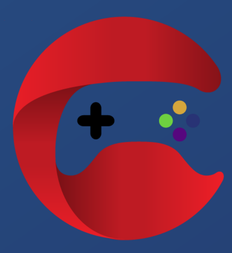
QuillAudits

Audit Report
February, 2022

For



RPGC token

Contents

Contents

Scope of the Audit

The scope of this audit was to analyse and document the RPGC 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 maths
- 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.

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	1	3	3

Introduction

During the period of **January 25, 2022 to January 31, 2022** - QuillAudits Team performed a security audit for RPGC token smart contracts.

The code for the audit was taken from following the official link:
Codebase: 440718a57a8e1e588a91dd8d0a5c091f9462e938
Fixed In: 2188d84c1b429aa31873346ca3f139380d1974308

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

- **unused function _transfer**

  _transfer is not been used anywhere

  Recommendation
  We recommend to use the transfer function while executing the transferToProxy and transferToExchange such that the blacklisted accounts can’t perform transfer/transferFrom.

  Status: Fixed

Low severity issues

- **Undeclared uint size**

  In multiple places the uint size has not been declared and used directly.

  Recommendation
  We recommend using uint256 instead of uint. Making the size of the data explicit reminds the reader how much data they've got to play with, which may help prevent or detect bugs.

  Status: Fixed

- **Used locked pragma version**

  The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.11 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

  pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.11
  pragma solidity 0.8.0; // good: compiles w 0.8.0 only but not the latest version
  pragma solidity 0.8.11; // best: compiles w 0.8.11

  Status: Fixed

- **Return value of this.transferFrom**

  The return value of an external transfer/transferFrom call is not checked.

  Recommendation
  Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

  Status: Fixed

Informational issues

- **Missing comments and description**

  Comments and Description of the methods and the variables are missing, it's hard to read and understand the purpose of the variables and the methods in context of the whole picture

  Recommendation
  Consider adding NatSpec format comments for the comments and state variables

  Status: Fixed

- **Unnecessary use of SafeMath**

  SafeMath library is imported and imposed on uint and not used anywhere. We recommend deleting the import and making the contract lighter.

  Status: Fixed

- **Public methods only being used externally**

  ‘public’ functions that are never used within the contract should be declared ‘external’ to save gas

  Recommendation
  Make these methods external
  transferToProxy, transferToExchange, removeAdmin, addTokenProvider, removeTokenProvider, mint, burn, ban, unban.

  Status: Fixed

Kovan Testnet Test Contract

RPGC Token: 0x498e7E41307917Df6Af44861ec0AFa81cb7b0C3c

- addAdmin
  - SuccessFully add admin PASS
  - Only Owner can add admin PASS

- removeAdmin
  - SuccessFully remove exist admin PASS
  - Only Owner can remove admin PASS

- addTokenProvider
  - SuccessFully add token provider PASS
  - Only Admin can add the token provider PASS

- transferToProxy
  - Transfer 1000 to exchanger PASS
  - Approve 500 to PRGC contract PASS
  - SuccessFully transfer from exchanger to tokenProvider PASS
  - Only valid tokenProvider can call this PASS

- removeTokenProvider
  - SuccessFully remove the present token provider PASS
  - Only Admin can remove the token provider PASS

- transferToExchange
  - transferFrom owner to exchange PASS
  - Only Owner can call this PASS

- ban
  - SuccessFully blacklist the account PASS
  - Only admin can blacklist accounts PASS

- unban
  - SuccessFully remove the blacklist account PASS
  - Only admin can blacklist accounts PASS

