

# Smart Contract Security Audit V1

## GLLE Smart Contract

15/2/2022



<https://saferico.com/>

[business@saferico.com](mailto:business@saferico.com)

[https://t.me/SFI\\_ANN](https://t.me/SFI_ANN)

—

# Table of Contents

## **Table of Contents**

## **Background**

## **Project Information**

Smart Contract Information

Executive Summary

## **File and Function Level Report**

**File in Scope:**

## **Issues Checking Status**

Severity Definitions

Audit Findings

## **Automatic testing**

Testing proves

Inheritance graph

Call graph

## **Unified Modeling Language (UML)**

**Functions signature**

**Automatic general report**

## **Conclusion**

## **Disclaimer**

# Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

## Project Information

- **Platform:** Polygon
- **OLD Contract Address:** 0x492f8bf43ce24da42b8672241Aa1F34845AE896F
- **NEW Contract Address:** 0x970BB013018dd06433945369331Eed33cDDf9d06
- **OLD Code:**

<https://polygonscan.com/address/0x492f8bf43ce24da42b8672241Aa1F34845AE896F#code>

- **NEW Code:**  
<https://polygonscan.com/address/0x970BB013018dd06433945369331Eed33cDDf9d06#code>

## Smart Contract Information

- Name: GLLE
- Total Supply: 20,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (Polygon)

GLLE Smart contract on Polygon test net to test every read and write function.

<https://mumbai.polygonscan.com/address/0x851ca63619904351a2ed7a0c76fa380828320009>

## Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
<b>Secured</b>	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 1 critical, 0 high, 0 medium, 2 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

GLLE.sol

# File and Function Level Report

## File in Scope:

Contract Name	SHA 256 hash	Contract Address
GLLE.sol	1309ccac33bf80dd765537e e2bea23fcb44deb90a08787 3a008c0d18c0b727ad	0x492f8bf43ce24da42b8672241Aa1F34845AE 896F

- Contract: GLLE
- Inherit: ERC20, ERC20Burnable, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	<b>Passed</b>
symbol	✓	Read / public	<b>Passed</b>
balanceOf	✓	Read / public	<b>Passed</b>
totalSupply	✓	Read / public	<b>Passed</b>
decimal	✓	Read / public	<b>Passed</b>
allowance	✓	Read / public	<b>Passed</b>
owner	✓	Read / public	<b>Passed</b>
approve	✓	Write / public	<b>Passed</b>
transferFrom	✓	Write / public	<b>Passed</b>
transfer	✓	Write / public	<b>Passed</b>
burn	✓	Write / public	<b>Passed</b>
decreaseAllowance	✓	Write / public	<b>Passed</b>
increaseAllowance	✓	Write / public	<b>Passed</b>

burnFrom	✓	Write / public	<b>Passed</b>
mint	✓	Write / public	<b>Passed</b>
transferOwnership	✓	Write / public	<b>Passed</b>
renounceOwnership	✓	Write / public	<b>Passed</b>

# Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Timestamp dependence.	Passed
6	Integer Overflow and Underflow.	Passed
7	DoS with Revert.	Passed
8	DoS with block gas limit.	Passed with notes
9	Methods execution permissions.	Passed
10	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	Passed
11	The impact of the exchange rate on the logic.	Passed
12	Private user data leaks.	Passed
13	Malicious Event log.	Passed
14	Scoping and Declarations.	Passed
15	Uninitialized storage pointers.	Passed
16	Arithmetic accuracy.	Passed
17	Design Logic.	Passed

## Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.



## Audit Findings

### Critical:

#The owner can mint any amount he wants

#### Description

The owner has the ability to mint as much as he wants; this represents a risk for the users because in that case, their funds will be lower in price.

```
function mint(address to, uint256 amount) public onlyOwner {  
    _mint(to, amount);  
}
```

#### Remediation

There are two possibilities to remediate the risk. Make this function internal which no one can control it . The second one is remove the mint function from the contract which is recommended by the auditor.

Status: **Acknowledged**.

Team Comment: The mint function is required to mint the rewards, so they can't make it internal or remove it.

