Quantstamp

Mars4 Shoot ERC-20 Security Review

Executive Summary

This security review was prepared by Quantstamp, the leader in blockchain security.

Type	ERC-20 Token
Timeline	2024-03-29 through 2024-03-29
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• Etherscan Link
Auditors	• Shih-Hung Wang Auditing Engineer

Documentation quality	Undetermined
Test quality	Undetermined
Total Findings	5 **Acknowledged: 5**
High severity findings	0
Medium severity findings	0
Low severity findings	0
Undetermined severity findings	0
Informational findings	5 **Acknowledged: 5**

Summary of Findings

Quantstamp conducted a security review for the `ShootERC20` token developed in the Mars Battle project.

The `ShootERC20` token slightly modifies the ERC-20 standard contract written by OpenZeppelin. Upon creation, the `ShootERC20` contract mints a fixed number of `1_000_000_000` tokens to the contract deployer. The `ShootERC20` holders can transfer or approve another address to transfer their tokens, the same as a typical ERC-20 token. Additionally, the `ShootERC20` holders can burn an arbitrary number of their tokens up to their entire balance. As there is no further minting of the token after the contract creation, the total supply of `ShootERC20` can only decrease through time when a user burns their token.

All issues included in this report are informational in severity. MS-1 and MS-2 point out this project needs more documentation and tests. MS-3 points out a best practice of smart contract development. MS-4 describes a well-known issue of ERC-20 tokens, and MS-5 describes a concern about locked funds in the contract.

Although the additions to the base ERC-20 contract are minimal, the project has no tests, so the changes are not covered. Quantstamp, as always, strongly recommends adding tests to safeguard against unpredictable code. All issues in this report are considered acknowledged.

ID	DESCRIPTION	SEVERITY	STATUS
MS-1	Token Holders Can Burn Their Tokens	Informational	Acknowledged
MS-2	Lack of Specification and Test Suite	Informational	Acknowledged
MS-3	Unlocked Pragma	Informational	Acknowledged
MS-4	Allowance Double-Spend Exploit	Informational	Acknowledged
MS-5	Accidentally Transferred Funds to the Contract Are Locked	Informational	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope is the deployed contract at address 0xbc61e13ca6830fc7f035fd0e90a01cd08be6dcaa on Ethereum.

Files Included

ShootERC20.sol

Findings

MS-1 Token Holders Can Burn Their Tokens

• Informational Acknowledged

File(s) affected: `ShootERC20.sol`

Description: The `ShootERC20` contract has a public `burn()` function, which allows the token holders to burn an arbitrary number of their tokens up to their entire balance. As high-level documentation or code comments for the project are not presented, the intention of such a design choice needs to be clarified. This issue is to note that users and third-party developers need to know that such a public `burn()` function exists.

Recommendation: Consider clarifying the design choice of a public `burn()` function in user documentation.

MS-2 Lack of Specification and Test Suite

• Informational Acknowledged

File(s) affected: `ShootERC20.sol`

Description: Although the `ShootERC20` is a typical ERC-20 token with an additional self-burning feature, it is still recommended to document the use cases of the ERC-20 token, either in the code comments or on a public-facing website. For example, document each role in the system and the expected behavior or possible actions for each role.

Additionally, it is recommended that a test suite for smart contracts be developed before they are used in production. The test suite's branch coverage should be at least 90%.

Recommendation: Consider adding specifications, documents, and tests to the project to improve the overall quality and maintainability.

MS-3 Unlocked Pragma

• Informational Acknowledged

File(s) affected: `ShootERC20.sol`

Description: The `ShootERC20` contract specifies in the header a version number of the format `pragma solidity ^0.8.0;` The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler may use the specified version and above, hence the term "unlocked".

Recommendation: It is recommended that the caret be removed and the file be locked onto a specific Solidity version for consistency and to prevent using a version not tested during development.

Consider using the latest Solidity version, `v0.8.25`, at the time of writing for contracts deployed on Ethereum. For other EVM chains, an older version, e.g., `v0.8.19`, can be used to be compatible with them.

MS-4 Allowance Double-Spend Exploit

• Informational Acknowledged

File(s) affected: `ShootERC20.sol`

Description: As it is currently constructed, the `ShootERC20` contract is vulnerable to the ERC-20 allowance double-spend exploit, as are most of the known ERC-20 tokens.

Exploit Scenario:
1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` function on the token smart contract, passing Bob's address and `N` as the function arguments.
1. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` function again, this time passing Bob's address and `M` as function arguments.
2. Bob notices Alice's second transaction before it is mined and quickly sends another transaction that calls the `transferFrom()` function to transfer `N` Alice's tokens somewhere.
3. If Bob's transaction is executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will be able to transfer another `M` token.
4. Before Alice notices any irregularities, Bob calls the `transferFrom()` function again to transfer `M` Alice's tokens.

Recommendation: This is a well-known issue with ERC-20 tokens. We recommend that developers of applications dependent on `approve()` of ERC-20 tokens remember to set the allowance to 0 first and verify if it was used before setting the new value.

On the other hand, the exploit as described above can be mitigated through the use of functions that increase or decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`, which are supported by the `ShootERC20` contract.

MS-5
Accidentally Transferred Funds to the Contract Are Locked

Informational | Acknowledged

File(s) affected: `ShootERC20.sol`

Description: As a typical ERC-20 token, the `ShootERC20` contract does not have a particular function to retrieve funds accidentally transferred to itself. As a result, these accidentally transferred funds will be locked in the contract.

Recommendation: Consider whether this issue should be addressed. If so, consider allowing a contract owner to retrieve funds accidentally transferred to the contract. However, this approach will introduce a privileged role and increase the contract's centralization risks.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `fe6...3a7 ./ShootERC20.sol`

Automated Analysis

N/A

Changelog

- 2024-04-12 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp’s mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp’s team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp’s collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp