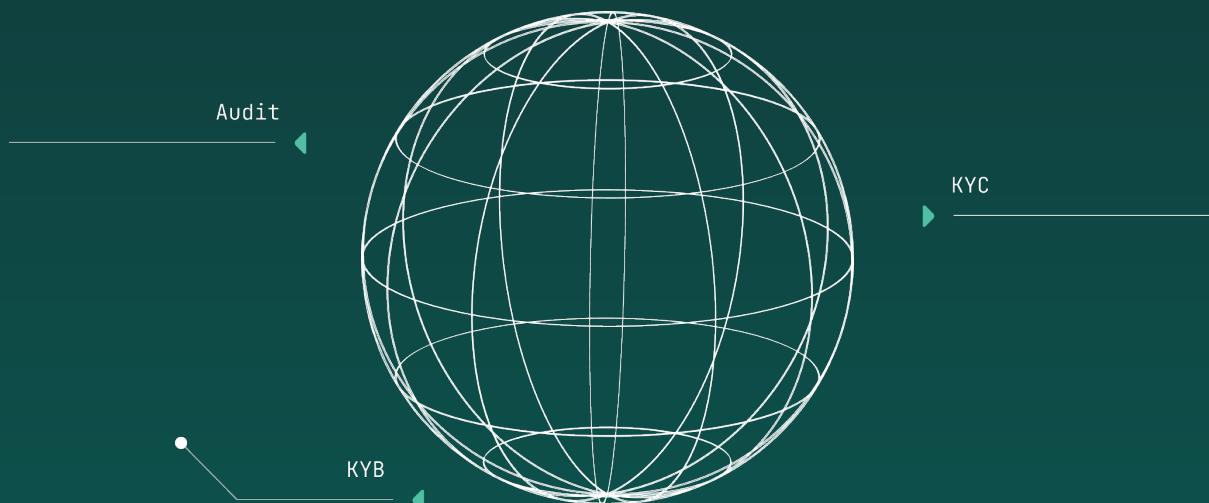




DAudit

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



CUSTOMER: MINTTY

DATE: May 18th, 2022



DISCLAIMERS

DAudit Disclaimer

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.

The smart contracts submitted for audit were examined in accordance with best industry practices at the time of this report in terms of cybersecurity vulnerabilities and issues in smart contract source code, which are detailed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no claims or guarantees about the code's security. It also cannot be deemed an adequate appraisal of the code's utility and safety, bug-free status, or any other contractual assertions. While we did our best in completing the study and generating this report, it is crucial to emphasize that you should not rely only on this report; we advocate doing many independent audits and participating in a public bug bounty program to assure smart contract security.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed

Technical Disclaimer

Smart Contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.



This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

DOCUMENT

Name	Smart contract code review and security analysis report for Mintty
Audit Team	DAudit.org team
Type	ERC1155 token
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/minttysm/smart-contract
Commit	a1f77167336685984496e74f66219d9b51a6d33e
Deployed Contract	https://bscscan.com/address/0xA5A9153dBA47cE8a86eDAee8be0b447dB24f0f8F
Technical Documentation	https://whitepaper.mintty.com/whitepaper-english/why-mintty
JS tests	No
Website	https://about.mintty.com/
Timeline	May 11th, 2022
Changelog	May 18th, 2022 - Initial audit

Table Of Content

Disclaimer.....2
Introduction.....5
Scope.....5
Executive Summary....7
Audit overview.....9
Conclusion.....10

INTRODUCTION

DAudit.org (Consultant) was contracted by Mintty (Customer) to conduct a Smart Contract Code Review and Security Analysis. Mintty (Customer) hired DAudit.org (Consultant) to do a Smart Contract Code Review and Security Analysis. This report details the conclusions of the Customer's smart contract security assessment and code review, which took place on May 18th, 2022.

SCOPE

The scope of the project is smart contracts in the repository:

Repository:

<https://github.com/minttysm/smart-contract>

Commit:

a1f77167336685984496e74f66219d9b51a6d33e

Technical Documentation: Yes

(<https://whitepaper.mintty.com/whitepaper-english/why-mintty>)

JS tests: No

Contracts:

