VIDMA

EG

SMART CONTRACT AUDIT

Project: EG
Date: December 14th, 2022

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

EG is a community token that powers the EG Ecosystem.
The EG Smart Contract accumulates a 5% fee from every Buy and a 5% fee from every Sell transaction. No fee is collected from transfers in default setting but all fee is configurable and can be changed by the owner. All fee is accumulated in the Smart Contract.

The fee is then distributed to community-governed wallets and used for tech development, holder rewards, and EG's massive social impact initiatives around the world.

Smart Contract supports 5 wallet addresses: Marketing, Liquidity, Tech, Donations, and Staking Rewards. The owner has the ability to change a percentage amount for each wallet. The owner can call "withdrawTokens()" to remove tokens accumulated in the contract. These tokens are sent to all 5 wallets, based on the configured percentages.

There are whitelisted and blacklisted wallets in the EG token. Whitelisted wallets do not pay a fee on Buy or Sell. Blacklisted wallets cannot trade but can only transfer tokens to EG token contract owner. All holders are not whitelisted by default and pay a fee. For transfers, if one or both wallets are whitelisted, no fee is paid. The fee is only paid on transfer if none of the wallets is whitelisted.

The EG token contract implements Anti-rug protection and Anti-Bot Protection. Only the owner can create a liquidity pool. Trading will not actually be enabled for several minutes after the “enable trading function" is called. Max transaction amount will be some percentage of the supply, which can be confirmed by the maxTransactionCoolDownAmount function.

During the audit process, the Vidma team found several issues. A detailed summary and the current state are displayed in the table below.

Vulnerability Summary

Severity of the issue	Total found	Resolved	Unresolved
Critical	0 issues	0 issues	0 issues
High	0 issues	0 issues	0 issues
Medium	3 issues	3 issues	0 issues
Low	5 issues	4 issues	1 issue
Informational	1 issue	1 issue	0 issues
Total	**9 issues**	**8 issues**	**1 issue**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contracts are fully operational and secure. Under the given circumstances, we set the following risk level:

Vidma Points System

To set the codebase quality mark, our auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and risk level of the issues are all taken into consideration. The Vidma team has developed a transparent evaluation codebase quality system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Codebase quality: 97.20

Evaluating the **initial commit** and the **last commit with the fixes**, Vidma audit team set the following **codebase quality** mark.

Score

Based on the **overall result of the audit** and the state of the final reviewed commit, the Vidma audit team grants the following **score**:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level.

The test results and coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

EG

EG's mission is to leverage community action and blockchain technologies to grow a global movement that defies the status quo and makes profitability intrinsically linked to positive social impact.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contracts.

The audit was conducted from December 2, 2022 to December 9, 2022. The outcome is disclosed in this document. The review of the fixes was conducted from December 13 – to December 14, 2022.

The scope of work for the given audit consists of the following contract:

- EG Token;

The source code was taken from the following **source**:
https://github.com/EG-Ecosystem/eg-token/blob/main/EGToken.sol

Initial commit submitted for the audit:
36376c7cdefb86cba579b170a34476a48b903d26

Last commit reviewed by the auditing team:
a68183203cb54e6fd8a9b6dfb1e0ab377829ae22

WORKFLOW OF THE AUDITING PROCESS

Vidma audit team uses the most sophisticated and contemporary methods and well-developed techniques to ensure contracts are free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After the Audit kick-off, our security team conducts research on the contract's logic and expected behavior of the audited contract.

Documentation reading

Vidma auditors do a deep dive into your tech documentation with the aim of discovering all the behavior patterns of your codebase and analyzing the potential audit and testing scenarios.

The outcome

At this point, the Vidma auditors are ready to kick off the process. We set the auditing strategies and methods and are prepared to conduct the first audit part.

Phase 2: Manual part of the audit

Manual check

During the manual phase of the audit, the Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract. The initial commit as stated in the agreement is taken into consideration.

Static analysis check

Static analysis tools are used to find any other vulnerabilities in smart contracts that were missed after a manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Within the testing part, Vidma auditors run integration tests using the Truffle or Hardhat testing framework. The test coverage and the test results are inserted in the accompanying section of this audit report.

The outcome

Second interim report with the list of new issues found during the testing part of the audit process.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings in accordance with the severity level of the issues. (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on if they have been fixed by EG or not. The issues with “Not Relevant” status are left on the list of findings but are not eligible for the score points deduction.

