



# Security Assessment BITCOIN SPARK

Vital Block **Verified** on August 9<sup>th</sup>, 2023

 @Vital-Block

 @VB\_Audit

 info@vitalblock.org




 www.vitalblock.org



PREPARED FOR:  
BITCOIN SPARK



## INTRODUCTION

|                    |   |
|--------------------|---|
| Auditing Firm      |  VITAL BLOCK SECURITY  |
| Client Project     |  BITCOIN SPARK         |
| Methodology        | Automated Analysis, Manual Code Review  |
| Language           | Solidity  |
| Contract           | 0xfbc3dc3ac9f76d9702c79ac3e4ccfda9999ca7eff   |
| Source Code        | Ethereum Mainnet  |
| License            | MIT   |
| Centralization     | Active ownership  |
| Compiler Version   | v0.8.18+commit.87f61d96   |
| Blockchain         |  Ethereum            |
| Telegram           | <a href="https://t.me/BitcoinSpark">https://t.me/BitcoinSpark</a>                                       |
| Medium             | <a href="https://medium.com/@bitcoinspark">https://medium.com/@bitcoinspark</a>                         |
| Twitter            | <a href="https://twitter.com/BitcoinSparkOrg">https://twitter.com/BitcoinSparkOrg</a>                   |
| Doc                | <a href="https://bitcoin-spark-foundation.gitbook.io/">https://bitcoin-spark-foundation.gitbook.io/</a> |
| Prelim Report Date | August 8 <sup>th</sup> 2023   |
| Final Report Date  | August 9 <sup>h</sup> 2023  |



Verify the authenticity of this report on our GitHub Repo: <https://www.github.com/vital-block>

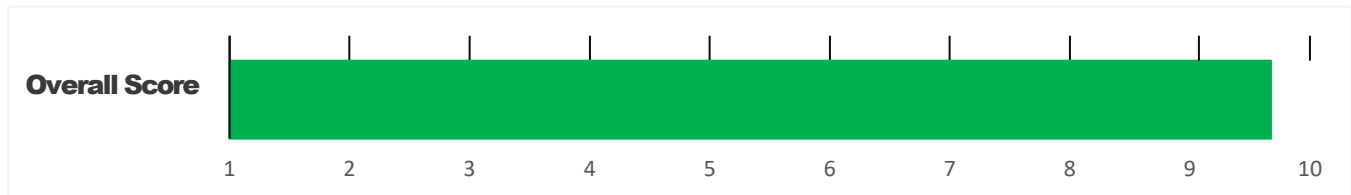



## EXECUTIVE SUMMARY

Vital Block Security has performed the automated and manual analysis of the BITCOIN SPARK Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

| Status  | Critical ! 🔴  | Major " 🟡 | Medium # 🟡 | Minor \$ 🟢 | Unknown % 🟤 |
|---|---|-----------|------------|------------|-------------|
| Open  | 0   | 0         | 1          | 3          | 0           |
| Acknowledged  | 0   | 0         | 3          | 2          | 0           |
| Resolved  | 0   | 0         | 0          | 0          | 0           |
|   |   |           |            |            |             |
| Noteworthy<br><a href="#">onlyOwner</a><br>Privileges | Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router |           |            |            |             |

BITCOIN SPARK Smart contract has achieved the following score: **92.0**



 Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

 Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.



# TABLE OF CONTENTS

|                              |    |
|------------------------------|----|
| TABLE OF CONTENTS .....      | 4  |
| SCOPE OF WORK .....          | 5  |
| AUDIT METHODOLOGY .....      | 6  |
| RISK CATEGORIES .....        | 8  |
| CENTRALIZED PRIVILEGES ..... | 9  |
| AUTOMATED ANALYSIS .....     | 10 |
| INHERITANCE GRAPH .....      | 15 |
| MANUAL REVIEW .....          | 16 |
| DISCLAIMERS .....            | 27 |
| ABOUT VITALBLOCK .....       | 30 |



## SCOPE OF WORK

Vital Block was consulted by BITCOIN SPARK to conduct the smart contract audit of its. Move source code. The audit scope of work is strictly limited to mentioned .Sol file only:

O.BITCOINSPARK.sol

 External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

| Public Contract Address                           |               |
|---|---------------|
| <b>0xfbcdc3ac9f76d9702c79ac3e4ccfda9999ca7eff</b> |               |
|   |               |
| Contract Name                                     | BITCOIN SPARK |
| Token Symbol                                      | BTCS          |
| Decimals  | 18            |
| Total Supply                                      | 4,300,000     |



## AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block

Security auditing process and methodology:

### CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

### AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges.

We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

|                      |   |
|----------------------|---|
| Centralized Exploits | <ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul> |
|----------------------|---|



### **Common Contract Vulnerabilities**

- **Integer Overflow**
- **Lack of Arbitrary limits**
- **Incorrect Inheritance Order**
- **Typographical Errors**
- **Requirement Violation**
- **Gas Optimization**
- **Coding Style Violations**
- **Re-entrancy**
- **Third-Party Dependencies**
- **Potential Sandwich Attacks**
- **Irrelevant Codes**
- **Divide before multiply**
- **Conformance to Solidity Naming Guides**
- **Compiler Specific Warnings**
- **Language Specific Warnings**

### **REPORT**

- **The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.**
- **The client's development team reviews the report and makes amendments to the codes.**
- **The auditing team provides the final comprehensive report with open and unresolved issues.**

### **PUBLISH**

- **The client may use the audit report internally or disclose it publicly.**

 **It is important to note that there is no pass or fail in the audit, it is recommended to view the audit**

**as an unbiased assessment of the safety of solidity codes.**



## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

| Risk Type  | Definition  |
|------------|---|
| Critical 🛑 | These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.  |
| Major 🟡    | These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.  |
| Medium 🟡   | These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits. |
| Minor 🟢    | These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.                 |
| Unknown 🟤  | These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.   |

All statuses which are identified in the audit report are categorized here for the reader to review:

| Status Type  | Definition                             |
|--------------|--|
| Open         | Risks are open.                        |
| Acknowledged | Risks are acknowledged, but not fixed. |
| Resolved     | Risks are acknowledged and fixed.      |





## CENTRALIZED PRIVILEGES

**Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.**

**There are some well-intended reasons have privileged roles, such as:**

- **Privileged roles can be granted the power to `pause()` the contract in case of an external attack.**
- **Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**






**Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.**

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

 **Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.**



## AUTOMATED ANALYSIS

| Symbol  | Definition              |
|---|-------------------------|
|  | Function modifies state |
|  | Function is payable     |
|  | Function is internal    |
|  | Function is private     |
|  | Function is important   |

```

**BITCOIN SPARK** | Interface | |||
| L | totalSupply | External | ! | NO |
| L | decimals | External | ! | NO |
| L | symbol | External | ! | NO |
| L | name | External | ! | NO |
| L | getOwner | External | NO |
| L | balanceOf | External | ! | NO |
| L | transfer | External | " ! ! | NO |
| L | allowance | External | ! | NO |
| L | approve | External | " ! ! | NO |
| L | transferFrom | External | " | NO |
|||||
**IFactoryV2** | Interface | |||
| L | getPair | External | NO |
| L | createPair | External | " | NO |
|||||
**IV2Pair** | Interface | |||
| L | factory | External | NO |
| L | getReserves | External | NO |
| L | sync | External | " | NO |

