QuillAudits

AUDIT REPORT

January 2025

For

GAMERGE

Table of Content

Executive Summary

Project Name	Gamerge
Overview	Gamerge is a DeFi project featuring a capped ERC20 token with burn and permit extensions, and a unique symbol update mechanism. It mints 100 million tokens to the deployer upon creation. Owners can update the token symbol, enhancing flexibility and branding opportunities. Built on OpenZeppelin contracts, it includes ERC20 standard functionalities, burnability for token scarcity, and EIP-2612 permit for gasless transactions. Ownership is securely managed via a two-step transfer process.
Audit Scope	The Scope of the Audit is to analyse the security and Correctness of Gamerge Token Contract.
Contracts in Scope	*https://github.com/gamerge-ai/contracts/blob/main/packages/foundry/contracts/Gamerge.sol*
Commit Hash	786d4ccfb4e9ac134034bd766914e8b01ae9c90d
Language	solidity
Blockchain	BSC
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	4th February 2025
Updated Code Received	5th February 2025
Review 2	5th February 2025
Fixed In	c472b2aa878b7dfa2111466ec71a69eb9a72b6e3

Number of Issues per Severity

1

Total Issues

🟥	High	0 (0%)
🟧	Medium	0 (0%)
🟩	Low	0 (0%)
🟪	Informational	1 (100%)

Severity

Issues

	🟥 High	🟧 Medium	🟩 Low	🟪 Informational
Open	0	0	0	0
Resolved	0	0	0	1
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ✓ Access Management
- ✓ Arbitrary write to storage
- ✓ Centralization of control
- ✓ Ether theft
- ✓ Improper or missing events
- ✓ Logical issues and flaws
- ✓ Arithmetic Computations Correctness
- ✓ Race conditions/front running
- ✓ SWC Registry
- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC's conformance
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls

Checked Vulnerabilities

- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Multiple Sends
- ✓ Using suicide
- ✓ Using delegatecall
- ✓ Upgradeable safety
- ✓ Using throw
- ✓ Using inline assembly
- ✓ Style guide violation
- ✓ Unsafe type inference
- ✓ Implicit visibility level

Techniques and Methods

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

- The overall quality of code
- Use of best practices
- Code documentation and comments, match logic and expected behavior
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper
- Implementation of ERC 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 analyzed 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

A 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.

Techniques and Methods

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 analyzed, 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 behavior 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, Foundry, Solhint, Mythril, Slither, Solidity statistic analysis.

Types of Severity

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.

🟥 High Severity Issues

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 Severity Issues

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 Severity Issues

Low-level severity issues can cause minor impact and 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 four 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.

Types of Issues

Open Security vulnerabilities identified that must be resolved and are currently unresolved.	**Resolved** These are the issues identified in the initial audit and have been successfully fixed.
Acknowledged Vulnerabilities which have been acknowledged but are yet to be resolved.	**Partially Resolved** Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Informational Issues

1. No fixed Solidity Version

Resolved

Path
Gamerge.sol

Description
Locking the pragma solidity version helps to ensure that contracts do not accidentally get deployed using, for example, an outdated or newer compiler version that might introduce bugs that affect the contract system negatively.

Recommendation
 It is recommended to use 0.8.20 which has been tested rather than deploying with a floating version.

Note
Incase if Gamerge Team has no plan to change the symbol in Near Future then ,The setSymbol function and related code can be removed since:

- The symbol is finalized as "GMG" in the constructor
- OpenZeppelin's ERC20 already handles symbol storage/retrieval

Functional Tests

Some of the tests performed are mentioned below:

- ✓ Should get the name of the token
- ✓ should get the symbol of the token
- ✓ Only the Owner should be able to change the symbol
- ✓ Should Revert if anyone tries to change the symbol expect owner
- ✓ should get the decimal of the token
- ✓ should get the Cap of the token
- ✓ should get the total supply of the token when deployed
- ✓ should get balance of the owner when contract is deployed
- ✓ should transfer tokens to other address
- ✓ should approve another account to spend token
- ✓ Current owner should be able to transfer ownership
- ✓ The new owner should be able to accept ownership.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. 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 Gamerge. We performed our audit according to the procedure described above.

One issue of informational severity was found. Some suggestions, gas optimizations and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Gamerge. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Gamerge. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of Gamerge to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

QuillAudits

6+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1K+** Projects Secured

Follow Our Journey

AUDIT REPORT

January 2025

For

GAMERGE

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillhash.com