



**Curate**

**XCUR Token (BSC)**

**SMART CONTRACT AUDIT**

**03.02.2022**

**Made in Germany by Chainsulting.de**



## Table of contents

1. Disclaimer.....	3
2. About the Project and Company .....	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level .....	6
4. Auditing Strategy and Techniques Applied.....	7
4.1 Methodology .....	7
4.2 Tested Contract Files .....	8
4.3 Used Code from other Frameworks/Smart Contracts .....	8
4.4 Metrics / CallGraph.....	9
4.5 Metrics / Source Lines & Risk.....	10
4.6 Metrics / Capabilities .....	11
4.7 Metrics / Source Unites in Scope .....	12
5. Scope of Work .....	13
5.1 Manual and Automated Vulnerability Test.....	14
5.1.1 Extensive Owner rights .....	14
5.1.2 Missing natspec documentation.....	15
5.1.3 A floating pragma is set.....	16
5.2. SWC Attacks .....	17
5.3 Verify claims .....	21
6. Executive Summary.....	22
7. Deployed Smart Contract .....	22
8. About the Auditor .....	23

## 1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Curate Group Ltd. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (28.02.2022)	Layout
0.4 (01.03.2022)	Automated Security Testing Manual Security Testing
0.5 (02.03.2022)	Verify Claims and Test Deployment
0.6 (02.03.2022)	Testing SWC Checks
0.9 (02.03.2022)	Summary and Recommendation
1.0 (03.03.2022)	Final document
1.1 (03.03.2022)	Added deployed contract

## 2. About the Project and Company

### Company address:

Curate Group Ltd  
50 Eastcastle St  
Fitzrovia London  
W1W 8EA United Kingdom

**Website:** <https://curate.style>

**Instagram:** <https://www.instagram.com/curateproject>

**Telegram:** <https://t.me/curate>

**Twitter:** <https://twitter.com/curateproject>

**YouTube:** <https://www.youtube.com/channel/UCVq646oBKp6CTFUSIHsfKAw>

**LinkedIn:** <https://www.linkedin.com/company/cur%CA%8Cte/>

**TikTok:** <https://www.tiktok.com/@curateproject>

**Medium:** <https://curate-xcur.medium.com>

**Facebook:** <https://www.facebook.com/CurateStyle>

**Coingecko:** <https://coingecko.com/en/coins/curate>

**CoinMarketCap:** <https://coinmarketcap.com/currencies/curate>



## 2.1 Project Overview

Curate is the world's first all-in-one marketplace app that uses blockchain technology as a payment infrastructure and rewards buyers/sellers on all transactions. Curate is built on its native marketplace, allowing buyers and sellers to exchange physical and digital goods such as gaming, electronics, NFTs, clothing, fashion, and more.

Curate utilizes a decentralized blockchain network as a means of providing a reward in the form of \$XCUR, our native token, to buyers and sellers on all successful sales. As well as this, Curate offers cryptocurrency payments as an optional form of payment outside traditional options such as credit/debit card and PayPal. Spending XCUR in-app offers zero gas fees as the gas fee amount is credited back to the buyer's wallet in XCUR.

The Curate platform is also rewarding the contributors. Every item or content posted by them will generate some reward for them in digital tokens. This will also serve as a form of motivation for them to sell and buy more on the marketplace.

### 3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

## 4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

### 4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
  - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
  - i. 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.
  - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## 4.2 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 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

