GreenLeage

Smart Contract Audit Report

SYSFIXED

20 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
GreenLeage	GL	BSC

Addresses

Contract address	0xeFDA7E5b3529123b6283426D60A0788420307ACf
Contract deployer address	0x4f5E308dFecA0391EB366300f2bDAEF08352ae77

Project Website

https://www.greenleage.top/

Codebase

https://bscscan.com/address/0xeFDA7E5b3529123b6283426D60A0788420307ACf#contracts

SUMMARY

Green League is an open protocol connecting funders to rural planters worldwide. The GreenPay Green Pay is a TON-based digital wallet bot, uses fiat currency to trade $GL. The Entity organization Project was jointly created by Rural tree planters, Individual and organizational tree funders, Partner NGOs, Open-source contributors, and Other organizations who care about the future of our planet. Blockchain Development - Berliner Experienced developer, it used to work for TeslaSafe(200X), Berliner, DK.

Contract Summary

Documentation Quality

GreenLeage provides a document with a very good standard of solidity base code.

- The technical description is provided clearly and structured but also has a low risk issue.

Code Quality

The Overall quality of the basecode is GOOD

- Standart solidity basecode and rules are already followed with GreenLeage Project .

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 145, 177, 200, 201, 236, 272, 529, and 557.
- SWC-103 | A floating pragma is set on lines 11, 91, and 118. The current pragma Solidity directive is ""^0.8.17"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- SWC-108 | State variable visibility is not set on lines 531.It is best practice to set the visibility of state variables explicitly. The default visibility for "protections" is internal. Other possible visibility settings are public and private.
- SWC-110 | Out of bounds array access on lines 732

CONCLUSION

We have audited the GreenLeage Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in GreenLeage Project. This process is used to find bugs, technical issues, and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with a low risk issue on the contract project.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. Some of the low issues that we found were assert violation, floating pragma set, and default visibility. The functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Thu Jan 19 2023 07:30:23 GMT+0000 (Coordinated Universal Time)
Finished	Fri Jan 20 2023 08:31:13 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	gl.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 145

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
144   function add(uint256 a, uint256 b) internal pure returns (uint256) {
145   uint256 c = a + b;
146   require(c >= a, "SafeMath: addition overflow");
147
148   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 177

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
176   require(b <= a, errorMessage);
177   uint256 c = a - b;
178
179   return c;
180   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 200

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
199
200   uint256 c = a * b;
201   require(c / a == b, "SafeMath: multiplication overflow");
202
203   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 201

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
200   uint256 c = a * b;
201   require(c / a == b, "SafeMath: multiplication overflow");
202
203   return c;
204   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 236

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
235   require(b > 0, errorMessage);
236   uint256 c = a / b;
237   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
238
239   return c;
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 272

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
271   require(b != 0, errorMessage);
272   return a % b;
273   }
274   }
275
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 529

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
528   address public deadAddress = 0x000000000000000000000000000000000000dEaD;
529   uint256 private minimumTokensBeforeSwap = 20000 * 10 ** 18;
530
531   bool inSwapAndLiquify;
532   bool public swapAndLiquifyEnabled = true;
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 557

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- gl.sol

Locations

```
556   _decimals = 18;
557   uint256 amount = 100000000 * 10 ** 18;
558   _totalSupply = amount;
559
560   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

low SEVERITY

The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- gl.sol

Locations

```
10
11   pragma solidity ^0.6.0;
12
13   /**
14   * @dev Interface of the ERC20 standard as defined in the EIP.
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 91

low SEVERITY

The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- gl.sol

Locations

```
90
91   pragma solidity ^0.6.0;
92
93   /*
94   * @dev Provides information about the current execution context, including the
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 118

low SEVERITY

The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- gl.sol

Locations

```
117
118   pragma solidity ^0.6.0;
119
120   /**
121   * @dev Wrappers over Solidity's arithmetic operations with added overflow
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 531

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source File

- gl.sol

Locations

```
530
531   bool inSwapAndLiquify;
532   bool public swapAndLiquifyEnabled = true;
533   bool public swapAndLiquifyByLimitOnly = false;
534
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 732

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- gl.sol

Locations

```
731   address[] memory path = new address[](2);
732   path[0] = address(this);
733   path[1] = uniswapV2Router.WETH();
734
735   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 733

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- gl.sol

Locations

```
732   path[0] = address(this);
733   path[1] = uniswapV2Router.WETH();
734
735   _approve(address(this), address(uniswapV2Router), tokenAmount);
736
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.