bitcci

Smart Contract Audit Report

June 16, 2021

IMMUNE BYTES

Introduction

1. About bitcci

bitcci has united the sex industry with the new blockchain world. The bitcci Group AG is made up of a group of public stock companies located in the cryptovalley of Liechtenstein and Switzerland. Founded by Christoph Elbert in 2017, the bitcci Group AG is building an entirely new global ecosystem for the next-generation sex industry, eliminating problems and making the industry legal, safe and free. The bitcci AG, a 100% subsidiary of the bitcci Group AG, located in Zug, Switzerland, has developed the bitcci cash token and will execute the token emission in 2021.

Visit https://bitcci.ag./ to know more about.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 15+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing 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 Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

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. Analyzing 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: bitcci
- Languages: Solidity(Smart contract), Javascript(Unit Testing)
- Github commit hash for audit: b57bac012b108d6e15568b624c2efdc6d6968969
- Testnet Deployment:
    - bitcciCash: 0xd0ABC143212E090F04C50B6e13a2Ef9E77a96876
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

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 into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level References

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

Admin/Owner Privileges can be misused either intentionally or unintentionally.

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.

Issues	High	Medium	Low
Open	-	-	1
Closed	-	-	-

Admin/Owner Privileges

The **admin/owner** of **bitcciCash** smart contract has various privileges over the smart contract. These privileges can be misused either intentionally or unintentionally (in case admin's private key gets hacked). We assume that these extra rights will always be used appropriately. Some of these admin rights are listed below.

1. **In the bitcciCash contract the MINTER_ROLE address can mint tokens upto the total supply cap.**
   The **bitcciCash** contract contains a **mint**() function by which the account with a **MINTER_ROLE** can mint **bitcciCash** tokens to any ethereum address. However, since a max total supply limit is implemented in the smart contract no more tokens can be minted beyond that limit.

2. **In the bitcciCash contract the PAUSER_ROLE address can pause the transfer of all tokens.**
   The **bitcciCash** contract contains a **pause**() function by which the account with a **PAUSER_ROLE** can pause the transfer of tokens. The **PAUSER_ROLE** is also responsible for unpausing the token transfers using the **unpause**() function.

3. **MINTER_ROLE and PAUSER_ROLE can be given to any address/account by the deployer (DEFAULT_ADMIN_ROLE) of bitcciCash contract.**
   The deployer of **bitcciCash** contract possesses the **DEFAULT_ADMIN_ROLE** for the contract. This role has the right to give a **MINTER_ROLE** and **PAUSER_ROLE** to any ethereum address. Please note that there could be more than one MINTER and PAUSER possible for the **bitcciCash** contract.

4. **The DEFAULT_ADMIN_ROLE or deployer of bitcciCash contract can add or remove any ethereum address to/from blacklist.**
   The **DEFAULT_ADMIN_ROLE** has the right to add or remove any ethereum address to a blacklist. The blacklisted addresses cannot send their **bitcciCash** tokens to any other address.

Recommendation:
Consider adding some governance for admin rights for the smart contract or use a multi-sig wallet as admin/owner address.

Low severity issues

1. **bitcciCash tokens can be sent to a blacklisted address.**
   The **bitcciCash** contract implements a **BLACKLISTED_ROLE** to stop the blacklisted addresses to perform token transfers. However the **_beforeTokenTransfer**() function of **bitcciCash** token contract only performs the blacklisting check for the **from** address (i.e. the sender) and not for the **to** address (recipient).