### High:

No High severity vulnerabilities were found

### Medium:

No Medium severity vulnerabilities were found

### Low:

# Functions that do not have a function visibility

#### Description

Functions that do not have a function visibility type specified are public by default. This can lead to a vulnerability if a developer forgot to set the visibility and a malicious user is able to make unauthorized or unintended state changes.

```
_mint(msg.sender, 20000000 * 12 ** decimals());
```

#### Remediation

Make this function like that

```
_mint(msg.sender, 20000000 * 12 ** 18);
```

Or

```
_mint(msg.sender, 2000000000000000000000000 * 12);
```

Status: **Closed**. Fixed in version2.

#### #Pragam version not fixed

##### Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.2 instead of ^0.8.2). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

##### Remediation

Remove the ^ sign to lock the pragma version.

Status: **Closed**. Fixed in version2.

#### **Very Low:**

No Very Low severity vulnerabilities were found.

#### **Notes:**

#### #Compiler version is old

##### Description

The compiler being used was released a year ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

Status: **Acknowledged**

# Automatic Testing

## 1- Check for security

1309ccac33bf80dd765537ee2bea23fcb44deb90a087873a008c0d18c0b727ad

File: GLE.sol | Language: solidity | Size: 495 bytes | Date: 2022-02-15T07:38:12.169Z

Critical	High	Medium	Low	Note
0	0	0	1	0



## 2- SOLIDITY STATIC ANALYSIS

### SOLIDITY STATIC ANALYSIS

☒ Select all ☒ Autorun Run

**Security**

☒ Select Security

- ☒ **Transaction origin:**  
'tx.origin' used
- ☒ **Check-effects-interaction:**  
Potential reentrancy bugs
- ☒ **Inline assembly:**  
Inline assembly used
- ☒ **Block timestamp:**  
Can be influenced by miners
- ☒ **Low level calls:**  
Should only be used by experienced devs
- ☒ **Block hash:**  
Can be influenced by miners
- ☒ **Selfdestruct:**  
Contracts using destructed contract can be broken

**Gas & Economy**

☒ Select Gas & Economy

- ☒ **Gas costs:**  
Too high gas requirement of functions
- ☒ **This on local calls:**  
Invocation of local functions via 'this'
- ☒ **Delete dynamic array:**  
Use require/assert to ensure complete deletion
- ☒ **For loop over dynamic array:**  
Iterations depend on dynamic array's size
- ☒ **Ether transfer in loop:**  
Transferring Ether in a for/while/do-while loop

### SOLIDITY STATIC ANALYSIS

**ERC**

☒ Select ERC

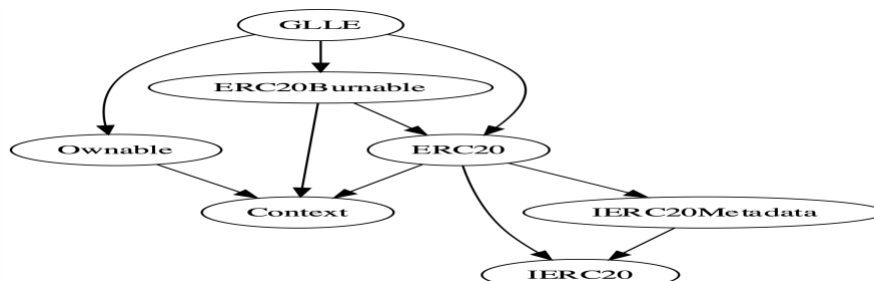
- ☒ **ERC20:**  
'decimals' should be 'uint8'

**Miscellaneous**

☒ Select Miscellaneous

- ☒ **Constant/View/Pure functions:**  
Potentially constant/view/pure functions
- ☒ **Similar variable names:**  
Variable names are too similar
- ☒ **No return:**  
Function with 'returns' not returning
- ☒ **Guard conditions:**  
Ensure appropriate use of require/assert
- ☒ **Result not used:**  
The result of an operation not used
- ☒ **String length:**  
Bytes length != String length
- ☒ **Delete from dynamic array:**  
'delete' leaves a gap in array
- ☒ **Data truncated:**  
Division on int/uint values truncates the result

