

# BTCCREDIT SMART CONTRACT FINAL AUDIT REPORT



by QuillAudits, July 2019

## **Introduction :**

This Audit Report highlights the overall security of [BtcCredit](#) Smart Contract. With this report, we have tried to ensure the reliability of their smart contract by complete assessment of their system's architecture and the smart contract codebase.

## **Auditing Approach and Methodologies applied :**

Quillhash team has performed thorough testing of the project starting with analysing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract in order to find any potential issues like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase we coded/conducted Custom unit tests written for each function in the contract to verify that each function works as expected. In Automated Testing, We tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests
4. Analysing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

## Audit Details

- Project Name: BTCCREDIT
- website/Etherscan Code : [Code](#)
- Languages: Solidity (Smart contract), Javascript (Unit Testing)

## Summary of BTCCREDIT Smart Contract :

QuillAudits conducted a security audit of a smart contract of BTCcredit. BTCcredit contract is used to create the **ERC20 utility token** which is a **BTCCredit token**, Smart contract contain basic functionalities of ERC20 token with total supply of 300 million and Mint function to mint more Btc credit tokens .

## Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient and working according to its specifications. The audit activities can be grouped in the following three categories:

**Security:** Identifying security related issues within each contract and within the system of contracts.

**Sound Architecture:** Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

**Code Correctness and Quality:** A full review of the contract source code. The primary areas of focus include:

- Correctness
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

## Security Level references :

Every issue in this report was assigned a severity level from the following:

**High severity issues** will bring problems and should be fixed.

**Medium severity issues** could potentially bring problems and should eventually be fixed.

**Low severity issues** are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

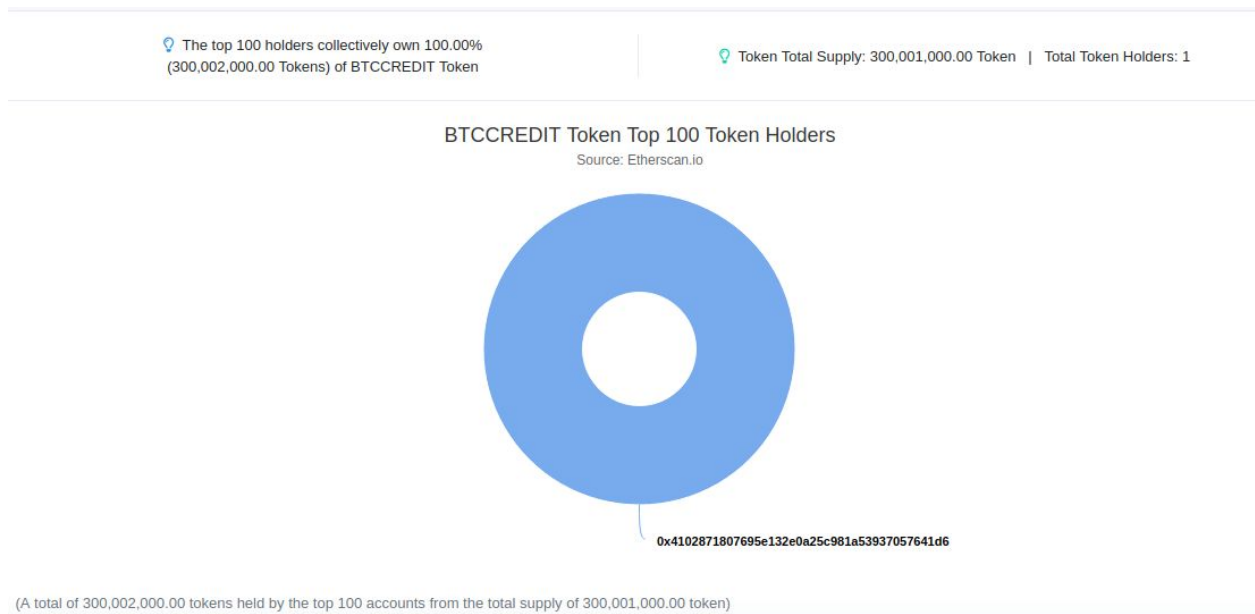
### Number of issues per severity

	Low	Medium	High
Open	0	1	0
Closed	2	1	1

## High severity issues:-

```
1 distributeTokens(address _address, uint _amount);
```

distributeTokens function of smart contract only check whether total supply is less than or equal to distributed tokens, but doesn't subtract from minted tokens, as minted tokens is already sent to particular address, hence owner will be able to distribute tokens more then total supply, as shown in the pictures below.