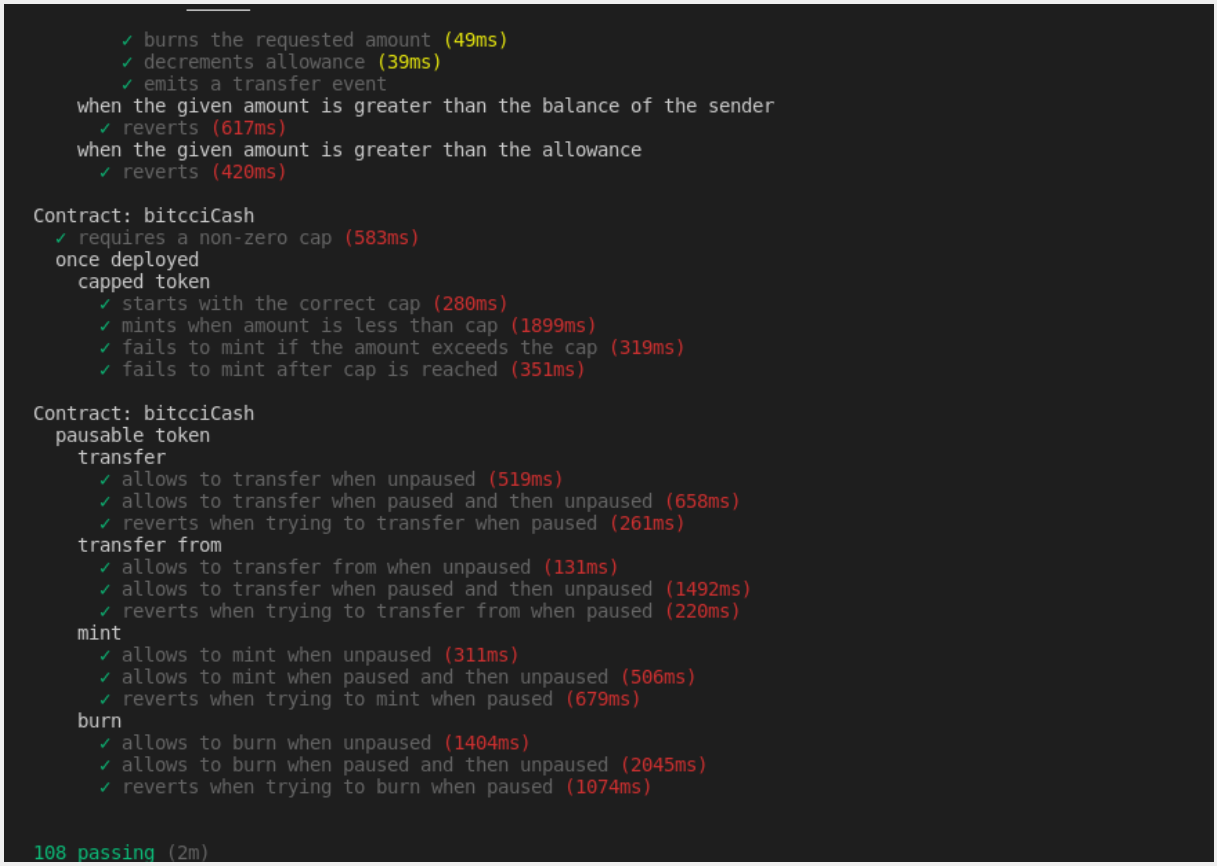
```
function _beforeTokenTransfer(address from, address to, uint256 amount)
      internal virtual override(ERC20, ERC20Pausable)
{
    require(!(hasRole(BLACKLISTED_ROLE, msg.sender)), "bitcciCash:
      account is Blacklisted");
    super._beforeTokenTransfer(from, to, amount);
}
```

   Due to this the **bitcciCash** tokens can be sent to a blacklisted address.

   Recommendation:
   Consider performing the blacklisting check for token recipients as well.
   Also consider using the **from** parameter instead of **msg.sender**.

Unit Test

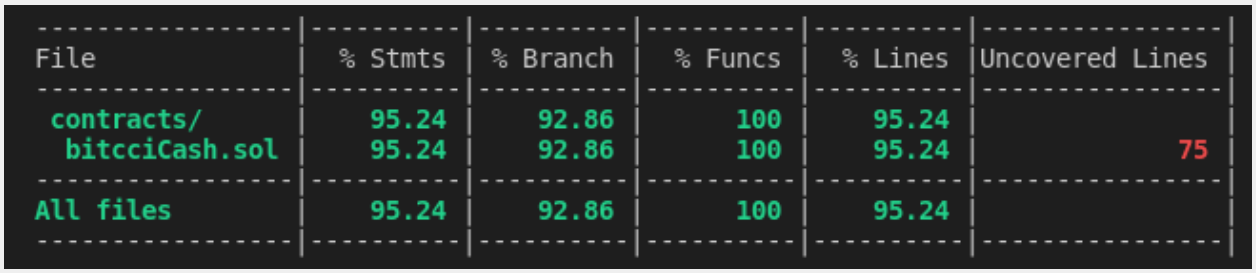
All unit tests provided by bitcciCash are passing without issues.



```
        ✓ burns the requested amount (49ms)
        ✓ decrements allowance (39ms)
        ✓ emits a transfer event
    when the given amount is greater than the balance of the sender
      ✓ reverts (617ms)
    when the given amount is greater than the allowance
      ✓ reverts (420ms)

Contract: bitcciCash
  ✓ requires a non-zero cap (583ms)
  once deployed
    capped token
      ✓ starts with the correct cap (280ms)
      ✓ mints when amount is less than cap (1899ms)
      ✓ fails to mint if the amount exceeds the cap (319ms)
      ✓ fails to mint after cap is reached (351ms)

Contract: bitcciCash
  pausable token
    transfer
      ✓ allows to transfer when unpaused (519ms)
      ✓ allows to transfer when paused and then unpaused (658ms)
      ✓ reverts when trying to transfer when paused (261ms)
    transfer from
      ✓ allows to transfer from when unpaused (131ms)
      ✓ allows to transfer when paused and then unpaused (1492ms)
      ✓ reverts when trying to transfer from when paused (220ms)
    mint
      ✓ allows to mint when unpaused (311ms)
      ✓ allows to mint when paused and then unpaused (506ms)
      ✓ reverts when trying to mint when paused (679ms)
    burn
      ✓ allows to burn when unpaused (1404ms)
      ✓ allows to burn when paused and then unpaused (2045ms)
      ✓ reverts when trying to burn when paused (1074ms)


108 passing (2m)
```