## 3- Inheritance graph



## 4- SOLIDITY UNIT TESTING

### SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

☒ Select all

☒ tests/GLLE\_test.sol

Progress: 1 finished (of 1)

**PASS** testSuite (tests/GLLE\_test.sol)

✓ Before all

✖

✓ Check success

✖

✓ Check success2

✖

✓ Check failure

✖

✓ Check sender and value

✖

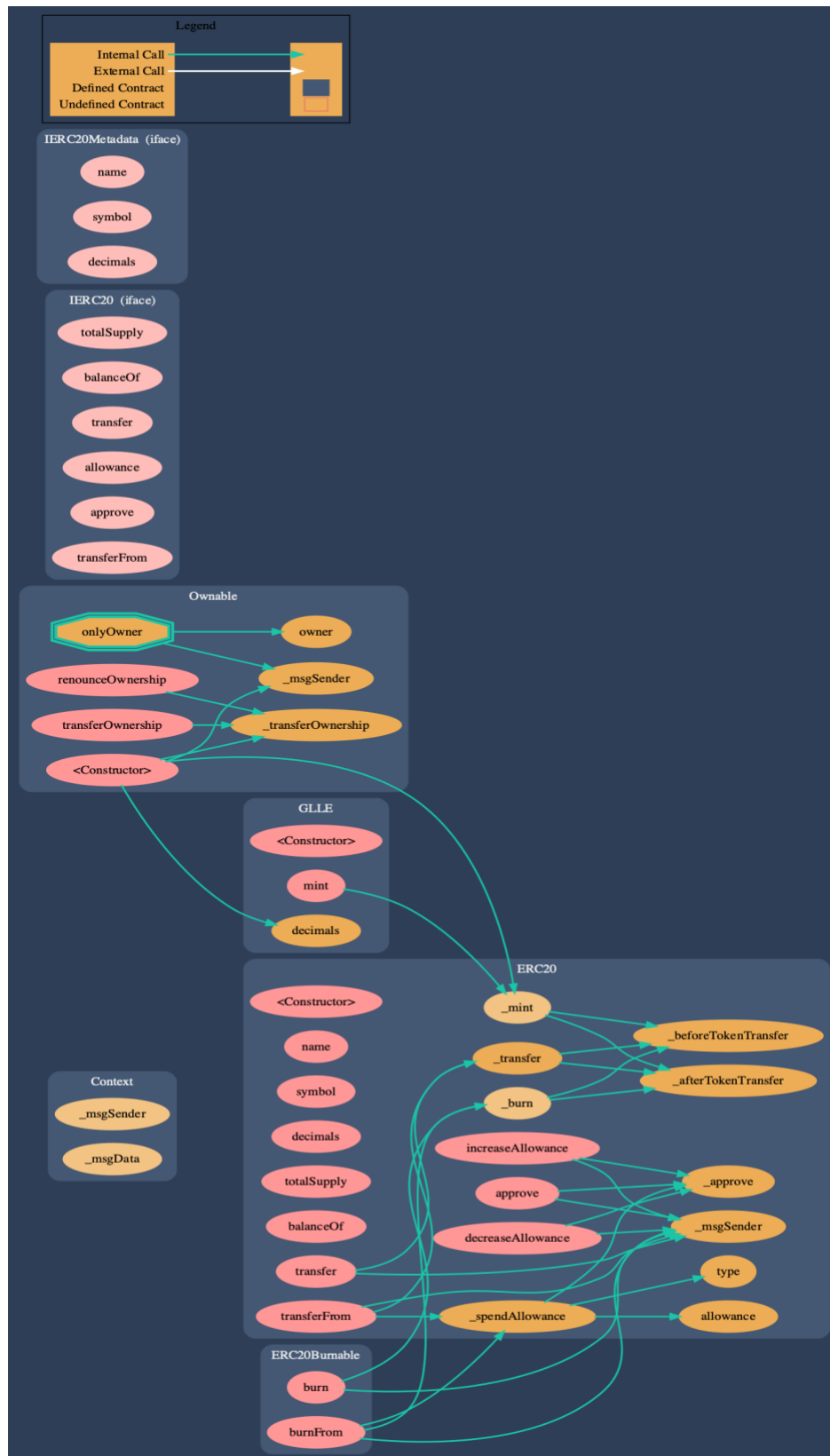
**Result for tests/GLLE\_test.sol**

Passed: 5

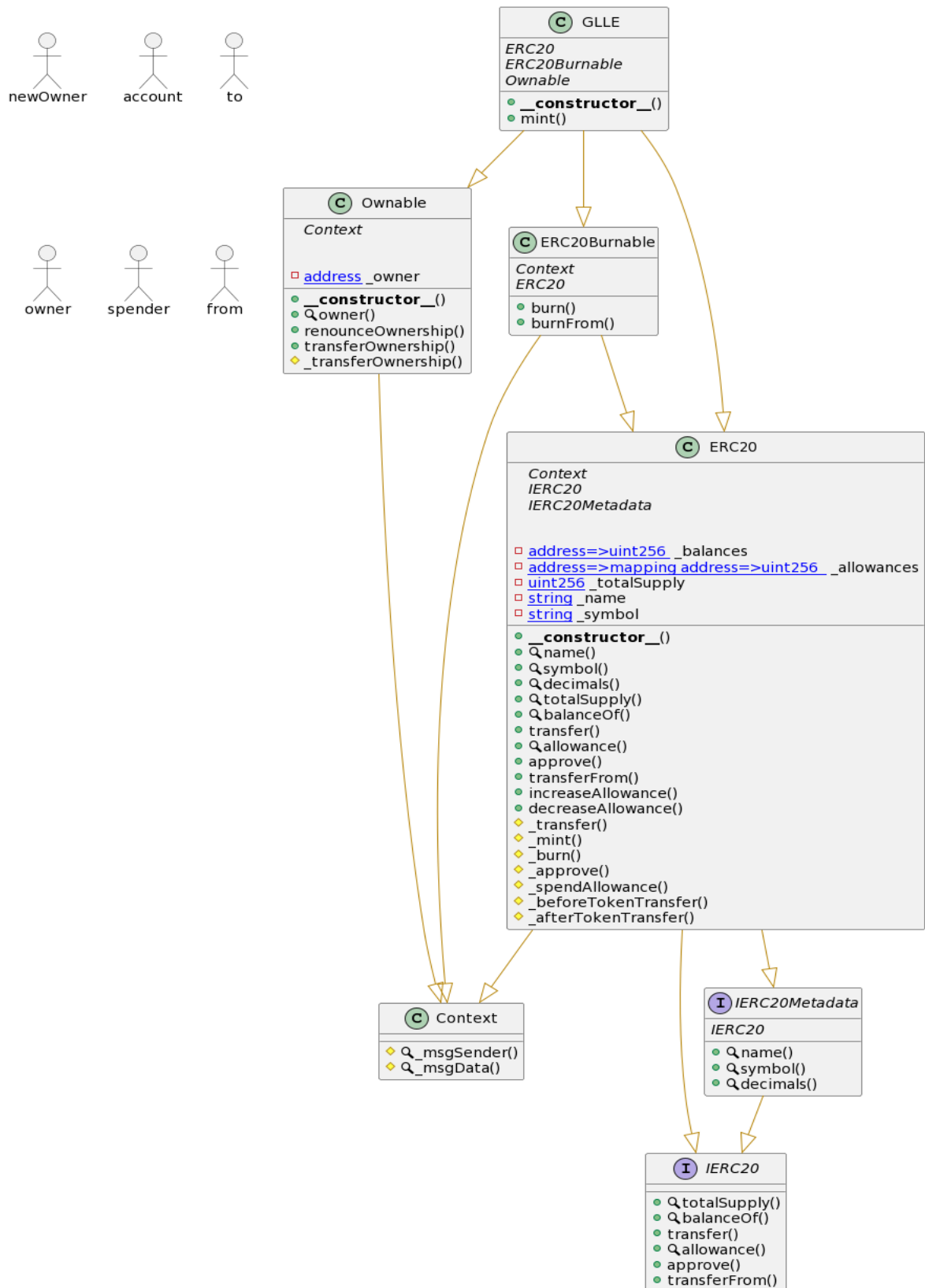
Failed: 0

Time Taken: 0.51s

## 5- Call graph



# Unified Modeling Language (UML)



## Functions signature

Sighash		Function Signature
=====		
39509351	=>	increaseAllowance(address,uint256)
119df25f	=>	_msgSender()
8b49d47e	=>	_msgData()
8da5cb5b	=>	owner()
715018a6	=>	renounceOwnership()
f2fde38b	=>	transferOwnership(address)
d29d44ee	=>	_transferOwnership(address)
18160ddd	=>	totalSupply()
70a08231	=>	balanceOf(address)
a9059cbb	=>	transfer(address,uint256)
dd62ed3e	=>	allowance(address,address)
095ea7b3	=>	approve(address,uint256)
23b872dd	=>	transferFrom(address,address,uint256)
06fdde03	=>	name()
95d89b41	=>	symbol()
313ce567	=>	decimals()
a457c2d7	=>	decreaseAllowance(address,uint256)
30e0789e	=>	_transfer(address,address,uint256)
4e6ec247	=>	_mint(address,uint256)
6161eb18	=>	_burn(address,uint256)
104e81ff	=>	_approve(address,address,uint256)
