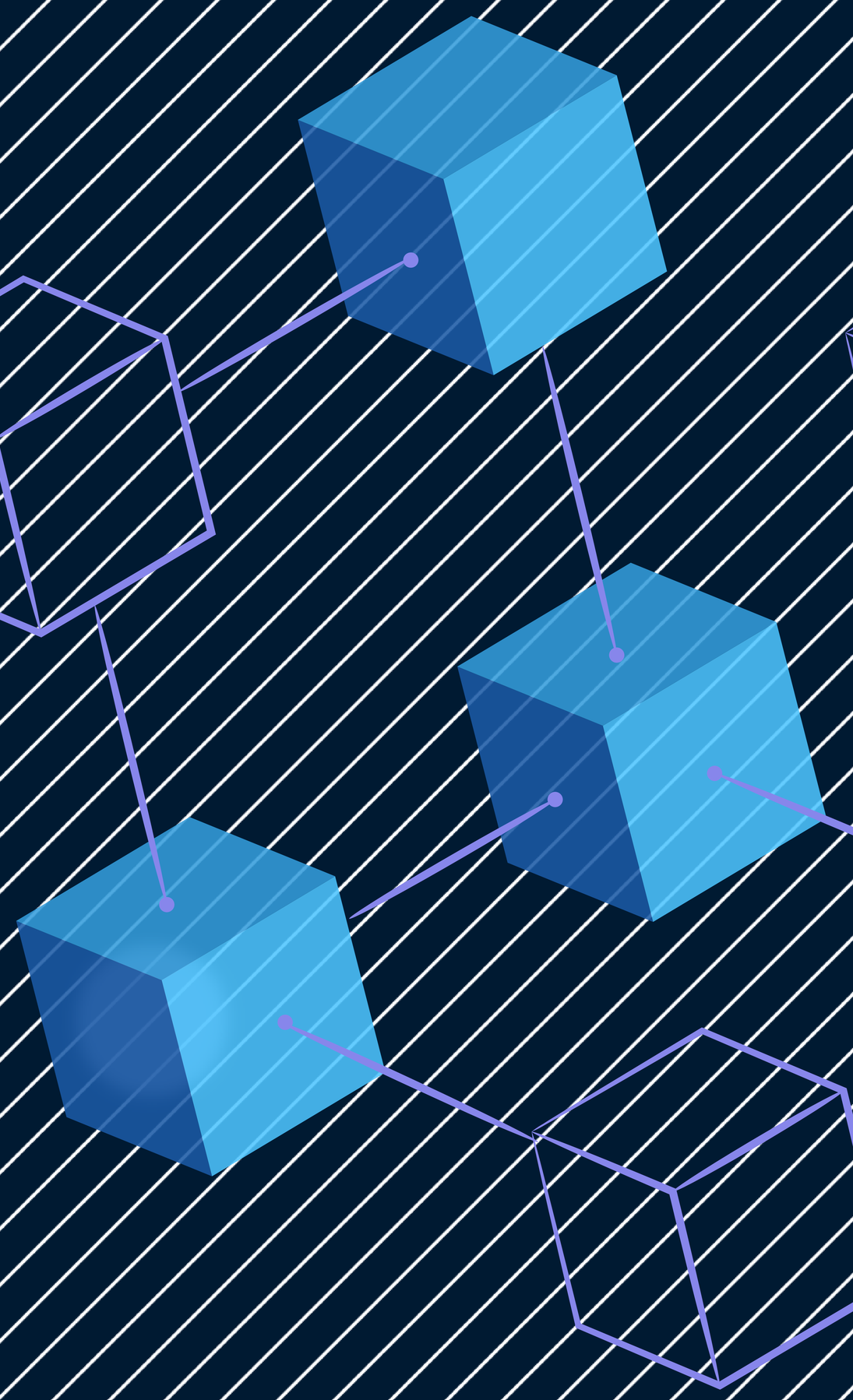
QuillAudits

Audit Report
December, 2021



For

GALILEO
EXCHANGE

Contents

Scope of the Audit

The scope of this audit was to analyze and document the GELT Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	0
Closed	0	0	0	3

Introduction

During the period of **Dec 03, 2021 to Dec 07, 2021** - QuillAudits Team performed a security audit for **GELT** smart contracts.

The code for the audit was taken from following the official link: https://github.com/CandleBets/contracts/blob/master/GELT.sol

V	Date	Github	Commit ID
1	Dec 03	https://github.com/akshayb28/GELT/blob/main/GELT%20v2.sol	8f4ab9cf70798b0a6b51fea75769714c58c7f4d5
2	Dec 07	https://github.com/CandleBets/contracts/blob/master/GELT.sol	14c3a899d7bef78633ffb8265aca7751c34d51da

Issues Found

A. Contract – ERC20

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

1. **Using old Solidity version**

   Throughout the contract, Solidity 0.8.0 is used and is deployable. However, at the time of writing there are 2 known bugs in version 0.8.0. Consider upgrading your contracts to use the latest version of Solidity, 0.8.9.