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to 6 levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or the issue could result in a significant loss.

Example: Underflow/overflow, precisions, locked funds.

High

The issue significantly affects the ability of the contract to compile or operate. These are potential security or operational issues.

Example: Compilation errors, pausing/unpausing of some functionality, a random value, recursion, the logic that can use all gas from block (too many iterations in the loop), no limitations for locking period, cooldown, arithmetic errors which can cause underflow, etc.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: Absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but are more related to optimization of the codebase.

Example: Unused variables, inappropriate function visibility (public instead of external), useless importing of SCs, misuse or disuse of constant and immutable, absent indexing of parameters in events, absent events to track important state changes, absence of getters for important variables, usage of string as a key instead of a hash, etc.

Informational

Are classified as every point that increases onboarding time and code reading, as well as the issues which have no impact on the contract's ability to operate.

Example: Code style, NatSpec, typos, license, refactoring, naming convention (or unclear naming), layout order, functions order, lack of any type of documentation.

MANUAL REPORT

Inappropriate contract libraries are used

Medium | MM – 01 | Resolved

According to the documentation, the EG contract should be upgradable but simple non-upgradable contract libraries from openzeppelin are used.

Recommendation:

Consider using Ownable, Context, IERC20 contracts from upgradable contract package by openzeppelin: @openzeppelin/contracts-upgradable

Unchecked tokens transfer return value

Medium | MM – 02 | Resolved

Function *withdrawAlienTokens()* ignores return value by token transfer at L1102.

Recommendation:

Consider using *SafeERC20*, to ensure that the transfer return value is checked, as the contract is designed to support different tokens using the *SafeERC20* library will help safely operate with non-standard tokens which don't have return value on transfers.

Constructor is declared

Medium | MM – 03 | Resolved

Due to a requirement of the proxy-based upgradeability system, no constructors can be used in upgradeable contracts.

Recommendation:

Consider removing the constructor from the EG contract.

Floating pragma

Low | ML – 01 | Resolved

The current version of solc in contract EG is ^0.8.9 and it is better to lock the pragma to a specific version.

Contracts should be deployed with the same compiler version and flags they have tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation:

Lock pragma to a specific version.

Useless SafeMath library usage

Low | ML – 02 | Resolved

Starting from 0.8.0 solc compiler has implemented checks for overflows and underflows, so there is no need for use of SafeMath for arithmetic operations.

Recommendation:

Consider removing SafeMath usage to decrease bytecode and gas usage.

Local variable shadowing

Low | ML – 03 | Resolved

In the contract EG there is shadowing of the state variable *owner* in the function *EG.allowance()* which is also defined in the inherited contract ERC20Upgradeable.

Recommendation:

Consider renaming the local variables that shadow another component.

Comparison to a boolean constant

Low | ML – 04 | Resolved

There is a comparison to the boolean constant in the functions *EG.inTokenCheck()*, and *EG.tokenCheck()*.

Recommendation:

Avoid comparing boolean expressions to boolean literals to decrease gas usage for the function call and deployment.

Re-Audit:

Straight comparison of the bool type is more gas efficient than comparing uint8 or boolean constant. For example:

```
require(!_checkingTokens);
return _checkingTokens;
```

Usage of *tx.origin*

Low | ML – 05 | Unresolved

There is the usage of the *tx.origin* to influence a control flow decision. Note that using *tx.origin* for authentication leaves the contract vulnerable to a phishing-like attack. It is recommended to use *msg.sender* instead.

Recommendation:

Avoid using *tx.origin* for authentication.

Re-Audit:

The EG team uses a multi-signature wallet to manage the owner wallet and admin related functions.

Unspecified state variable visibility

Informational | MI – 01 | Resolved

It is best practice to set the visibility of state variables explicitly. The default visibility for balances, *_allowances*, *lastTransfer* in the contract EG (L138, L140, L149) is internal.

