Smart Contract Security Audit Report for Gala.Games ERC20

AnChain.AI Inc.

Auditors: AnChain.AI Audit Team

Contact: Audit@AnChain.ai

Date: 2020-09-09

Index

1. Introduction

1.1 Audit Info

This document includes the results of the security audit for the client's smart contract code Gala.Games ERC20 as found in the section 3. The security audit was conducted by the AnChain.AI team from Sep 04, 2020, to Sep 09, 2020.

The purpose of this audit is to review source code, functionality on test net, and provide feedback on the design, architecture, and quality of the source code with an emphasis on validating the correctness and security of the software in its entirety.

1.2 Methodology

AnChain.AI team adopts a suite of tools and best practice from cybersecurity to audit smart contract source code for vulnerabilities.

3 key aspects in the AnChain.AI security audit methodology:

- Vulnerability audit. We use various tools to scan for known vulnerabilities, including our proprietary CAS sandbox.
- Statistical audit. Our sandbox will predict the risk score for the audited code with hundreds of thousands of mainnet smart contracts to understand security stance.
- Business logic audit. Our experts will design specific test cases to cover the key custom functions, to identify potential risk exposures.

1.3 Environment

Compiled Version: Solidity Compiler v0.6.12+commit.27d51765

Compiled System: macOS

Compiled Tool: Remix IDE, Truffle

2. Executive Summary

2.1 Key Findings

We **did not identify high severity vulnerabilities** that would compromise the integrity of the project. This ERC20 token smart contract **meets** the AnChain.AI smart contract auditing standards.

3. Smart Contract Scope

3.1 Audited Source Code File

File Name	Md5
Address.sol	`924db4c2ed840ce3a4a968a3e3c7b633`
Context.sol	`68e2547567157fd1daba6b64e9b39bd7`
ERC20.sol	`adf7d466b64969bf49a6023209943021`
ERC20Burnable.sol	`3f26d66d1782e796516bac2025c7e97b`
Gala.sol	`c9c6dc0f7c291c402190155c3e860e89`
IERC20.sol	`0a6d3d7703195ab5b26cac4ad6ef464d`
MinterRole.sol	`6c3505f09dfb1d4f16959f6c854df26d`
Roles.sol	`6bccbc8b6cfd8d80a42bd0ace68586ae`
SafeMath.sol	`72f5a806b2330b83f92380adb5e5a0f3`

4. Vulnerability Audit

4.1 Vulnerability Checklist

Type	Status
Numerical Overflow	PASS
Authorization Check	PASS
Transaction Ordering Dependency	PASS
Parity Multisig Bug	PASS
Random Number Practice	PASS
Re-entrancy	PASS
Function Visibility	PASS
Event Emitting	PASS
ERC1155 Token Formatting	PASS

5. Business Logic Audit

This section is dedicated to the functionality test of the audited smart contract. The purpose is to cover the main business logic for completeness and correctness.

The smart contract has been deployed on Testnet, with the specified environment and parameters.

Compile Test

tool	Remix IDE
version	solc v0.6.12+commit.27d51765
result	PASS

Code modification for compiling

file name	n/a
before	n/a
after	n/a

There is no code modification for this audit.

Deployment Test:

Network	Local Testnet
Result	PASS

5.1 addMinter

```
function addMinter(address account) public override onlyOwner;
```

description

Grant Minter authorization to ***account***.

specification

must emit MinterAdded event with parameter (***account***)	PASS
must reflect correct value changes	PASS
the caller must be contract owner	PASS
account must not be a minter	PASS

test cases

standard add minter	PASS
add minter without contract owner authority	PASS
add already existed minter	PASS

5.2 removeMinter

```
function removeMinter(address account) public onlyOwner;
```

description

Remove Minter authorization from ***account***.

specification

must emit MinterRemoved event with parameter (***account***)	PASS
must reflect correct value changes	PASS
the caller must be contract owner	PASS
account must be a minter	PASS

test cases

standard remove minter	PASS
remove minter without contract owner authority	PASS
remove non existed minter	PASS

5.3 renounceMinter

```
function renounceMinter() public;
```

description

Renounce the caller's Minter authorization.

specification

must emit MinterRemoved event with parameter (***msg.sender***)	PASS
must reflect correct value changes	PASS
the caller must have minter authorization	PASS

test cases

standard renounce minter	PASS
renounce minter without minter authority	PASS

5.4 approve

```
function approve(
    address spender,
    uint256 amount
) virtual override returns (bool);
```

description

Sets ***amount*** as the allowance of ***spender*** over the ***owner***'s tokens.