1532335e	=>	_spendAllowance(address,address,uint256)
cad3be83	=>	_beforeTokenTransfer(address,address,uint256)
8f811a1c	=>	_afterTokenTransfer(address,address,uint256)
42966c68	=>	burn(uint256)
79cc6790	=>	burnFrom(address,uint256)
40c10f19	=>	mint(address,uint256)

# Automatic general report

## Files Description Table




















File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/GLLE.sol	82b9700562e3992541e3875bc016d5538826e937

## Contracts Description Table

Contract	Type	Bases	
:-----: :-----: :-----: :-----:			
L	**Function Name**	**Visibility**	**Mutability**
**Modifiers**			
**Context**	Implementation		
L   _msgSender	Internal		
L   _msgData	Internal		
**Ownable**	Implementation	Context	
L   <Constructor>	Public		NO
L   owner	Public		NO
L   renounceOwnership	Public		onlyOwner
L   transferOwnership	Public		onlyOwner
L   _transferOwnership	Internal		
**IERC20**	Interface		
L   totalSupply	External		NO
L   balanceOf	External		NO
L   transfer	External		NO
L   allowance	External		NO
L   approve	External		NO
L   transferFrom	External		NO
**IERC20Metadata**	Interface	IERC20	
L   name	External		NO
L   symbol	External		NO
L   decimals	External		NO
**ERC20**	Implementation	Context, IERC20, IERC20Metadata	
L   <Constructor>	Public		NO
L   name	Public		NO
L   symbol	Public		NO