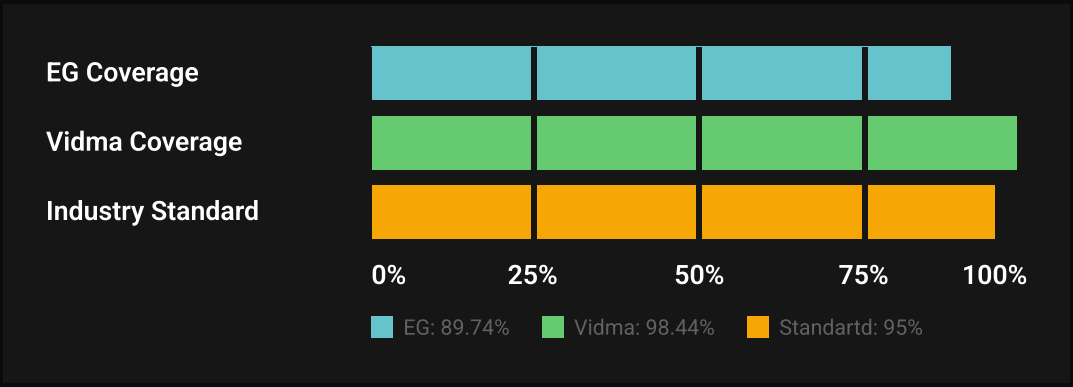
Recommendation:

Explicitly specify the visibility of state variables.

TEST RESULTS

To verify the security of contracts and their performance, a number of integration tests were carried out using the Hadrhat testing framework.

In this section, we provide both tests written by EG and tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the EG repository. We write totally separate tests with code coverage of a minimum of 95% to meet industry standards.

Tests Written by EG

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	89.74	56.64	90.20	89.72
EGToken.sol	89.74	56.64	90.20	89.72
All Files	**89.74**	**56.64**	**90.20**	**89.72**

Test Results

```
Contract: EGToken
  EGToken TestCase
    Deployment
      ✓ should set owner correctly (185ms)
      ✓ check the initialize function only once (192ms)
    Initialize
      ✓ Set maxTransactionAmount as 0% (216ms)
      ✓ Set maxTransactionCoolDownAmount as 0.1% (202ms)
      ✓ Set buyFee as 5% (94ms)
      ✓ Set sellFee as 5% (117ms)
      ✓ Set transferFee as 0% (141ms)
      ✓ Set marketingWalletFee as 20% (108ms)
      ✓ Set liquidityWalletFee as 20% (99ms)
      ✓ Set techWalletFee as 30% (108ms)
      ✓ Set donationsWalletFee as 10% (133ms)
      ✓ Set stakingRewardsWalletFee as 20% (99ms)
      ✓ Should set uniswap v2 router address correctly (134ms)
      ✓ Set owner balance as totalSupply (364ms)
    Check all functions
      ✓ transfer(address recipient, uint256 amount) (1064ms)
      ✓ Check transferFrom, approve (819ms)
```

```
    ✓ Check allowance(address ownerAddress, address delegate)
      delegate) (434ms)
    ✓ Check owner in whiteList (177ms)
    ✓ Check isTradingEnabled (100ms)
    ✓ Check inTradingStartCoolDown (93ms)
    ✓ Check setTradingEnabled (1312ms)
    ✓ Check setRouterAddress (436ms)
    ✓ Check setMarketingWallet (661ms)
    ✓ Check setMarketingWalletFee (354ms)
    ✓ Check setLiquidityWallet (628ms)
    ✓ Check setLiquidityWalletFee (362ms)
    ✓ Check setTechWallet (636ms)
    ✓ Check setTechWalletFee (294ms)
    ✓ Check setDonationsWallet (656ms)
    ✓ Check setDonationsWalletFee (325ms)
    ✓ Check setStakingRewardsWallet (670ms)
    ✓ Check setStakingRewardsWalletFee (265ms)
    ✓ Check setMaxTransactionAmount (640ms)
    ✓ Check setMaxTransactionCoolDownAmount (670ms)
    ✓ Check addClientsToWhiteList (441ms)
    ✓ Check removeClientsFromWhiteList (310ms)
    ✓ Check addClientsToBlackList (249ms)
    ✓ Check transferFrom the blacklist address to onwer (433ms)
    ✓ Check removeClientsFromBlackList (338ms)
    ✓ Check setBuyFee (477ms)
    ✓ Check setSellFee (405ms)
    ✓ Check setTransferFee (425ms)
    ✓ Check withdrawTokens (5747ms)
  EGToken TestCase for SWAP
    Check swap functions
    ✓ No one should be able to create a liquidity pool
      before (1072ms)
    ✓ Only the owner can create a liquidity pool (2694ms)
    ✓ No one should be able to create a liquidity pool an
      alternative liquidity pool outside the official
      liquidity (391ms)
    ✓ set trading enable by owner (326ms)
    ✓ isTradingEnabled is false and cooldown is true after
      trading started (307ms)
    ✓ start trading after startDelay time later (143ms)
    ✓ checking cooldown status is ture in cooldown time (139ms)
```

```
    ✓ Max transaction amount will be 0.01% of the supply; swap
      will fail because the swap token amount is bigger than
      the maxTransactionCoolDownAmount (1249ms)
    ✓ Max transaction amount will be 0.01% of the supply,
      swap is working well less than
      maxTransactionCoolDownAmount, checked sellFee (3288ms)
    ✓ Multiple trades in the others blockchain blocks are
      working well (2593ms)
    ✓ checking cooldown status is false after cooldown
      time (185ms)
    ✓ maxTransactionAmount is working after cooldown time (2600ms)
    ✓ token _transfer will be fail in case blacklist member (734ms)
    ✓ swap will be success after remove clients from
      blacklist (2789ms)

  57 passing (53s)
```