Coverage Report

Test coverage of bitcciCash's smart contract is not 100%.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	95.24	92.86	100	95.24	
bitcciCash.sol	95.24	92.86	100	95.24	75
All files	95.24	92.86	100	95.24	

Recommendation:
We recommend 100% line and branch coverage for unit test cases.

Automated Auditing

Solhint Linting Violations

Solhint is an open-source project for linting solidity code, providing both security and style guide validations. It integrates seamlessly into most mainstream IDEs. We used Solhint as a plugin within our VScode for this analysis. Multiple Linting violations were detected by Solhint, it is recommended to use Solhint's npm package to lint the contracts.

Contract Library

Contract-library contains the most complete, high-level decompiled representation of all Ethereum smart contracts, with security analysis applied to them in real-time. We performed analysis using contract Library on the Rinkeby address of the bitcciCash contract used during manual testing:

- bitcciCash: 0xd0ABC143212E090F04C50B6e13a2Ef9E77a96876

It raises no major concern for the contracts.

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.

The concerns slither raises have already been covered in the manual audit section.

```
AccessControlEnumerable.grantRole(bytes32,address) (Flattened.sol#1309-1312) ignores return value by _roleMembers[rol
e].add(account) (Flattened.sol#1311)
AccessControlEnumerable.revokeRole(bytes32,address) (Flattened.sol#1317-1320) ignores return value by _roleMembers[ro
le].remove(account) (Flattened.sol#1319)
AccessControlEnumerable.renounceRole(bytes32,address) (Flattened.sol#1325-1328) ignores return value by _roleMembers[
role].remove(account) (Flattened.sol#1327)
AccessControlEnumerable._setupRole(bytes32,address) (Flattened.sol#1333-1336) ignores return value by _roleMembers[ro
le].add(account) (Flattened.sol#1335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
bitcciCash.constructor(string,string,uint8,uint256).name (Flattened.sol#1377) shadows:
        - ERC20.name() (Flattened.sol#189-191) (function)
        - IERC20Metadata.name() (Flattened.sol#95) (function)
bitcciCash.constructor(string,string,uint8,uint256).symbol (Flattened.sol#1377) shadows:
        - ERC20.symbol() (Flattened.sol#197-199) (function)
        - IERC20Metadata.symbol() (Flattened.sol#100) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (Flattened.sol#935-938) is never used and should be removed
Context._msgData() (Flattened.sol#125-128) is never used and should be removed
ERC20Pausable._beforeTokenTransfer(address,address,uint256) (Flattened.sol#584-588) is never used and should be remov
ed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (Flattened.sol#1101-1103) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (Flattened.sol#1210-1212) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (Flattened.sol#1139-1141) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (Flattened.sol#1248-1250) is never used and should be removed
EnumerableSet.contains(EnumerableSet.AddressSet,address) (Flattened.sol#1172-1174) is never used and should be remove
d
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (Flattened.sol#1118-1120) is never used and should be remove
d
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (Flattened.sol#1227-1229) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (Flattened.sol#1125-1127) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (Flattened.sol#1234-1236) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (Flattened.sol#1111-1113) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (Flattened.sol#1220-1222) is never used and should be removed
Strings.toHexString(uint256) (Flattened.sol#629-640) is never used and should be removed
Strings.toString(uint256) (Flattened.sol#604-624) is never used and should be removed
bitcciCash._beforeTokenTransfer(address,address,uint256) (Flattened.sol#1458-1461) is never used and should be remove
```

Concluding Remarks

While conducting the audits of the bitcci smart contract, it was observed that the contracts contain Low severity issues, along with a few admin area of admin/owner privileges.

Our auditors suggest that Low severity issues should be resolved by the bitcci developers. Resolving the areas of recommendations are up to the team's discretion. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the bitcci platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes Pvt Ltd.