



# SOLIDProof

**Blockchain Security | Smart Contract Audits**

MADE IN GERMANY

# Audit Passed

**Security Assessment**  
**19. June, 2021**

**For**

**bitccci<sup>®</sup>**

Disclaimer	4
Description	6
Project Engagement	6
Logo	6
Contract Link	6
Methodology	8
Used Code from other Frameworks/Smart Contracts (direct imports)	9
Source Lines	10
Risk Level	10
Capabilities	11
CallGraph	12
Source Units in Scope	12
Critical issues	13
High issues	13
Medium issues	13
Low issues	13
Informational issues	13
SWC Attacks	15
Comments	18



## Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

# Overview

## **Network**

Ethereum (ERC20)

## **Website**

<https://bitcci.ag/>

<https://bitcci.to/>

## **Telegram**

<https://t.me/bitcci>

## **Twitter**

<https://twitter.com/bitcci>

## **Facebook**

<https://www.facebook.com/bitcci>

## **Github**

<https://github.com/bitcci-AG/bitcci-cash>

## Description

They regulate, modernize and tokenize the oldest business of humanity.

bitcci liberates the sex industry from conflicts and problems and establishes a new, modern culture of acceptance, transparency, trust and free self-determination. They optimise, regulate and decentralise the entire industry, develop new business areas, introduce innovative technologies and make the industry and all its participants free and safe.

## Project Engagement

During the 16th of June, **bitcci Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. **bitcci Team** provided Solidproof.io with access to their code repository and whitepaper.

## Logo

The logo for bitcci, featuring the word "bitcci" in a bold, lowercase, sans-serif font, followed by a registered trademark symbol (®).

## Contract Link

TBA

Perversion:

<https://github.com/bitcci-AG/bitcci-cash/blob/master/contracts/bitcciCash.sol>

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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

# 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.

## **Methodology**

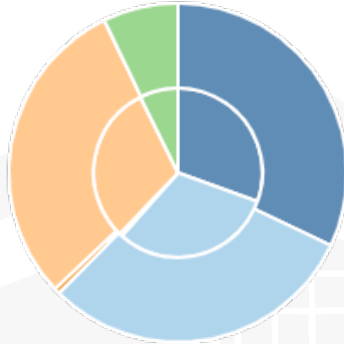
The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i) Review of the specifications, sources, and instructions provided to SolidProof 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 SolidProof 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.



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

■ source 
 ■ comment 
 ■ single 
 ■ block 
 ■ mixed 
 ■ empty 
 ■ todo 
 ■ blockEmpty



Dependency / Import Path	Count
@openzeppelin/contracts/GSN/Context.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol	1
@openzeppelin/contracts/utils/Context.sol	1