specification

must emit Approval event with parameter (***owner***, ***spender***, ***value***)	PASS
must reflect correct value changes	PASS

test cases

set approval for one spender	PASS
set approval for multiple spenders	PASS

5.5 increaseAllowance

```
function increaseAllowance(
    address spender,
    uint256 addedValue
) public virtual returns (bool);
```

description

Atomically increases the allowance granted to ***spender*** by the caller.

specification

must emit Approval event with parameter (***owner***, ***spender***, ***updated_allowance***)	PASS
must reflect correct value changes	PASS

test cases

increase allowance for one spender	PASS
increase allowance for multiple spenders	PASS

5.6 decreaseAllowance

```
function decreaseAllowance(
    address spender,
    uint256 subtractedValue
) public virtual returns (bool);
```

description

Atomically decreases the allowance granted to ***spender*** by the caller.

specification

must emit Approval event with parameter (***owner***, ***spender***, ***updated_allowance***)	PASS
must reflect correct value changes	PASS
subtractedValue must be less or equal to current allowance	PASS

test cases

decrease allowance for one spender	PASS
decrease allowance for multiple spenders	PASS
subtractedValue is greater than allowance	PASS

5.7 burn

```
function burn(uint256 amount) public virtual;
```

description

Destroy ***amount*** quantities of token from the caller balance.

specification

must emit Transfer event with parameter (***msg.sender***, ***address 0x0***, ***amount***)	PASS
must reflect correct value changes	PASS
total supply is decreased by ***amount***	PASS
amount must be less or equal to the caller balance	PASS

test cases

standard burn	PASS
burn more than owned quantity	PASS

5.8 burnFrom

```
function burnFrom(
    address account,
    uint256 amount
) public virtual;
```

description

Destroy ***amount*** quantities of token from the ***account*** balance.

specification

must emit Transfer event with parameter (***account***, ***address 0x0***, ***amount***)	PASS
must reflect correct value changes	PASS
must have sufficient allowance	PASS
total supply is decreased by ***amount***	PASS
amount must be less or equal to ***account*** balance	PASS

test cases

standard burn	PASS
burn more than owned tokens	PASS
burn more than allowance	PASS

5.9 mintBulk

```
function mintBulk(
    address[] memory accounts,
    uint256[] memory amounts
) public onlyMinter returns (bool);
```

description

Mint tokens with quantity extracted from each element of ***amounts[]*** to the corresponding element of ***accounts[]***.

specification

must emit Transfer event with parameter (***address 0x0***, ***accounts[i]***, ***amounts[i]***)	PASS
length of ***accounts[]*** must be equal to length of ***amounts***	PASS
authorization must be minter	PASS
quantity of minted tokens must not exceed cap	PASS
quantity of minted tokens must be greater than 0	PASS
must reflect correct value changes	PASS

test cases

standard mint	PASS
mint without minter authorization	PASS
mint tokens exceeding cap	PASS
mint 0 token	PASS
length of ***accounts*** is not equal to length of ***amounts***	PASS

5.10 transfer

```
function transfer(
    address recipient,
    uint256 amount
) public virtual override returns (bool);
```

description

Transfer ***amount*** quantity of tokens from the caller to ***recipient***.

specification

must reflect correct value changes	PASS
must not allow transfer to zero address	PASS
the caller must have sufficient balance to send token(s)	PASS
must emit Transfer event with (***msg.sender***, ***recipient***, ***amount***)	PASS

test cases

standard transfer tokens	PASS
send more than owned quantities	PASS
send to 0 address	PASS

5.11 transferFrom

```
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public virtual override returns (bool);
```

description

Transfer ***amount*** quantity of tokens from ***sender*** to ***recipient***.

specification

must reflect correct value changes	PASS
must not allow transfer from/to zero address	PASS
the caller must be an authorized operator of ***sender***	PASS
sender must have sufficient balance to send token(s)	PASS
the caller must have sufficient allowance to send token(s)	PASS
must emit Transfer event with (***sender***, ***recipient***, ***amount***)	PASS
must emit Approval event with (***sender***, ***msg.sender***, ***updated_allowance***)	PASS

test cases

standard transferFrom	PASS
send tokens more than the balance of ***sender***	PASS
transfer with allowance	PASS
transfer exceeding allowance	PASS
send from/to zero address	PASS

Recommendation

It is recommended to forbid function call with zero values to eliminate meaningless transactions, depending on operation scenario of the project team.