Total supply is : 300001000

0x410 is holding : 300002000

Rank	Address	Quantity	Percentage
1	<a href="#">0x4102871807695e132e0a25c981a53937057641d6</a>	300,002,000	100.0003%



You can use one variable that will automatically update when mint functions is called and will update the no of tokens minted, while distributing tokens check whether distributed tokens is less than the total supply subtracted from minted tokens.

1 Use transfer event while burning tokens, as etherscan reads transfer events. As now burn doesn't emit transfer function now, etherscan couldn't be able to read balance of an account that has burned all the tokens and still showing previous balance of that address.

<div> <div>Overview</div> <div>[ERC-20]</div> </div>		<div> <div>Profile Summary</div> </div>	
Total Supply:	1 0 BTC	Contract:	0x772981390e0bd3a72e7de6f3011e31769cba6e87
Holders:	1 addresses	Decimals:	18
Transfers:	4		

Transfers	Holders	Read Contract	Write Contract
-----------	---------	---------------	----------------

Token Holders Chart

A total of 1 token holder

First

<

Page 1 of 1

>

Last

Rank	Address	Quantity	Percentage
1	0x4102871807695e132e0a25c981a53937057641d6	300,002.000	0%

In both burn() and burnFrom() function of token contract.

**Status : Issues Fixed by Developer**

### **Low Severity Issues:-**

1 Solidity version must be fixed (Always use latest Version).

It should not `pragma solidity ^0.4.25;`

It should be `pragma solidity 0.4.25;`

**Status : Issues Fixed by Developer**

version should be fixed so that development phase and deployment phase should have the same solidity version.

2 SafeMath.sol contract functions access modifier can be changed to internal, as there is no use to make them public functions.

Use : access modifier internal in place of public.

**Status : Issues Fixed by Developer**

## Issues while checking initial audit fixes

### Medium Severity Issues:-

Total Supply:599,000,000 BTCC


Holders:2 addresses

Transfers:4

Contract:0x50970772981878114d3fb2129efe285b6f800a45


Decimals:18

DSP  
MUTUAL FUND



INVEST ONLINE IN  
DSP TAX  
SAVER FUND

INVEST NOW



TransfersHoldersRead ContractWrite Contract

Token Holders Chart

A total of 2 token holders

First<Page 1 of 1>Last

Rank	Address	Quantity	Percentage
1	0x4102871807695e132e0a25c981a53937057641d6	300,000,000	50.0835%
2	0x02afc30d49bbb0661b0c3c0e4a1b1d43bd648865	300,000,000	50.0835%

As you can see in the above picture, after minting 300 million tokens total supply was 600 million and, when 1 million tokens are burnt from account 1, transfer event was omitted from contract to address(0), but it should be from address (msg.sender/caller) to address(0) that's why percentage of token holder is higher than 100% (total supply).

**Status : Not Fixed**

# Unit Testing

## Test Suite

### Contract: BTCcredit Token Contracts

- ✓ Should correctly initialize constructor values of BTCCToken Token Contract (250ms)
- ✓ Should check the Total Supply of BTCCToken Tokens (80ms)
- ✓ Should check the Name of a token of BTCC Token contract (79ms)
- ✓ Should check the symbol of a token of BTCCToken contract (64ms)
- ✓ Should check the decimal of a token of BTCCToken contract (74ms)
- ✓ Should check the balance of a Owner (66ms)
- ✓ Should check the owner of a contract (57ms)
- ✓ Should check the New owner of a contract (57ms)
- ✓ Should check the balance of a contract (64ms)
- ✓ Should check the distributed tokens of a contract (68ms)
- ✓ Should check the owner account is not frozen (66ms)
- ✓ Should check the minted tokens (52ms)

✓ Should Not be able to distribute tokens to accounts[1] by Non Owner account (154ms)

✓ Should Not be able to distribute tokens more then total supply (170ms)

✓ Should be able to distribute tokens to accounts[1] (257ms)

✓ Should check the distributed tokens of a contract after sent to account[1] (62ms)

✓ Should Not be able to Freeze Accounts[1] by non owner account (263ms)

✓ Should Freeze Accounts[1] (210ms)

✓ Should Not be able to transfer Tokens when account is freezed (97ms)

✓ Should Not be able to UnFreeze Accounts[1] by non owner account (94ms)