Automated Tests

Slither

```
Context._msgData() (Context.sol#20-22) is never used and should be removed
Context._msgSender() (Context.sol#10-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying w
ith 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

Context._msgData() (Context.sol#20-22) is never used and should be removed
ERC20._burn(address,uint256) (ERC20Standart.sol#373-388) is never used and should be removed
ERC20._mint(address,uint256) (ERC20Standart.sol#350-360) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying w
ith 0.6.12/0.7.6
Pragma version^0.8.0 (ERC20Standart.sol#3) necessitates a version too recent to be trusted. Consider deplo
ying 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

name() should be declared external:
        - ERC20.name() (ERC20Standart.sol#160-162)
symbol() should be declared external:
        - ERC20.symbol() (ERC20Standart.sol#168-170)
decimals() should be declared external:
        - ERC20.decimals() (ERC20Standart.sol#185-187)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20Standart.sol#192-194)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20Standart.sol#199-201)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20Standart.sol#211-214)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20Standart.sol#219-221)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20Standart.sol#230-233)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20Standart.sol#248-262)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20Standart.sol#276-279)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20Standart.sol#295-303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-dec
lared-external

RPGC.transferToProxy(address,uint256) (RPGC.sol#24-26) ignores return value by this.transferFrom(exchanger
,msg.sender,amount) (RPGC.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
Context._msgData() (Context.sol#20-22) is never used and should be removed
PauserRole._addPauser(address) (Roles.sol#198-201) is never used and should be removed
SafeMath.add(uint256,uint256) (SafeMath.sol#93-95) is never used and should be removed
SafeMath.div(uint256,uint256) (SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (SafeMath.sol#121-123) is never used and should be removed
SafeMath.sub(uint256,uint256) (SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying w
ith 0.6.12/0.7.6
Pragma version^0.8.0 (ERC20Standart.sol#3) necessitates a version too recent to be trusted. Consider deplo
ying with 0.6.12/0.7.6
Pragma version^0.8.0 (RPGC.sol#2) necessitates a version too recent to be trusted. Consider deploying with
 0.6.12/0.7.6
Pragma version^0.8.0 (Roles.sol#2) necessitates a version too recent to be trusted. Consider deploying wit
h 0.6.12/0.7.6
Pragma version^0.8.0 (SafeMath.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

name() should be declared external:
        - ERC20.name() (ERC20Standart.sol#160-162)
symbol() should be declared external:
        - ERC20.symbol() (ERC20Standart.sol#168-170)
decimals() should be declared external:
        - ERC20.decimals() (ERC20Standart.sol#185-187)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ERC20Standart.sol#192-194)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20Standart.sol#199-201)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20Standart.sol#219-221)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20Standart.sol#230-233)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20Standart.sol#248-262)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20Standart.sol#276-279)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20Standart.sol#295-303)
transferToProxy(address,uint256) should be declared external:
        - RPGC.transferToProxy(address,uint256) (RPGC.sol#24-26)
transferToExchange(address,uint256) should be declared external:
```

```
transferToExchange(address,uint256) should be declared external:
        - RPGC.transferToExchange(address,uint256) (RPGC.sol#28-31)
removeAdmin(address) should be declared external:
        - RPGC.removeAdmin(address) (RPGC.sol#38-41)
addTokenProvider(address) should be declared external:
        - RPGC.addTokenProvider(address) (RPGC.sol#43-46)
removeTokenProvider(address) should be declared external:
        - RPGC.removeTokenProvider(address) (RPGC.sol#48-51)
mint(uint256) should be declared external:
        - RPGC.mint(uint256) (RPGC.sol#53-55)
burn(uint256) should be declared external:
        - RPGC.burn(uint256) (RPGC.sol#57-59)
ban(address) should be declared external:
        - RPGC.ban(address) (RPGC.sol#61-64)
unban(address) should be declared external:
        - RPGC.unban(address) (RPGC.sol#66-69)
owner() should be declared external:
        - Ownable.owner() (Roles.sol#68-70)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Roles.sol#94-97)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Roles.sol#103-105)
addSigner(address) should be declared external:
        - SignerRole.addSigner(address) (Roles.sol#151-153)
renounceSigner() should be declared external:
        - SignerRole.renounceSigner() (Roles.sol#158-160)
renouncePauser() should be declared external:
        - PauserRole.renouncePauser() (Roles.sol#194-196)
paused() should be declared external:
        - Pausable.paused() (Roles.sol#226-228)
pause() should be declared external:
        - Pausable.pause() (Roles.sol#249-252)
unpause() should be declared external:
        - Pausable.unpause() (Roles.sol#257-260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

AdminRole._addAdmin(address) (Roles.sol#294-297) is never used and should be removed
AdminRole._removeAdmin(address) (Roles.sol#299-302) is never used and should be removed
Context._msgData() (Context.sol#20-22) is never used and should be removed
PauserRole._addPauser(address) (Roles.sol#198-201) is never used and should be removed
TokenProviderRole._addTokenProvider(address) (Roles.sol#336-339) is never used and should be removed
TokenProviderRole._removeTokenProvider(address) (Roles.sol#341-344) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.8 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.8 (Roles.sol#2) 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

owner() should be declared external:
        - Ownable.owner() (Roles.sol#68-70)
```

```
owner() should be declared external:
        - Ownable.owner() (Roles.sol#68-70)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Roles.sol#94-97)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Roles.sol#103-105)
addSigner(address) should be declared external:
        - SignerRole.addSigner(address) (Roles.sol#151-153)
renounceSigner() should be declared external:
        - SignerRole.renounceSigner() (Roles.sol#158-160)
renouncePauser() should be declared external:
        - PauserRole.renouncePauser() (Roles.sol#194-196)
paused() should be declared external:
        - Pausable.paused() (Roles.sol#226-228)
pause() should be declared external:
        - Pausable.pause() (Roles.sol#249-252)
unpause() should be declared external:
        - Pausable.unpause() (Roles.sol#257-260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-dec
lared-external

SafeMath.add(uint256,uint256) (SafeMath.sol#93-95) is never used and should be removed
SafeMath.div(uint256,uint256) (SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (SafeMath.sol#121-123) is never used and should be removed
SafeMath.sub(uint256,uint256) (SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.8 (SafeMath.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
. analyzed (27 contracts with 75 detectors), 104 result(s) found
ethsec@579a3a6018f2:/code/RPGC-Token$
```

Results

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

Closing Summary

Overall, smart contracts are very well written, documented, and adhere to guidelines. Several issues of Medium, Low severity and information issues have been reported and fixed by the team.
The contract is good to deploy to public EVM chains.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the RPGC 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 RPGC Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
February, 2022

For

RPGC token

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com