```

|||||

| **\*\*IRouter01\*\*** | Interface | |||

| L | factory | External ¶ | |NO¶|

| L | Sol | External ¶ | |NO¶|

| L | addLiquidityETH | External ¶ | # |NO¶|

| L | addLiquidity | External ¶ | " |NO¶|

| L | swapExactETorTokens | External ¶ | # |NO¶|

| L | getAmountsOut | External ¶ | |NO¶|

| L | getAmountsIn | External ¶ | |NO¶|

|||||

| **\*\*IRouter02\*\*** | Interface | IRouter01 |||

| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External ¶ | " |NO¶|

| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ¶ | # |NO¶|

| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ¶ | " ! 🔴 |NO¶|

| L | swapExactTokensForTokens | External ¶ | " |NO¶|

|||||

| **\*\*Protections\*\*** | Interface | |||

| L | checkUser | External ¶ | " ! 🔴 |NO¶|

| L | setLaunch | External ¶ | " |NO¶|

| L | setLpPair | External ¶ | " |NO¶|

| L | **BTCS** | External ¶ | " |NO¶|

| L | removeSniper | External ¶ | " |NO¶|

|||||

| **\*\*Cashier\*\*** | Interface | |||

| L | setRewardsProperties | External ¶ | " |NO¶|

| L | tally | External ¶ | " |NO¶|

| L | load | External ¶ | # |NO¶|

| L | cashout | External ¶ | " |NO¶|

| L | giveMeWelfarePlease | External ¶ | " |NO¶|

| L | getTotalDistributed | External ¶ | |NO¶|

| L | getUserInfo | External ¶ | |NO¶|

| L | getUserRealizedRewards | External ¶ | |NO¶|

```

| L | getPendingRewards | External | | | NO |
| L | initialize | External | | " | NO |
| L | getCurrentReward | External | | | NO |
|||||
| **SOL** | Implementation | SafeMath | |||
| L | <Constructor> | Public | | # | NO |
| L | transferOwner | External | | " | onlyOwner |
| L | renounceOwnership | External | | " | NO |
| L | setOperator | Public | | | NO |
| L | renounceOriginalDeployer | External | | " | NO |
| L | <Receive ETH> | External | | # | NO |
| L | totalSupply | External | | | NO |
| L | decimals | External | | | NO |
| L | symbol | External | | | NO |
| L | name | External | | | NO |
| L | getOwner | External | | ! | NO |
| L | balanceOf | Public | | ! | NO |
| L | allowance | External | | ! | NO |
| L | approve | External | | " ! | NO |
| L | _approve | Internal | $ | " | NO |
| L | approveContractContingency | Public | | " ! | onlyOwner |
| L | transfer | External | | " | NO |
| L | transferFrom | External | | " | NO |
| L | setNewRouter | External | | " | onlyOwner |
| L | setLpPair | External | | " | onlyOwner |
| L | setInitializers | External | | " | onlyOwner |
| L | isExcludedFromFees | External | | | NO |
| L | isExcludedFromDividends | External | | | NO |
| L | isExcludedFromProtection | External | | | NO |
| L | setDividendExcluded | Public | | " | onlyOwner |
| L | setExcludedFromFees | Public | | " | onlyOwner |

```

## BTV-01 POSSIBLE OVERFLOW

| Category   | Severity <span>●</span> | Location                  | Status       |
|------------|-------------------------|---------------------------|--------------|
| Suboptimal | Minor                   | Contract/BitcoinSpark.sol | Acknowledged |

### Description

In `updateForToken`, the following equation is used inside an unchecked block

```
_beforeTokenTransfer(from, to, amount);  
  
uint256 fromBalance = _balances[from];  
require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");  
unchecked {  
    _balances[from] = fromBalance - amount;  
}
```

Where parameters. Block `Token` less Used is a this and override In is a this.  
As these two are multiplied together in an unchecked block, they may overflow.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never cause an overflow

## BST-02 POSSIBLE OVERFLOW

| Category                       | Severity ● | Location                  | Status       |
|--------------------------------|------------|---------------------------|--------------|
| Status Mathematical Operations | Minor      | Contract/BitcoinSpark.sol | Acknowledged |

### Description

In `updateForMinter`, the following equation is used inside an unchecked block

```
contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;
```

Minter can not issue more `BTCS` tokens indefinitely.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the `BTCS` contract.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the `PairsIn` is close enough it will never cause an overflow.

## FZT-03 POSSIBLE OVERFLOW

| Category      | Severity ●    | Location                  | Status       |
|---------------|---------------|---------------------------|--------------|
| Inconsistency | Informational | Contract/BitcoinSpark.Sol | Acknowledged |

### Description

In `updateForaddress`, the following equation is used inside an unchecked block






```
function transfer(address to, uint256 amount) public virtual override returns (bool) {  
    address owner = _msgSender();  
    _transfer(owner, to, amount);  
    return true;  
}
```

The function `address ()` does not have the override specifier. It should be noted that since `price0 > a` function that overrides only a single interface function does not require the override specifier (see doc). However, all other instances of this in the code base contain the override specifier.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the `PairsIn` is close enough it will never cause an overflow.