✓ Should UnFreeze Accounts[1] (201ms)

✓ Should be able to transfer Tokens after Unfreeze (340ms)

✓ Should Not be able to burn tokens when account doesn't have tokens (96ms)

- ✓ Should be able to burn tokens (217ms)
- ✓ Should be able to emit transfer event while burn tokens (217ms)
- ✓ Should check the Total Supply of BTCCToken Tokens after token burnt
- ✓ Should Not be able to Mint Tokens by Non owner account (96ms)
- ✓ Should be able to Mint Tokens by owner only and send to accounts[3] (210ms)
- ✓ Should check the minted tokens after minted (44ms)
- ✓ Should Not be able to distribute tokens to accounts[6] less then total supply and minted tokens (127ms)
- ✓ Should be able to distribute tokens to accounts[6] less then total supply and minted tokens (192ms)
- ✓ Should check the Total Supply of BTCCToken Tokens after tokens minted (59ms)
- ✓ Should be able to transfer ownership of token Contract (160ms)
- ✓ Should check the New owner of a contract
- ✓ Should be able to Accept ownership of token Contract (124ms)

- ✓ Should check the New owner of a contract after ownership accepted
- ✓ should Approve address to spend specific token (155ms)
- ✓ Should be able to transfer Tokens approved by account[3] to account[5] (299ms)
- ✓ should check the allowance later
- ✓ Should be able to burn Tokens approved by account[3] to account[5] (162ms)
- ✓ Should be able to emit transfer event while burnFrom tokens (217ms)
- ✓ Should check the Total Supply of BTCCToken Tokens after tokens burn
- ✓ Should correctly initialize constructor values of sample Token Contract (110ms)
- ✓ Should check the balance of a sample contract Owner
- ✓ Should be able to transfer sample tokens to BTCCToken contract (219ms)
- ✓ Should be able to transfer sample tokens to owner of BTCCToken, sent to contract of BTCCToken (225ms)

✓ Should Not be able to send ethers to contract

Contract: BTCCToken Token Contract for final audit testing at maximum values

✓ Should correctly initialize constructor values of BTCCToken Token Contract (99ms)

✓ Should check the Total Supply of BTCCToken Tokens (39ms)

✓ Should check the Name of a token of BTCC Token contract

✓ Should check the symbol of a token of BTCCToken contract

✓ Should check the decimal of a token of BTCCToken contract

✓ Should check the balance of a Owner

✓ Should check the owner of a contract

✓ Should check the New owner of a contract

✓ Should check the balance of a contract (57ms)

✓ Should check the distributed tokens of a contract

✓ Should check the owner account is not frozen

✓ Should check the minted tokens (38ms)



✓ Should be able to Mint Tokens by owner only more then total supply (128ms)

✓ Should check the minted tokens after minted

✓ Should check the Total Supply of BTCCToken Tokens after tokens minted

✓ Should be able to distribute tokens to accounts[1] (192ms)

✓ Should be able to self destruct (192ms)

## Final Result of Test:

✓ 64 Passing (8s) PASSED

**✗ 0 Failed**

## Manual Transactions

**Network** : Rinkeby

Contract Creation	0xe1293ac890cefe3ec24376ba736e3627c50357361b2b76e383827d06732b2fc6
Distribute tokens	0x9718970006fb25c2cd8060fcb995b150e407b9625c236e4b8f59b6b14c4ff366
Mint tokens	0xb31697d901e7933ba5636f1361a52a6df2242b34446ee36da61a3deb0b84d46c
Fall back function (Should failed)	0x6e135a696036d6cbb0a8e5e6c6cafc066b4f59e62d5a9827ccb45ef44f2cd9d9
Distribute tokens more than token supply should be failed (1000 distributed first, 1000 minted, distributed 300mn + 1000 tokens)	0x7e51c9593df05d1d209cd3c4200d61b34924725f1fcd5a737da7d066dd3ea135
Freeze Account	0xee54d3bad0047483ead846b16548e2a9f819eb0dee80d380769a3bdaa889e63f
Burn tokens(Failed when try to burn tokens that account have, because total supply is less then account holders token)	0xe73f81fe90892e1450c9294359e0de387bdb49df986aa432bd2654a22c0be396
Burn tokens (No transfer event when tokens burn)	0x03917d8c4562db959fb985713b6a0576d8eff63193843acbac686b62f368d05c

