SMART CONTRACT AUDIT

Report for:	LOTT
Date:	07.10.2021 - 08.10.2021
Reaudit Date:	

HackCtrl

Table of Contents

Executive Summary

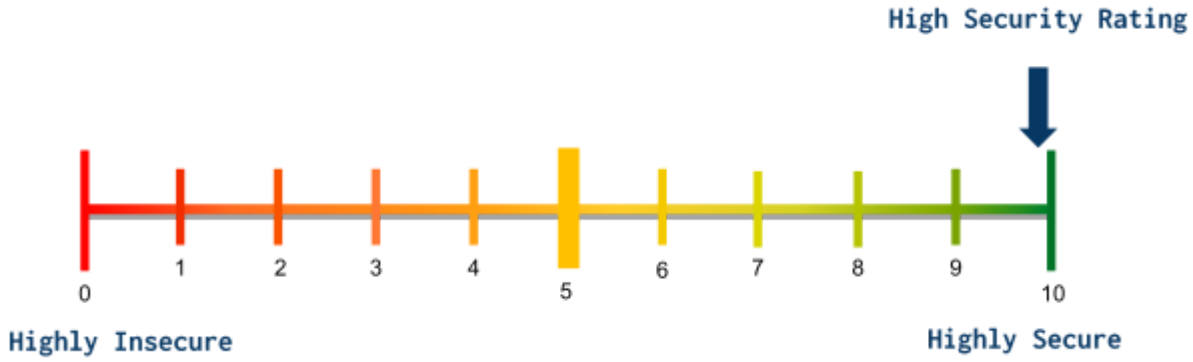
Hackcontrol (Provider) was contracted by **Lottrade** (Client) to carry out a smart contract audit.

The audit was conducted during **07.10.2021 - 08.10.2021**.

Objectives of the audits are the following:

1. Determine correct functioning of the contract, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

According to our research, after performing the audit, the security rating of the client's smart contract is **Highly secure**.



Scope

The Smart contract source code was taken from the Client's repository.

Repository - https://github.com/lottrade/smartcontract

Commit id - 1ccf358dd95c4812efb438414d814a9b5a4ba63c

The following list of information systems was in scope of the audit.

#	Name
1.	LOTT.sol

Project Tree

The following files have been checked within the audit process:

```
contracts
├─ LOTT.sol
```

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the contract's intended purpose by reading the available documentation.
2. Automated scanning of the contract with static code analysis tools for security vulnerabilities and use of best practice guidelines.
   - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix, Mythril and Solhint
   - we manually verify (reject or confirm) all the issues found by tools
3. Manual line by line analysis of the contracts source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
   - Reentrancy analysis
   - Race condition analysis
   - Front-running issues and transaction order dependencies
   - Time dependencies
   - Under- / overflow and math precision issues
   - Function visibility Issues
   - Possible denial of service attacks
   - Storage Layout Vulnerabilities
4. Report and remediate recommendation writing

HackCtrl

Severity Definition

The level of criticality of each vulnerability is determined based on the potential impact of loss from successful exploitation as well as ease of exploitation, existence of exploits in public access and other factors.

Severity	Description
High	High-level vulnerabilities are easy to exploit and may provide an attacker with full control of the affected systems, also may lead to significant data loss or downtime. There are exploits or PoC available in public access.
Medium	Medium-level vulnerabilities are much harder to exploit and may not provide the same access to affected systems. No exploits or PoCs are available for public access. Exploitation provides only very limited access.
Low	Low-level vulnerabilities exploitation is extremely difficult, or impact is minimal.
Info	Information-level vulnerabilities provide an attacker with information that may assist them in conducting subsequent attacks against target information systems or against other information systems, which belong to an organization.

Summary of Findings

The table below shows the vulnerabilities and their severity. A total of 2 issues were found.

Title	Severity
Solidity version update	Info
Pre-defined contract	Info

The overall quality of the code submitted for the audit is very good.

Best practice recommendations have largely been followed. Existing, the audited code has been used whenever possible in the form of the OpenZeppelin libraries (https://openzeppelin.com/). A safe math library has been used for arithmetic operations to avoid overflow and underflow issues. Code layout mostly follows the official Solidity style guide.

HackCtrl

Key Findings

Solidity version update

#1	Description
The solidity version should be updated.	

Evidences

The project utilizes solidity ^0.8.4. Though, the standard auditor's recommendation is to use the fixed version of Solidity, and to use the latest stable version.
Thus it is recommended to use fixed version solidity 0.8.9

Recommendations

It is recommended to use solidity 0.8.9.

Use predefined supply

#2	Description
Use constant for the initial supply	

Evidences

Since the token is not mintable, it is recommended to use the constant within the contract with total supply instead of the parameter passed. Such an action allows to avoid the mistakes during the deployment, passing incorrect amounts, zero amount or providing any other manual mistake.

Recommendations

Provide a public constant with the initial supply instead of passing the parameter.

Appendix A. Automated Tools

Scope	Tools Used
Smart-contracts Security	Mythril Solhint Slither Smartdec