MinttyCollectionManager.sol

MinttyNFT.sol

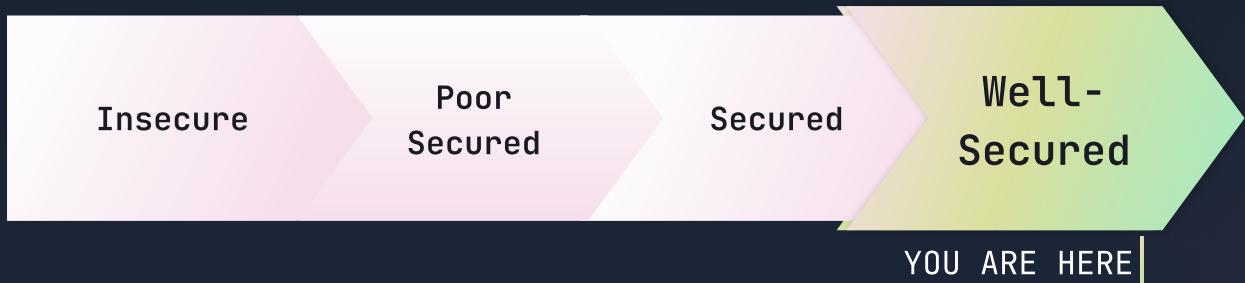
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:



Category	Check items
Code review	<ul style="list-style-type: none">▪ Reentrancy▪ Ownership Takeover▪ Timestamp Dependence▪ Gas Limit and Loops▪ DoS with (Unexpected) Throw▪ DoS with Block Gas Limit▪ Transaction-Ordering Dependence▪ Style guide violation▪ Costly Loop▪ ERC20 API violation▪ Unchecked external call▪ Unchecked math▪ Unsafe type inference▪ Implicit visibility level▪ Deployment Consistency▪ Repository Consistency▪ Data Consistency
Functional review	<ul style="list-style-type: none">▪ Business Logics Review▪ Functionality Checks▪ Access Control & Authorization▪ Escrow manipulation▪ Token Supply manipulation▪ Assets integrity▪ User Balances manipulation▪ Data Consistency manipulation▪ Kill-Switch Mechanism▪ Operation Trails & Event Generation

EXECUTIVE SUMMARY

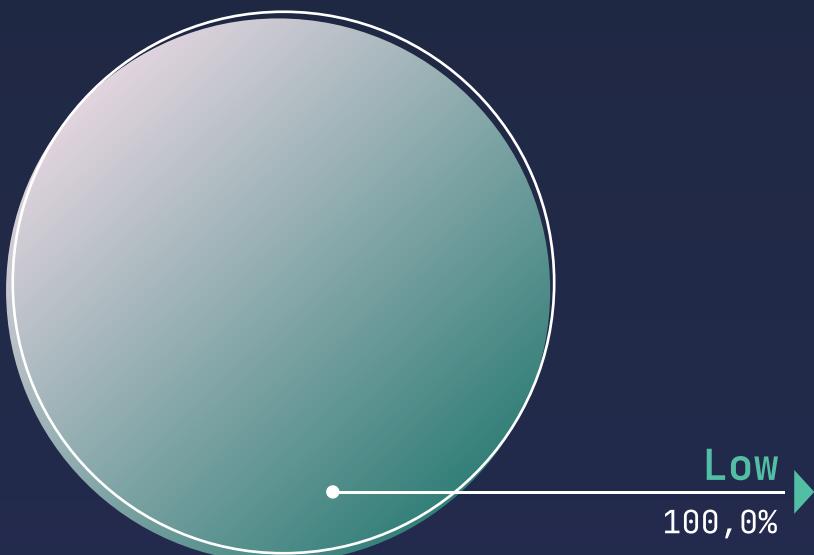
According to the assessment, the Customer's smart contracts are well-secured.



With Mythril, SmartCheck, Solgraph, and Slither, DAudit did a code analysis, manual audit, and automated checks. All concerns discovered during automated analysis were carefully examined, and the Audit summary section contains critical vulnerabilities. The audit summary section contains a list of all problems discovered.

Security engineers discovered one low-severity problems as a result of the audit.

Graph 1. The distribution of vulnerabilities after the audit.





Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to asset loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.

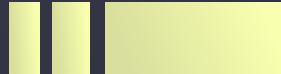
AUDIT OVERVIEW

Critical



- ▶ No critical issues were found.

High



- ▶ No critical issues were found.

Medium



- ▶ No critical issues were found.

Low



1. Detection of shadowing using local variables:

It is possible to use the same variable twice in Solidity, but it can lead to unintended side effects. This is especially difficult regarding working with multiple contracts.

In:

- MintyNFT.mint._uri
- MintyNFT.burn._owner
- MintyNFT.burnBatch._owner

Recommendation: Rename the local variables that shadow another component.

DECENTRALAB PTE.LTD.

160 Robinson Road, #14-04 Singapore
Singapore (068914)
support@daudit.org



DAudit

CONCLUSION

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** low severity issues.