Mint tokens	0xec6b306d78c8d68e11d5c95a10dffa 044d8ccaa675ddb3f05b543add63e71f aa
Transfer function	0xaf3a913d6e59038897921c13b9d465 cecb3d028ffb6d77b1ffb96ad87405ca9 d
Transfer Ownership	0xf8cc6713f3397f8e66b236ac57d7608 d3ab4a1ddba0622915153c74f8beac2e f
Accept ownership by non New owner account (Failed)	0x3f66afc8c0e2493a165e0560a23820 5cfc0a8fce3ff4525d577ee2632a57b5c 6
Accept ownership	0xd65d67d585fe3485f969783529c8f0f 6bef0ebfc2e42eef2c01c9df89328d457
Approve	0x497b1f7cc1a6e037dc222d58575df4 2f8974ad76121b0818ee2d1d4c0d7f07 18
transferFrom fail by frozen account	0x7ae353f779ad671891f8095241d948 8e521870ea7b4cf9eae228fa056f398da c
BurnFrom	0xe8f058f6d95aa96049612b88d4354f 5e88188a10cf90546341422a328981a3 79
Fallback function	0x8758fbcf020eba66173fd160a90d483 1ff8b4f6e4ce4eba3155be092adb848fd
Unfreezed account	0xfa1ff4f82112aab86419a91408ab51a 2972772ba2b2fc7e392b57bce1cdc186 8
Minted 300 min tokens to contract	0xfcf0cf0b7173067d607f3f39c22dab64

address	fe5d5301aa6c4bd4356d005314795726
Self destruct by non owner account	0xc63b634bb2a3041dd9ab9c6404b7b a2d4e4e928cffffe262d8bfe9c165c903f8 a
Self destruct by owner account	0x0aa1a1a76372ec4549ab96c8d8d393 be04308c99dbb0252d3b9c03fc9adb44 79

## Tool Result :

```
INFO:Detectors:
ERC20Interface.totalSupply (btcCredit/contracts/BTCCToken.sol#38) should be declared external
BTCCToken.totalSupply (btcCredit/contracts/BTCCToken.sol#141-143) should be declared external
ERC20Interface.balanceOf (btcCredit/contracts/BTCCToken.sol#39) should be declared external
BTCCToken.balanceOf (btcCredit/contracts/BTCCToken.sol#149-151) should be declared external
BTCCToken.allowance (btcCredit/contracts/BTCCToken.sol#248-250) should be declared external
ERC20Interface.allowance (btcCredit/contracts/BTCCToken.sol#40) should be declared external
BTCCToken.transfer (btcCredit/contracts/BTCCToken.sol#159-165) should be declared external
ERC20Interface.transfer (btcCredit/contracts/BTCCToken.sol#41) should be declared external
BTCCToken.approve (btcCredit/contracts/BTCCToken.sol#176-180) should be declared external
ERC20Interface.approve (btcCredit/contracts/BTCCToken.sol#42) should be declared external
ERC20Interface.transferFrom (btcCredit/contracts/BTCCToken.sol#43) should be declared external
BTCCToken.transferFrom (btcCredit/contracts/BTCCToken.sol#192-199) should be declared external
ApproveAndCallFallback.receiveApproval (btcCredit/contracts/BTCCToken.sol#60) should be declared external
Owned.transferOwnership (btcCredit/contracts/BTCCToken.sol#82-84) should be declared external
Owned.acceptOwnership (btcCredit/contracts/BTCCToken.sol#85-90) should be declared external
BTCCToken.distributeTokens (btcCredit/contracts/BTCCToken.sol#119-129) should be declared external
BTCCToken.distributedTokenCount (btcCredit/contracts/BTCCToken.sol#134-136) should be declared external
BTCCToken.mintToken (btcCredit/contracts/BTCCToken.sol#208-215) should be declared external
BTCCToken.freezeAccount (btcCredit/contracts/BTCCToken.sol#217-220) should be declared external
BTCCToken.burn (btcCredit/contracts/BTCCToken.sol#224-230) should be declared external
BTCCToken.burnFrom (btcCredit/contracts/BTCCToken.sol#232-242) should be declared external
BTCCToken.fallback (btcCredit/contracts/BTCCToken.sol#256-258) should be declared external
BTCCToken.transferAnyERC20Token (btcCredit/contracts/BTCCToken.sol#264-266) should be declared external
BTCCToken.destroyContract (btcCredit/contracts/BTCCToken.sol#268-271) should be declared external
Migrations.setCompleted (btcCredit/contracts/Migrations.sol#15-17) should be declared external
Migrations.upgrade (btcCredit/contracts/Migrations.sol#19-22) should be declared external
```