L   decimals	Public		NO
L   totalSupply	Public		NO
L   balanceOf	Public		NO
L   transfer	Public		NO
L   allowance	Public		NO
L   approve	Public		NO
L   transferFrom	Public		NO
L   increaseAllowance	Public		NO





```

| L | decreaseAllowance | Public ! |  | NO ! |
| L | _transfer | Internal  |  | |
| L | _mint | Internal  |  | |
| L | _burn | Internal  |  | |
| L | _approve | Internal  |  | |
| L | _spendAllowance | Internal  |  | |
| L | _beforeTokenTransfer | Internal  |  | |
| L | _afterTokenTransfer | Internal  |  | |
| | | | |
| **ERC20Burnable** | Implementation | Context, ERC20 | | |
| L | burn | Public ! |  | NO ! |
| L | burnFrom | Public ! |  | NO ! |
| | | | |
| **GLLE** | Implementation | ERC20, ERC20Burnable, Ownable | | |
| L | <Constructor> | Public ! |  | ERC20 |
| L | mint | Public ! |  | onlyOwner |

```

### Legend

Symbol	Meaning
	Function can modify state
	Function is payable

## Conclusion

The contracts are written systematically. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “secured”.

- ✓ No volatile code.
- ✓ Not many high severity issues were found.

# Disclaimer

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 the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. 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 Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.