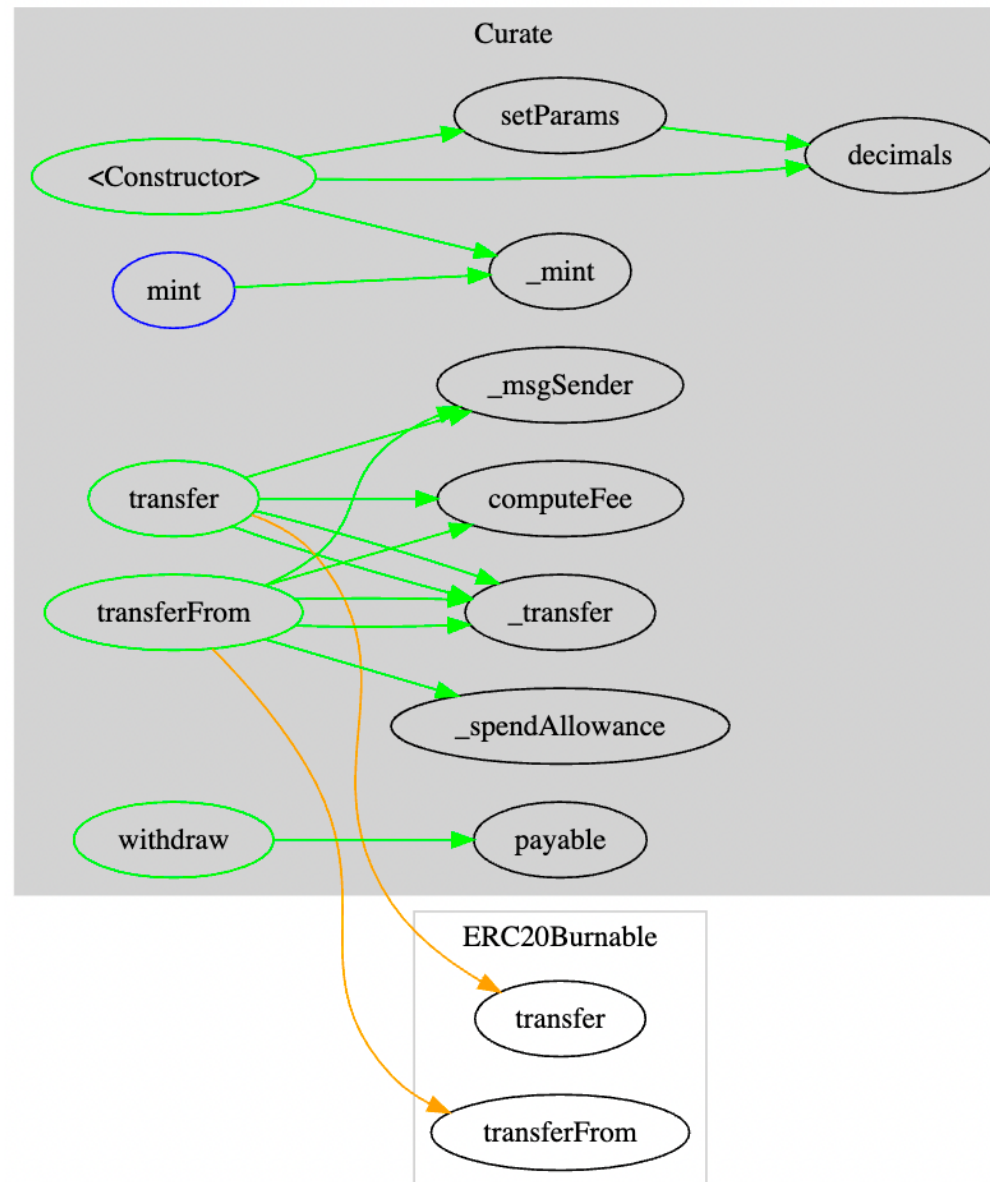
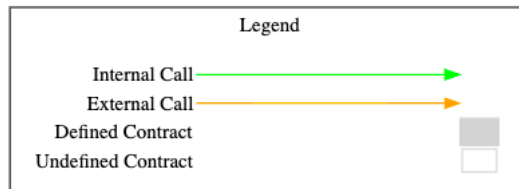
File	Fingerprint (MD5)
XCUR.sol	783c63eadb8e4bf864df0e3cada0e3a3

## 4.3 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/access/Ownable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/access/Ownable.sol</a>
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/extensions/ERC20Burnable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/extensions/ERC20Burnable.sol</a>