Above functions can be declared as an external functions

```
Parameter '_newOwner' of Owned.transferOwnership (btcCredit/contracts/BTCCToken.sol#82) is not in mixedCase
Parameter '_address' of BTCCToken.distributeTokens (btcCredit/contracts/BTCCToken.sol#119) is not in mixedCase
Parameter '_amount' of BTCCToken.distributeTokens (btcCredit/contracts/BTCCToken.sol#119) is not in mixedCase
Parameter '_target' of BTCCToken.mintToken (btcCredit/contracts/BTCCToken.sol#208) is not in mixedCase
Parameter '_mintedAmount' of BTCCToken.mintToken (btcCredit/contracts/BTCCToken.sol#208) is not in mixedCase
Parameter '_target' of BTCCToken.freezeAccount (btcCredit/contracts/BTCCToken.sol#217) is not in mixedCase
Parameter '_freeze' of BTCCToken.freezeAccount (btcCredit/contracts/BTCCToken.sol#217) is not in mixedCase
Parameter '_burnedAmount' of BTCCToken.burn (btcCredit/contracts/BTCCToken.sol#224) is not in mixedCase
Parameter '_from' of BTCCToken.burnFrom (btcCredit/contracts/BTCCToken.sol#232) is not in mixedCase
Parameter '_burnedAmount' of BTCCToken.burnFrom (btcCredit/contracts/BTCCToken.sol#232) is not in mixedCase
Variable 'BTCCToken._totalSupply' (btcCredit/contracts/BTCCToken.sol#102) is not in mixedCase
Parameter 'new_address' of Migrations.upgrade (btcCredit/contracts/Migrations.sol#19) is not in mixedCase
Variable 'Migrations.last_completed_migration' (btcCredit/contracts/Migrations.sol#5) is not in mixedCase
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#conformance-to-solidity-naming-conventions
```

Some of the variables are not mixed case variables

### Smart contract Description

File Name	SHA-1 Hash			
BTCCToken.sol	54e714c34ca96b6675e5d7ebeca89e112e68a22f			
### Contracts Description Table				
Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
**BTCCToken**	Implementation	ERC20Interface, Owned, SafeMath		
L	<Constructor>	Public	!	●
L	distributeTokens	Public	!	●
L	distributedTokenCount	Public	!	onlyOwner
L	totalSupply	Public	!	NO
L	balanceOf	Public	!	NO
L	transfer	Public	!	NO
L	approve	Public	!	NO
L	transferFrom	Public	!	NO
L	mintToken	Public	!	onlyOwner
L	freezeAccount	Public	!	onlyOwner
L	burn	Public	!	NO
L	burnFrom	Public	!	NO
L	allowance	Public	!	NO
L	<Fallback>	Public	!	NO
L	transferAnyERC20Token	Public	!	onlyOwner
L	destroyContract	Public	!	onlyOwner
### Legend				
Symbol	Meaning			
●	Function can modify state			
!!	Function is payable			

## Coverage report

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	57.72	36.54	54.55	57.6	
BTCCToken.sol	96.77	68.18	90.91	96.83	257,265
TokenA.sol	18.03	13.33	18.18	17.74	... 287,288,289
All files	57.72	36.54	54.55	57.6	

Token A is a sample token contract to check transferAnyERC20Token() function of BTCCtoken contract function.

## Implementation Recommendations :

- `_totalSupply` variable can be declared as private variable, as `totalSupply()` function already using `_totalSupply` variable.
- Use `SafeMath` while subtracting `totalSupply` with `balance[address(0)]`.
- `transferFrom()` function should check allowance first.
- Use transfer event while initialising tokens in constructor, transfer event from `address(0)` to contract address and also update balance of contract.
- While distributing tokens use transfer event from contract address to beneficiary to maintain all the balance events on etherscan.

### Comments:

Use case of smart contract is very well designed and Implemented. Overall, the code is clearly written, and demonstrates effective use of abstraction, separation of concerns, and modularity. **BTC CREDIT** development team demonstrated high technical capabilities, both in the design of the architecture and in the implementation. All the severity issues has been fixed by BTC CREDIT team and we request BTC credit team to go through recommendations as well to maintain standardisation of code.

Note : Smart contract code contain **self destruct** function which can be called by owner only.