## OPTIMIZATIONS | BITCOIN SPARK

| ID  | Title                             | Category         | Status   |
|-----|-----------------------------------|------------------|--|
| FTV | Logarithm Refinement Optimization | Gas Optimization | Acknowledged  |
| FOP | Checks Can Be Performed Earlier   | Gas Optimization | Acknowledged  |
| FDP | Unnecessary Use Of SafeMath       | Gas Optimization | Acknowledged  |
| FWY | Struct Optimization               | Gas Optimization | Acknowledged  |
| FGT | Unused State Variable             | Gas Optimization | Acknowledged  |



## General Detectors

### Missing Zero Address Validation

Some functions in this contract may not appropriately check for zero addresses being used.









































Attention  
Required

### Incorrect Solidity Version

This contract uses an unconventional or very old version of Solidity



Attention  
Required

- |  |  |
|--|--|
|  No compiler version inconsistencies found      |  No tautologies or contradictions found                       |
|  No unchecked call responses found              |  No faulty true/false values found                            |
|  No vulnerable self-destruct functions found    |  No innacurate divisions found                                |
|  No assertion vulnerabilities found            |  No redundant constructor calls found                        |
|  No old solidity code found                   |  No vulnerable transfers found                              |
|  No external delegated calls found            |  No vulnerable return values found                          |
|  No external call dependency found            |  No uninitialized local variables found                     |
|  No vulnerable authentication calls found     |  No default function responses found                        |
|  No invalid character typos found             |  No missing arithmetic events found                         |
|  No RTL characters found                      |  No missing access control events found                     |
|  No dead code found                           |  No redundant true/false comparisons found                  |
|  No risky data allocation found               |  No state variables vulnerable through function calls found |
|  No uninitialized state variables found       |  No buggy low-level calls found                             |
|  No uninitialized storage variables found     |  No expensive loops found                                   |
|  No vulnerable initialization functions found |  No bad numeric notation practices found                    |
|  No risky data handling found                 |  No missing constant declarations found                     |
|  No number accuracy bug found                 |  No missing external function declarations found            |
|  No out-of-range number vulnerability found   |  No vulnerable payable functions found                      |
|  No map data deletion vulnerabilities found   |  No vulnerable message values found                         |



## Vulnerability Scan

### REENTRANCY

✓ No reentrancy risk found

Severity Minor

Confidence Parameter Certain

✗ **Not Mintable:** A large amount of this token can not be minted by a private wallet or contract.

## Vulnerability Description

## Scanning Line:

```
function _approve(address owner, address spender, uint256
amount) internal virtual {
    require(owner != address(0), "ERC20: approve from the
zero address");
    require(spender != address(0), "ERC20: approve to the
zero address");
```

```
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

```
function _spendAllowance(address owner, address spender,
uint256 amount) internal virtual {
    uint256 currentAllowance = allowance(owner, spender);
    if (currentAllowance != type(uint256).max) {
        require(currentAllowance >= amount, "ERC20:
insufficient allowance");
        unchecked {
            _approve(owner, spender, currentAllowance -
amount);
        }
    }
}
```

| Identifier | Definition                 | Severity  |
|------------|----------------------------|---|
| CEN-02     | Initial asset distribution | Minor  |