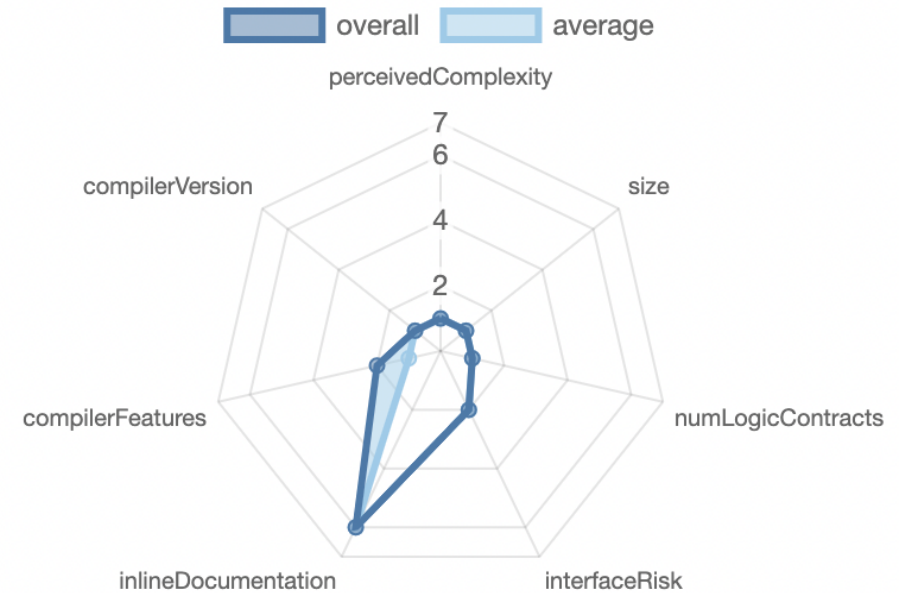
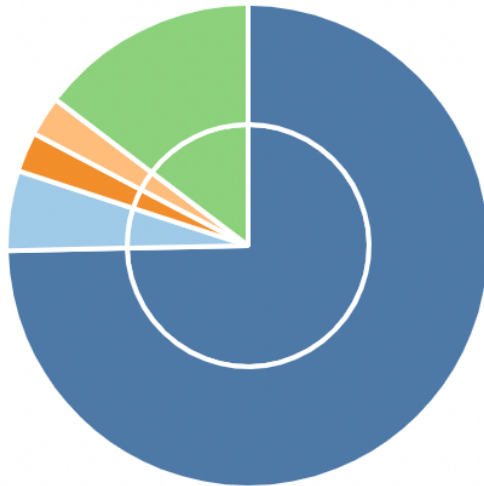


## 4.4 Metrics / CallGraph













## 4.5 Metrics / Source Lines & Risk

source comment single block mixed  
empty todo blockEmpty





## 4.6 Metrics / Capabilities


<b>Solidity Versions observed</b>	 <b>Experimental Features</b>	 <b>Can Receive Funds</b>	 <b>Uses Assembly</b>	 <b>Has Destroyable Contracts</b>	
<input type="text" value="^0.8.9"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>	
 <b>Transfers ETH</b>	 <b>Low-Level Calls</b>	 <b>DelegateCall</b>	 <b>Uses Hash Functions</b>	 <b>ECRecover</b>	 <b>New/Create/Create2</b>
<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

### Exposed Functions





This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 <b>Public</b>	 <b>Payable</b>				
6	0				
<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>	
1	7	0	0	2	

### StateVariables

<b>Total</b>	 <b>Public</b>
3	3

## 4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	XCUR.sol	1	_____	71	71	56	4	47	
	Totals	1	_____	71	71	56	4	47	

Legend: [ ]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

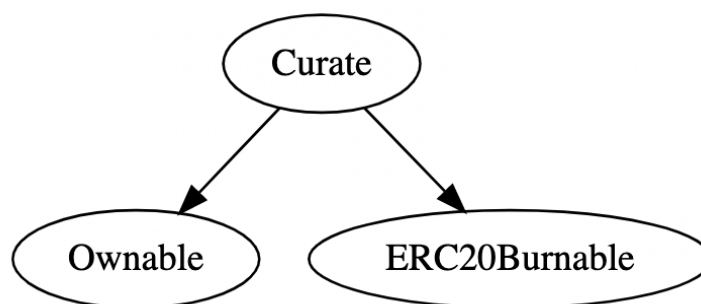
## 5. Scope of Work

The Curate Team provided us with the file that needs to be tested. The scope of the audit is the XCUR Token contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The BEP-20 Token standard is correctly implemented
- Deployer/Owner cannot mint any new Token
- Deployer/Owner cannot burn or lock user funds
- Deployer/Owner cannot pause the contract
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



