



XCUR Token (curate.style)

SMART CONTRACT AUDIT

10.02.2021

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 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files	9
4.4 Metrics / CallGraph.....	10
4.5 Metrics / Source Lines	11
4.6 Metrics / Capabilities	12
4.7 Metrics / Source Unites in Scope.....	12
5. Scope of Work.....	13
5.1 Manual and Automated Vulnerability Test.....	14
5.2 Verify Claims	14
6. Executive Summary.....	17
7. Deployed Smart Contract	18



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 Publicity Clerks Ltd (Curate.Style). 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 (08.02.2021)	Layout
0.5 (09.02.2021)	Automated Security Testing Manual Security Testing
0.8 (09.02.2021)	Verify Claims
1.0 (10.02.2021)	Summary and Recommendation

2. About the Project and Company

Company address:

Publicity Clerks Ltd
Suite 202b, 55 Marble Arch Towers
London, W1W 8EA, GB

Website: <https://curate.style>

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

LinkedIn: <http://linkedin.com/company/curateproject>

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

Reddit: <https://reddit.com/r/curatestyle>

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

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

2.1 Project Overview

The CURATE decentralized fashion discovery platform offers a one-stop solution to people looking for fashion inspiration and ideas. CURATE platform provides a discovery platform for both men's and women's luxury fashion, and as time goes on we will boast of the best decentralized crypto-asset based fashion platform the world has to offer. The fact that we act as curators also means that we will have a huge library and collection of fashion trends from designers world over which also means that people have different unique styles to pick from. The CURATE platform is also rewarding the contributors. Every idea posted by them will generate some sort of reward for them in digital tokens. This will also serve as a form of motivation for them to post more bright ideas and quality contents. Of course, this does not mean that the reward is the driving force behind the contributor involving themselves in our project, but it serves as a means of commending their hard work.

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 Used Code from other Frameworks/Smart Contracts

1. SafeMath.sol (0.4.24)

<https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v2.0.0/contracts/math/SafeMath.sol>

2. ERC20.sol (0.4.24)

<https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v2.0.0/contracts/contracts/token/ERC20/ERC20.sol>

3. Ownable.sol (0.4.24)

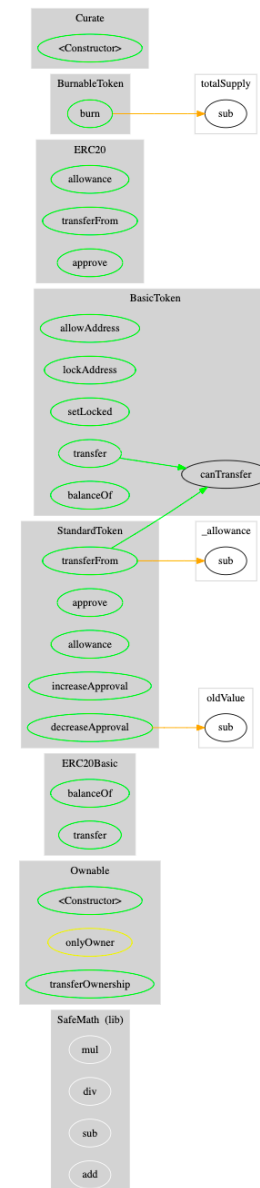
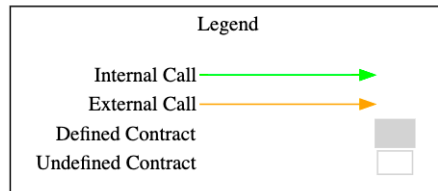
<https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v2.0.0/contracts/contracts/access/Ownable.sol>