```
function _mint(address account, uint256 amount) internal virtual
{
    require(account != address(0), "ERC20: mint to the zero
address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply += amount;
    unchecked {
```

## Description:

Floating point calculations can vary across different architectures.

## Alleviation:

This exhibit was acknowledged and ultimately discarded by the **BITCOIN SPARK** team due to low severity. We consider the exhibit fully attended to as it doesn't impose any meaningful security concerns.

## RECOMMENDATION

**Project stakeholders should be consulted during the initial asset distribution process.**



## Contract Owner Address:

<https://etherscan.io/address/0xD40a3b6Bb78C78e0F61f57be1140B12Fd790a8e5>

## Audited Files

BITCOINSPARK.SOL

## Contracts:

Contract

BITCOIN SPARK::

[https://etherscan.io/address/0xfbcdc3ac9f76d9702c79ac3e4ccfda9999ca7eff#code:](https://etherscan.io/address/0xfbcdc3ac9f76d9702c79ac3e4ccfda9999ca7eff#code)



## Vulnerability Run check

### Risk Analysis

#### ✔ Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

#### ✔ No Proxy

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

#### ✔ No mint function

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

#### ✔ No function to retrieve ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

#### ✔ Owner cant change balance

The contract owner does not have the authority to modify the balance of tokens at other addresses.



### Honeypot Risk

#### ✔ This does not appear to be a honeypot

We are not aware of any code that prevents the sale of tokens.

#### ✔ No trading cooldown

The token contract has no trading cooldown function. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

#### ✔ No Anti Whale

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

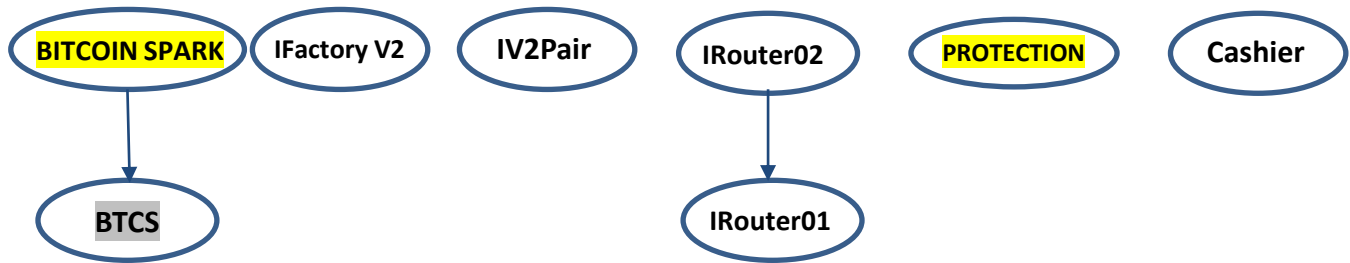
#### ✔ No blacklist function

No blacklist function is included.

#### ✔ No whitelist function

Whitelist function found

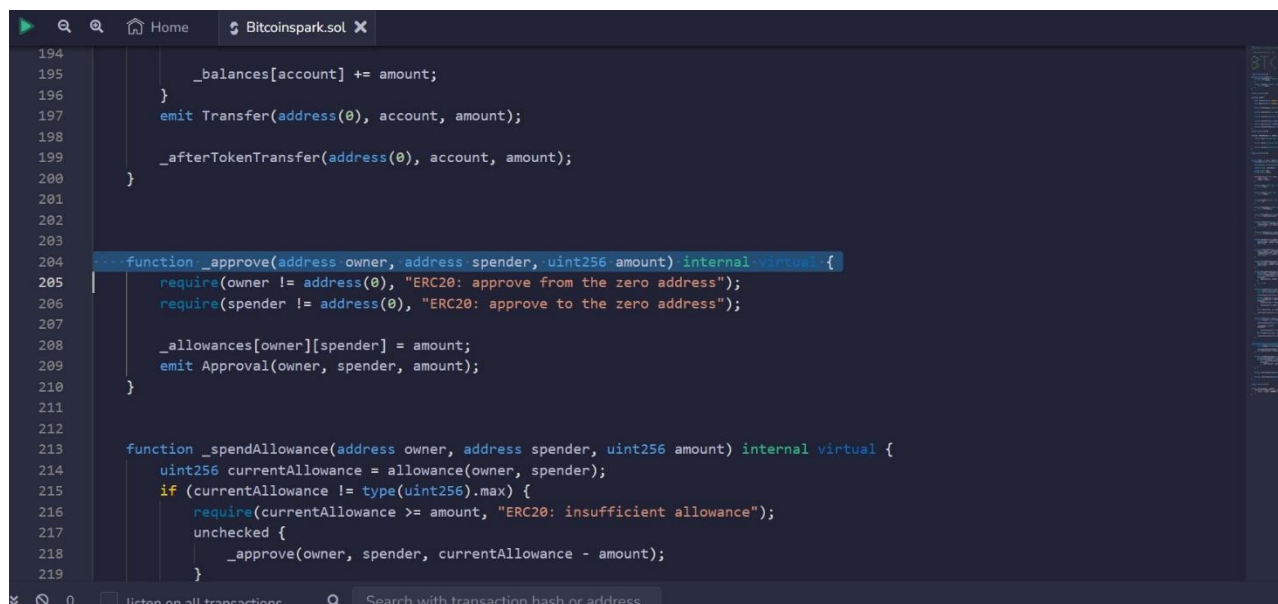
## INHERITANCE GRAPH



| Identifier | Definition                                 | Severity   |
|------------|--|------------|
| CEN-12     | Centralization privileges of BITCOIN SPARK | Medium # 🟡 |

Vulnerability 0 : No important security issue detected.

Threat level: Low