Tests Written by Vidma Auditors

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	98.44	93.16	100.00	98.42
EGToken.sol	98.44	93.16	100.00	98.42
All Files	**98.44**	**93.16**	**100.00**	**98.42**

Test Results

```
Contract: EGToken
  deploy
    ✓ cannot initialize twice (227ms)
    ✓ should set name, symbol, decimals and totalSupply
      correctly (499ms)
    ✓ should set maxTransactionCoolDownAmount correctly (186ms)
    ✓ should set buyFee, sellFee, transferFee correctly (318ms)
    ✓ should set marketingWalletFee correctly (90ms)
    ✓ should set liquidityWalletFee correctly (108ms)
    ✓ should set techWalletFee correctly (110ms)
    ✓ should set donationsWalletFee correctly (90ms)
    ✓ should set stakingRewardsWalletFee correctly (96ms)
  addClientsToWhiteList
    ✓ cannot add zero address (222ms)
    ✓ should add to the whiteList correctly (483ms)
  removeClientsFromWhiteList
    ✓ should remove from the whiteList correctly (439ms)
  addClientsToBlackList
    ✓ cannot add zero address (191ms)
    ✓ should add to the blackList correctly (339ms)
  removeClientsFromBlackList
    ✓ should remove from the blackList correctly (468ms)
```

```
transfer
  ✓ cannot transfer tokens to zero address (222ms)
  ✓ cannot transfer zero amount (210ms)
  ✓ cannot transfer if user tokens balance is insufficient (196ms)
  ✓ cannot transfer if sender is in the blacklist (454ms)
  ✓ cannot transfer if recipient is in the blacklist (518ms)
  ✓ cannot transfer if recipient and sender`s addresses
    are the same (236ms)
  ✓ should transfer tokens correctly (391ms)
  ✓ cannot transfer if trading is not enabled (770ms)
  ✓ transfer after set trading enable (2416ms)
  ✓ cannot transfer when transfer amount > than
    maxTransactionCoolDownAmount (1659ms)
  ✓ should transfer when trading is enabled and operation is
    SELL (2374ms)
  ✓ should transfer when trading is enabled and operation is
    BUY (1782ms)
  ✓ should transfer when trading is enabled and operation is
    exclude fee (5625ms)
  ✓ cannot transfer after coolDown if amount <=
    maxTransactionAmount (1586ms)
  ✓ should transfer after coolDown (2242ms)
transferFrom
  ✓ cannot transfer tokens from zero address (196ms)
  ✓ cannot transfer tokens to zero address (215ms)
  ✓ cannot transfer zero amount (253ms)
  ✓ cannot transfer if tokens amount exceeds allowance (281ms)
  ✓ cannot transfer if user tokens balance is insufficient (472ms)
  ✓ cannot transfer if sender is in the blacklist (720ms)
  ✓ cannot transfer if recipient is in the blacklist (936ms)
  ✓ cannot transfer if recipient and sender`s addresses
    are the same (392ms)
  ✓ should transfer tokens correctly (649ms)
approve
  ✓ cannot approve tokens to zero address (208ms)
  ✓ should approve tokens correctly (344ms)
setRouterAddress
  ✓ only owner can set (239ms)
  ✓ cannot set zero address (160ms)
  ✓ should set correctly (282ms)
setMarketingWallet
  ✓ only owner can set (234ms)
```

```
    ✓ cannot set zero address (280ms)
    ✓ should set correctly (459ms)
  setMarketingWalletFee
    ✓ only owner can set (252ms)
    ✓ cannot set if fee > 100% (231ms)
    ✓ should set correctly (337ms)
  setLiquidityWallet
    ✓ only owner can set (234ms)