# Metrics







Source Lines

Risk Level

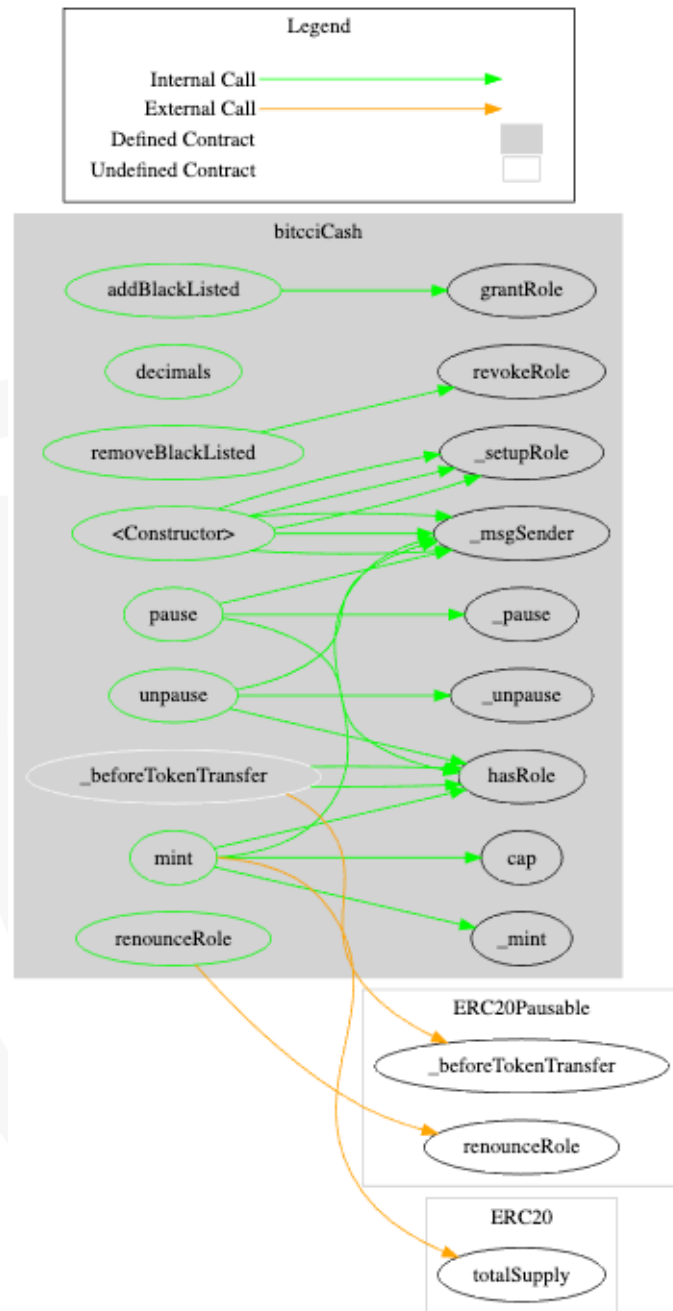


## Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
^0.8.0			**** (0 asm blocks)	

 Transfers ETH	 Low-Level Calls	 Delegate Call	 Uses Hash Functions	 ECRewriter	 New/Create/Create2
			Yes		

# CallGraph



## Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/bitcciCash.sol	1	————	126	122	54	55	68	
	Totals	1	————	126	122	54	55	68	

# Audit Results

# AUDIT PASSED

## Critical issues

- no critical issues found -

## High issues

- no high issues found -

## Medium issues

- no medium issues found -

## Low issues

- no low issues found -

## Informational issues

Issue	File	Type	Line	Description
#1	Main	Local variables shadowing (shadowing-local)	40	bitcciCash.constructor(string, string,uint8,uint256).name shadows: <ul style="list-style-type: none"><li>• ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol, Line: 60-62) (function)</li><li>• IERC20Metadata.name() (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol, Line: 16) (function)</li></ul>

#2	Main	Local variables shadowing (shadowing-local)	40	bitcciCash.constructor(string, string,uint8,uint256).symbol shadows: • ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70) (function) • IERC20Metadata.symbol() (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol, Line: 21) (function)
----	------	---	----	---

How to fix shadowing problem:

- Issue #1:
  - Replace “string memory name” (Line, see above) parameter for example to bitcciCashName (\_name doesn’t work here!) and don’t forget to pass this parameter to ERC20 constructor in same line
- Issue #2:
  - Replace “string memory symbol” (Line, see above) parameter for example to bitcciCashSymbol (\_symbol doesn’t work here!) and don’t forget to pass this parameter to ERC20 constructor in same line

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-13 1</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-13 0</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-12 9</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-12 8</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED
<a href="#">SW C-12 7</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	PASSED
<a href="#">SW C-12 5</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	PASSED
<a href="#">SW C-12 4</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	PASSED
<a href="#">SW C-12 3</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	PASSED
<a href="#">SW C-12 2</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	PASSED

<a href="#">SW C-12 1</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<a href="#">SW C-12 0</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	PASSED
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	PASSED
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	PASSED
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	PASSED
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	PASSED
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	PASSED
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	PASSED
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	PASSED



<a href="#">SW C-111</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-10 9</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-10 8</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-10 7</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-10 6</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW C-10 5</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW C-10 4</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW C-10 3</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW C-10 2</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW C-10 1</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-10</a> <a href="#">0</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
---	-----------------------------------	---	---------------

## Comments

bitcci cash token implements best practices in its smart contract.





*Solid  
Proofed*

**Blockchain Security | Smart Contract Audits**

  
MADE IN GERMANY