```

194     _balances[account] += amount;
195   }
196   emit Transfer(address(0), account, amount);
197   _afterTokenTransfer(address(0), account, amount);
198 }
199
200
201
202
203
204 function _approve(address owner, address spender, uint256 amount) internal virtual {
205     require(owner != address(0), "ERC20: approve from the zero address");
206     require(spender != address(0), "ERC20: approve to the zero address");
207
208     _allowances[owner][spender] = amount;
209     emit Approval(owner, spender, amount);
210 }
211
212
213 function _spendAllowance(address owner, address spender, uint256 amount) internal virtual {
214     uint256 currentAllowance = allowance(owner, spender);
215     if (currentAllowance != type(uint256).max) {
216         require(currentAllowance >= amount, "ERC20: insufficient allowance");
217         unchecked {
218             _approve(owner, spender, currentAllowance - amount);
219         }
220     }
221 }
  
```

## MANUAL REVIEW

**Bitcoin Spark:** is a cutting-edge cryptocurrency designed to provide users with unparalleled security, speed, and scalability. This technical whitepaper delves into the innovative architecture and features that set Bitcoin Spark apart from existing cryptocurrencies, such as Bitcoin and Ethereum. By leveraging state-of-the-art blockchain technology, advanced consensus algorithms, and a unique governance model, Bitcoin Spark aims to revolutionize the way we perceive and utilize digital currencies in our everyday lives...

**TOKEN NAME: BITCOIN SPARK**

**Ticker: BTCS**

**Chain/Standard: Ethereum Network**

**LAUNGUGE: Solidity**



**The BITCOIN SPARK Platform Is Launching On the Ethereum Network**





# ISSUES CHECKING STATUS

Issue Description

Checking Status

|     |   |        |
|-----|---|--------|
| 1.  | Compiler errors.  | PASSED |
| 2.  | Race Conditions and reentrancy. Cross-Function Race Conditions. | PASSED |
| 3.  | Possible Delay In Data Delivery.                                | PASSED |
| 4.  | Oracle calls.   | PASSED |
| 5.  | Front Running.  | PASSED |
| 6.  | SOL Dependency.   | PASSED |
| 7.  | Integer Overflow And Underflow.                                 | PASSED |
| 8.  | DoS with Revert.  | PASSED |
| 9.  | Dos With Block Gas Limit.                                       | PASSED |
| 10. | Methods execution permissions.                                  | PASSED |
| 11. | Economy Model of the contract.                                  | PASSED |
| 12. | The Impact Of Exchange Rate On the sol Logic.                   | PASSED |
| 13. | Private use data leaks.   | PASSED |
| 14. | Malicious Event log.  | PASSED |
| 15. | Scoping and Declarations.                                       | PASSED |
| 16. | Uninitialized storage pointers.                                 | PASSED |
| 17. | Arithmetic accuracy.  | PASSED |
| 18. | Design Logic.   | PASSED |
| 19. | Cross-Function race Conditions                                  | PASSED |
| 20. | Save Upon Move contract Implementation and Usage.               | PASSED |
| 21. | Fallback Function Security                                      | PASSED |



**AUDIT RESULT**

**PASSED**

SMART CONTRACT AUDIT OF BITPAD



| Identifier | Definition                 | Severity  |
|------------|----------------------------|---|
| CEN-02     | Initial asset distribution | Minor  |

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

```
contract ERC20 is Context, IERC20, IERC20Metadata {  
    mapping(address => uint256) private _balances;  
  
    mapping(address => mapping(address => uint256)) private _allowances;
```

## RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.



## RECOMMENDATION

**Deployer and/or contract owner private keys are secured carefully.**

**Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.**

## ALLEVIATION

**The BITPAD project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project**



| Identifier | Definition               | Severity  |
|------------|--------------------------|---|
| COD-10     | Third Party Dependencies | Minor  |

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

## RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.



## DISCLAIMERS

Vital Block provides the easy-to-understand audit of Solidity, Move and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

## CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

## NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way



to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

**FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.**

### **TECHNICAL DISCLAIMER**

**ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.**

**WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.**

### **TIMELINESS OF CONTENT**

**The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.**



## **LINKS TO OTHER WEBSITES**

**This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.**



## ABOUT VITAL BLOCK

**Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.**

**Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.**

**Website:** <https://Vitalblock.org>

**Email:** [info@vitalblock.org](mailto:info@vitalblock.org)

**GitHub:** <https://github.com/vital-block>

**Telegram (Engineering):** [https://t.me/vital\\_block](https://t.me/vital_block)

**Telegram (Onboarding):** [https://t.me/vitalblock\\_cmo](https://t.me/vitalblock_cmo)





**vital-block**



**info@vitalblock.org**



**www.Vitalblock.org**



Vital Block Dedicated to securing Public and Private Blockchain Ecosystem