    ✓ cannot set zero address (222ms)
    ✓ should set correctly (316ms)
  setLiquidityWalletFee
    ✓ only owner can set (268ms)
    ✓ cannot set if fee > 100% (222ms)
    ✓ should set correctly (388ms)
  setTechWallet
    ✓ only owner can set (188ms)
    ✓ cannot set zero address (218ms)
    ✓ should set correctly (328ms)
  setTechWalletFee
    ✓ only owner can set (172ms)
    ✓ cannot set if fee > 100% (263ms)
    ✓ should set correctly (360ms)
  setDonationsWallet
    ✓ only owner can set (252ms)
    ✓ cannot set zero address (238ms)
    ✓ should set correctly (344ms)
  setDonationsWalletFee
    ✓ only owner can set (264ms)
    ✓ cannot set if fee > 100% (240ms)
    ✓ should set correctly (298ms)
  setStakingRewardsWallet
    ✓ only owner can set (212ms)
    ✓ cannot set zero address (157ms)
    ✓ should set correctly (301ms)
  setStakingRewardsWalletFee
    ✓ only owner can set (203ms)
    ✓ cannot set if fee > 100% (141ms)
    ✓ should set correctly (318ms)
  setMaxTransactionAmount
    ✓ only owner can set (237ms)
    ✓ should set correctly (752ms)
  setMaxTransactionCoolDownAmount
```

```
  ✓ only owner can set (237ms)
  ✓ cannot set 0 amount (232ms)
  ✓ cannot set if amount not less than maxTransactionAmount
    maxTransactionAmount (1189ms)
  ✓ should set correctly (811ms)
setBuyFee
  ✓ only owner can set (228ms)
  ✓ cannot set fee more than 100% (193ms)
  ✓ should set correctly (409ms)
setSellFee
  ✓ only owner can set (272ms)
  ✓ cannot set fee more than 100% (226ms)
  ✓ should set correctly (494ms)
setTransferFee
  ✓ only owner can set (267ms)
  ✓ cannot set fee more than 100% (161ms)
  ✓ should set correctly (299ms)
withdrawTokens
  ✓ only owner can withdraw (185ms)
  ✓ cannot withdraw 0 tokens (252ms)
  ✓ cannot withdraw if total fees is more than 100% (1044ms)
  ✓ cannot withdraw if marketing wallet is not set (2022ms)
  ✓ cannot withdraw if liquidity wallet is not set (2047ms)
  ✓ cannot withdraw if tech wallet is not set (2711ms)
  ✓ cannot withdraw if donations wallet is not set (2473ms)
  ✓ cannot withdraw if staking rewards wallet is not set (3736ms)
  ✓ should withdraw correctly (4967ms)
withdrawAlienTokens
  ✓ only owner can withdraw (287ms)
  ✓ cannot withdraw if token address is zero (267ms)
  ✓ cannot withdraw to zero address (173ms)
  ✓ cannot withdraw EG token (266ms)
  ✓ cannot withdraw 0 tokens (326ms)
  ✓ cannot withdraw more than contract balance (268ms)
  ✓ should withdraw correctly (1151ms)
withdrawNativeTokens
  ✓ only owner can withdraw (204ms)
  ✓ cannot withdraw if token address is zero (235ms)
  ✓ cannot withdraw 0 tokens (188ms)
```

```
    ✓ cannot withdraw more than contract balance (233ms)
    ✓ should withdraw correctly (2338ms)

  110 passing (1m)
```

We are delighted to have a chance to work with the EG team and contribute to your company's success by reviewing and certifying the security of your smart contracts.

The statements made in this document should be interpreted neither as investment or legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists. Our company is based in Ukraine, known for its strong engineering, cryptography, and cybersecurity culture.

Website: **vidma.io**
Email: **security@vidma.io**