## 5.1 Manual and Automated Vulnerability Test

### CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

### HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

### MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract

### LOW ISSUES

#### 5.1.1 Extensive Owner rights

Severity: LOW

Status: ACKNOWLEDGED

Code: NA

File(s) affected: XCUR.sol

Attack / Description	Code Snippet	Result/Recommendation
The owner has extensive rights. The auditor has not recognized any multi-sig structure.	Line: 57 <code>function mint(uint256 _amount) external onlyOwner</code>	The owner has rights to initiate minting of unlimited tokens and set a tax rate. If the wallet/private key gets into the wrong hands caused by a leak or hack, then it's easily possible to drain out user funds by setting a high fee or mint unlimited tokens and dump

	Line: 61 <pre>function setParams(uint256 _newBasisPoints, uint256 _newMaxFee, address _masterAccount) public onlyOwner</pre>	on the market. We recommend to protect the owner rights with a multi-signature structure such as gnosis safe or hardcode a MaxSupply and MaxFee rate.
--	---	---

#### 5.1.2 Missing natspec documentation

Severity: LOW

Status: ACKNOWLEDGED

Code: CWE-1056

File(s) affected: XCUR.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

## INFORMATIONAL ISSUES

5.1.3 A floating pragma is set.

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: SWC-103

File(s) affected: XCUR.sol

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is "^0.8.9". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	Line 1: <code>pragma solidity ^0.8.9;</code>	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee.  i.e. Pragma solidity 0.8.9







## 5.2. SWC Attacks

ID	Title	Relationships	Test Result
<a href="#">SWC-131</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	✓
<a href="#">SWC-130</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	✓
<a href="#">SWC-129</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	✓
<a href="#">SWC-128</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	✓
<a href="#">SWC-127</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	✓
<a href="#">SWC-125</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	✓
<a href="#">SWC-124</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	✓
<a href="#">SWC-123</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	✓


ID	Title	Relationships	Test Result
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	✓
<a href="#">SWC-121</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓
<a href="#">SWC-120</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	✓
<a href="#">SWC-119</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-118</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	✓
<a href="#">SWC-117</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓
<a href="#">SWC-116</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-115</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-114</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-113</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	✓
<a href="#">SWC-112</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-111</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-110</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	✓
<a href="#">SWC-109</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	✓
<a href="#">SWC-108</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-107</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	✓
<a href="#">SWC-106</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-105</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-104</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-103</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	
<a href="#">SWC-102</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	
<a href="#">SWC-101</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	
<a href="#">SWC-100</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	

## 5.3 Verify claims

5.3.1 The BEP-20 Token standard is correctly implemented


**Status:** tested and verified 

5.3.2 Deployer/Owner cannot mint any new Token


**Status:** tested and verified 

```
function mint(uint256 _amount) external onlyOwner {  
    _mint(msg.sender, _amount);  
}
```


5.3.3 Deployer/Owner cannot burn or lock user funds

**Status:** tested and verified 

5.3.4 Deployer/Owner cannot pause the contract

**Status:** tested and verified 

5.3.5 The smart contract is coded according to the newest standards and in a secure way.

**Status:** tested and verified 

## 6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found, after the manual and automated security testing. Only Low and Informational issue have been found.

## 7. Deployed Smart Contract

VERIFIED

<https://bscscan.com/address/0xd52669712f253CD6b2Fe8A8638F66ed726cb770C#code>

## 8. About the Auditor

Chainsulting is a professional software development firm based in Germany that provides comprehensive distributed ledger technology (DLT) solutions. Some of their services include blockchain development, smart contract audits and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secured the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Chainsulting currently secures [\\$100 billion](#) in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions tailored to the clients' evolving business needs.

The blockchain security provider brings the highest security standards to crypto and blockchain platforms, helping to foster growth and transparency within the fast-growing ecosystem.

Check our website for further information: <https://chainsulting.de>

### How We Work



**1** -----

#### PREPARATION

Supply our team with audit ready code and additional materials



**2** -----

#### COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



**3** -----

#### AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



**4** -----

#### FIXES

Your development team applies fixes while consulting with our auditors on their safety.



**5** -----

#### REPORT

We check the applied fixes and deliver a full report on all steps done.