – Functionality is removed from the contract.

## #8 | Owner can claim contract's own tokens.

File	Severity	Location	Status
GyroWin.sol	Low	L1489-1492	ACK

**Description** – Add a 'require' check that the owner cannot claim the contract's own tokens. It is recommended that the owner of the contract should not have the authority to claim the contract's token from contract balance.

## #9 | Missing 'require' check.

File	Severity	Location	Status
GyroWin.sol	Low	L1423-1459	Fixed

**Description** - Add a 'require' check that the amount from the account balance must be greater than the subtracted value otherwise the operation will fail.



#### #10 | Incorrect 'require' check.

File	Severity	Location	Status
GyroWin.sol	Low	L784, L818	Fixed

**Description** – Add '==' operator for comparison between two values. Here, In the function assignment operator is used. This will not be able to check and will give incorrect results.

#### #11 | Missing zero check.

File	Severity	Location	Status
GyroWin.sol	Low	L1342-1347	Fixed

**Description** – Add a 'require' check that the account length must be greater than zero to perform the operation.

#### #12 | Incorrect solidity naming conventions.

File	Severity	Location	Status
GyroWin.sol	Informational	L536-539, L569	ACK

**Description** – constant variable starts with uppercase. Follow the Solidity [naming convention] (<https://solidity.readthedocs.io/en/v0.4.25/style-guide.html#naming-conventions>).

**Note** – This contract contains some external contracts that are out of the scope of audits, and we cannot claim any security concerns related to them and will not be responsible for any security issues. We recommend doing your own research before investing.



## Legend for the Issue Status

Attribute or Symbol	Meaning
<b>Open</b>	The issue is not fixed by the project team.
<b>Fixed</b>	The issue is fixed by the project team.
<b>Acknowledged(ACK)</b>	The issue has been acknowledged or declared as part of business logic.







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

MADE IN GERMANY