4.3 Tested Contract Files

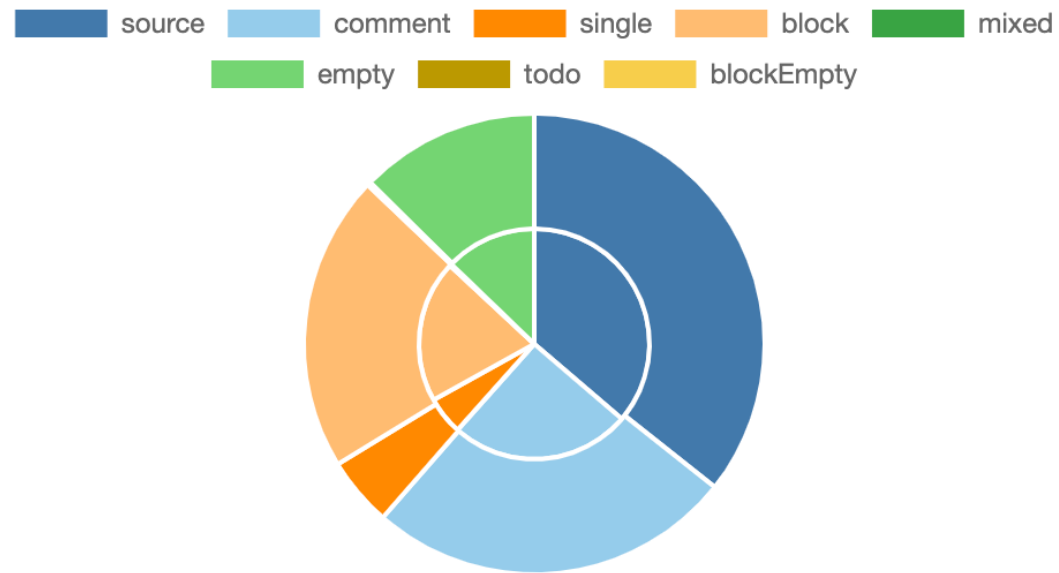
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
xcur_token.sol	cc311ca4a4d852965da2717164058618











4.4 Metrics / CallGraph





4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed		 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<div>^0.4.24</div>			<div></div>	**** (0 asm blocks)	<div></div>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRecover	 New/Create/Create2
<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	xcur_token.sol	8		304	278	136	106	111	
	Totals	8		304	278	136	106	111	

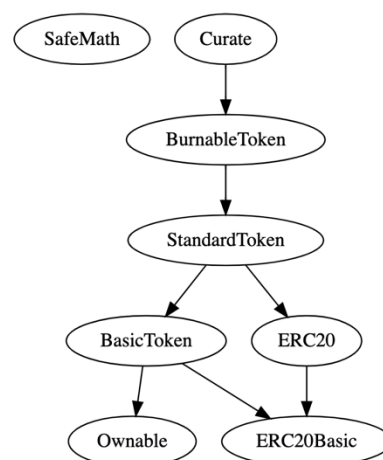
5. Scope of Work

The Curate team provided us with the files that needs to be tested. The scope of the audit is the ERC20 Token contract (XCUR)

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

- ERC-20 Token implementation
- Developer cannot mint any new tokens.
- Developer cannot burn or lock user funds
- Developer cannot pause the contract
- Project funds are locked via Team.finance
- Checking the overall security of the contracts

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

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

5.2 Verify Claims

5.2.1 ERC-20 Token implementation

5.2.2 Deployer address cannot mint any new tokens.

Status: tested and verified

Code: Ln 295

```
uint256 public constant initialSupply = 10000000 * (10 ** uint256(decimals));
```

5.2.3 Deployer address cannot burn or lock user funds ❌

Status: tested

Code: Ln 105 - 132

```
mapping(address => uint256) balances;
// allowedAddresses will be able to transfer even when locked
// lockedAddresses will *not* be able to transfer even when *not locked*
mapping(address => bool) public allowedAddresses;
mapping(address => bool) public lockedAddresses;
bool public locked = true;

function allowAddress(address _addr, bool _allowed) public onlyOwner {
    require(_addr != owner);
    allowedAddresses[_addr] = _allowed;
}

function lockAddress(address _addr, bool _locked) public onlyOwner {
    require(_addr != owner);
    lockedAddresses[_addr] = _locked;
}

function setLocked(bool _locked) public onlyOwner {
    locked = _locked;
}

function canTransfer(address _addr) public constant returns (bool) {
    if(locked){
        if(!allowedAddresses[_addr]&&_addr!=owner) return false;
    }else if(lockedAddresses[_addr]) return false;
```

```
    return true;  
}
```

5.2.4 Deployer address cannot pause the contract ✓

Status: tested and verified

5.2.5 Project funds are locked via Team.finance ✓

Status: tested and verified

<https://team.finance/view-coin/0xE1c7E30C42C24582888C758984f6e382096786bd?name=Curate&symbol=XCUR>

5.2.6 Checking the overall security of the contract ✓



6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debrief took place on the February 10, 2021. The overall code quality of the project is good, not overloaded with unnecessary functions, these is greatly benefiting the security of the contract. It correctly implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit, no issues were found after the manual and automated security testing.



7. Deployed Smart Contract

VERIFIED

XCUR Token

[0xE1c7E30C42C24582888C758984f6e382096786bd](#)