6. Gas Consumption Analysis

We tested gas consumptions of main functions, and all results are within normal gas consumption range.

Function Name	Gas Consumption
addMinter	~25,000
removeMinter	~20,000
renounceMinter	~15,000
approve	30,000~45,000
increaseAllowance	~30,000
decreaseAllowance	~30,000
burn	~35,000
burnFrom	30,000~45,000
mintBulk	~25,000 + 15,000/iteration
transfer	~50,000
transferFrom	~50,000

7. Conclusion

The audited contract source code implemented the standard erc20 & erc20 burnable token contract[1], with additional implementation of Address library & Roles library. There is no vulnerability found in this contract.

8. Appendix

8.1 Vulnerability Technical Explanation

These vulnerability scannings are powered by the patented AnChain.AI CAS (Smart Contract Auditing Sandbox), including static analysis, dynamic analysis, and statistical analysis.

- **Numerical Overflow**

Severity Level: high

In ETH, failing to run a numerical check on arithmetic operations may cause the values to overflow or underflow, which may cause crypto asset loss. It is a good practice to use "SafeMath" library for all the numeric operations to take care of overflow and underflow issue of all the numeric operations.

The code provided pass the 'numerical overflow' vulnerability check.

- **Authorization**

Severity Level: high

Parameters passed into a function should be consistent with the actual caller.

The best practice is to use 'require()' to check if the user executing the action matches the provided parameter.

The code provided pass the 'Authorization' vulnerability check.

- **Transaction Ordering Dependency:**

Severity Level: high

In ETH, timestamp of a transaction getting approved can be manipulated by vicious minors. Transaction Ordering Dependency refer to critical logic fault if minors approve the later submitted transaction PRIOR than the earlier ones.

Transaction ordering dependency does not have critical impact on this contract.

- **Parity Multisig Bug/ Delegate Call:**

Severity Level: High

The code does not call any external contracts and thus there is no issue about parity multisig bug.

- **Random Number**

Severity Level: High

Random number generator algorithm should not use predictable seeds.

The code does not generate or use any random number, and thus there is no issue about the random number issue.

The code provided pass the "Random Number" vulnerability check.

- **Re-entrancy Attack:**

Severity Level: High

The code does not contain any function that is vulnerable to re-entrancy attack.

- **Function Visibility:**

Severity Level: Low

The code does not contain any function that is improperly set with visibility.

- **Event Emitting:**

Severity Level: Low

The ERC code has emit transfer & approval event properly, which meets a ERC token standard.

- **ERC Token Standard:**

Severity Level: Low

The token contract has the following variables: name, symbol, decimal, totalSupply, which meets the ERC token standard.

8.2 Severity Level

Level	Description
High	The issue poses an existential risk to the project, and the issue identified could lead to massive financial or reputational repercussions.
Medium	The potential risk is large, but there is some ambiguity surrounding whether or not the issue would practically manifest.
Low	The risk is small, unlikely, or not relevant to the project in a meaningful way.
Code Quality	The issue identified does not pose any obvious risk, but fixing it would improve overall code quality, conform to recommended best practices, and perhaps lead to fewer development issues in the future.

8.3 Reference

[1]. OpenZeppelin ERC20 documentation. Retrieved from https://docs.openzeppelin.com/contracts/2.x/api/token/erc20

[2]. OpenZeppelin ERC20 source code. Retrieved from https://github.com/OpenZeppelin/openzeppelin-contracts/tree/9b3710465583284b8c4c5d2245749246bb2e0094/contracts/token/ERC20

9. Disclaimer

Disclaimer - Smart Contract Auditing - ETH

AnChain.ai makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and AnChain.ai specifically disclaims all implied warranties or merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

AnChain.ai will not be liable for any lost profits, business, contracts, revenues, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand against the company by any other party. If no event will AnChain.ai be liable for consequential, incidental, special, indirect, or exemplary damages arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if AnChain.ai has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the client's team and only the source code AnChain.ai notes as being within the scope of AnChain.ai's review within this report. This report does not include an audit of the deployment scripts used to deploy the contracts in the repository corresponding to this audit. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. In this report you may through hypertext or other computer links, gain access to websites operated by persons other than AnChain.ai. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' owners. You agree that AnChain.ai is not responsible for the content or operation of such websites and that AnChain.ai shall have no liability to your or any other person or entity for the use of third party websites. AnChain.ai assumes no responsibility for the use of third-party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.