   Status: **Fixed** in version 02

2. **Typos**

   There are multiple typos in the contract:

   L18: **Initalizes** should be **Initializes**

   Status: **Fixed** in version 02

3. Variables declared as uint instead of uint256

To favor explicitness, consider changing all instances of **uint** into **uint256** in the entire codebase. The following instances were found:

L21: constructor(string memory _name, string memory _symbol, uint _initialSupply) ERC20(_name, _symbol)
L31: function mint(uint _amount)
L42: function burn(uint _amount)

Status: **Fixed** in version 02

Functional test

Address: 0x6C60C368a1970BeDFec6Bb7E00E31187417a830E
Evn: https://testnet.bscscan.com/

Function name	Input	Output	TX	Status
approve (From 0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC)	0x153b057d5d7262dC92099B59c975255ecE66784F, 10000000	true	0xf23a1b70ff585339661614b1040742e06d68f7b9a8f386dd1261a8a36e61686f	Passed
allowance	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d7262dC92099B59c975255ecE66784F"	10000000	N/A	Passed
decreaseAllowance (From 0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC)	0x153b057d5d7262dC92099B59c975255ecE66784F, 10000000	true	0xc57a401aaa22bf89cfc0a0abb9ceff7757fed39332b109aa122fca9f3e88943b	Passed
allowance	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d7262dC92099B59c975255ecE66784F"	0	N/A	Passed
increaseAllowance (From 0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC)	0x153b057d5d7262dC92099B59c975255ecE66784F, 1000000	true	0x212be6e3abbf7a4ce65eca56c34ace54ca5aa8fbe3a78bf055e6deae1ab6d0a0	Passed
allowance	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d726	1000000	N/A	Passed

	2dC92099B59c975255ecE66784F"			
transferFrom	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","0x153b057d5d7262dC92099B59c975255ecE66784F",1000000	true	0x7988eebaedd07d3185c7ff30c7e0fa7c117963b920786c97d3ad9c88f29da075	Passed
balanceOf	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	9000000	N/A	Passed
balanceOf	0x153b057d5d7262dC92099B59c975255ecE66784F	1000000	N/A	Passed
burn (from Owner)	90000000	true	0x4ed60c9963057e7eb517087c0f44fb85705d19e4e3f2fc52ea81adc84b039d23	Passed
balanceOf	0x1dd64394E29c5988f04A8E074D0DBACd4D614729	9999999999999999900000000	N/A	Passed
transfer (From owner)	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","10000000"	true	0xd45507635054515e09935dc8209084974b6035721ff0f76273fc8221196878a6	Passed
balanceOf	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	10000000	N/A	Passed
balanceOf	0x1dd64394E29c5988f04A8E074D0DBACd4D614729	9999999999999999990000000	N/A	Passed
mint (from Owner)	90000000	true	0x1c55435cceeaa1a70f9b89ab18a7898ac2eb100af45f6083c130fe514f0219b2	Passed
balanceOf	0x1dd64394E29c5988f04A8E074D0DBACd4D614729	9999999999999999990000000	N/A	Passed

Automated Tests

Slither

```
INFO:Detectors:
OwnersERC20.constructor(string,string,uint256)._name (GETLv2.sol#21) shadows:
        - ERC20._name (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#42) (state variable)
OwnersERC20.constructor(string,string,uint256)._symbol (GETLv2.sol#21) shadows:
        - ERC20._symbol (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#43) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (openzeppelin-contracts/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (GETLv2.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin-contracts/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter OwnersERC20.mint(uint256)._amount (GETLv2.sol#31) is not in mixedCase
Parameter OwnersERC20.burn(uint256)._amount (GETLv2.sol#42) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
mint(uint256) should be declared external:
        - OwnersERC20.mint(uint256) (GETLv2.sol#31-36)
burn(uint256) should be declared external:
        - OwnersERC20.burn(uint256) (GETLv2.sol#42-47)
name() should be declared external:
        - ERC20.name() (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#113-116)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#121-123)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#150-164)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#178-181)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#197-205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GETLv2.sol analyzed (5 contracts with 75 detectors), 24 result(s) found
```

SOLHINT LINTER

```
======SOLHINT======


GETLv2.sol
   1:1  error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement                          compiler-version
  21:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility

✖ 2 problems (1 error, 1 warning)
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the **GELT** platform. We performed our audit according to the procedure described above.

The audit showed several informational severity issues. In the end, all issues were fixed by the Auditee

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **GELT** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **GELT** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
December, 2021

For

GALILEO
EXCHANGE

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com