Cyberscope

# Audit Report

# BUTANE

June 2023

| | |
|---|---|
| Network | BSC |
| Address | 0x849b8cA35485d28B8fe4EE8d695D98491e024024 |
| Audited by | © cyberscope |

# Analysis

Critical Medium Minor / Informative Pass

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | ST | Stops Transactions | Unresolved |
| ● | OCTD | Transfers Contract's Tokens | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | ULTW | Transfers Liquidity to Team Wallet | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

# Diagnostics

Critical Medium Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | RVD | Redundant Variable Declaration | Unresolved |
| ● | RED | Redundant Event Declaration | Unresolved |
| ● | IDI | Immutable Declaration Improvement | Unresolved |
| ● | L02 | State Variables could be Declared Constant | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L11 | Unnecessary Boolean equality | Unresolved |

# Table of Contents

**Analysis** 1
**Diagnostics** 2
**Table of Contents** 3
**Review** 4
Audit Updates 4
Source Files 4
**Findings Breakdown** 5
ST - Stops Transactions 6
Description 6
Recommendation 6
RVD - Redundant Variable Declaration 7
Description 7
Recommendation 7
RED - Redundant Event Declaration 8
Description 8
Recommendation 8
IDI - Immutable Declaration Improvement 9
Description 9
Recommendation 9
L02 - State Variables could be Declared Constant 10
Description 10
Recommendation 10
L04 - Conformance to Solidity Naming Conventions 11
Description 11
Recommendation 11
L09 - Dead Code Elimination 12
Description 12
Recommendation 13
L11 - Unnecessary Boolean equality 14
Description 14
Recommendation 14
**Functions Analysis** 15
**Inheritance Graph** 18
**Flow Graph** 19
**Summary** 20
**Disclaimer** 21
**About Cyberscope** 22

# Review

| | |
|---|---|
| **Contract Name** | BUTANE |
| **Compiler Version** | v0.8.19+commit.7dd6d404 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0x849b8ca35485d28b8fe4ee8d695d98491e024024 |
| **Address** | 0x849b8ca35485d28b8fe4ee8d695d98491e024024 |
| **Network** | BSC |
| **Symbol** | WBBC |
| **Decimals** | 18 |
| **Total Supply** | 10.000.000 |

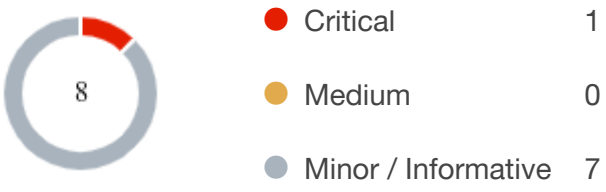
## Audit Updates

| | |
|---|---|
| **Initial Audit** | 28 May 2023 https://github.com/cyberscope-io/audits/blob/main/13-btc/v1/audit.pdf |
| **Corrected Phase 2** | 09 Jun 2023 |

## Source Files

| | |
|---|---|
| **Filename** | SHA256 |
| **BUTANE.sol** | 3d5acbc334045e8d5b5a2e33a54c1da41302773d3d1746ba535aeb8e6401e478 |

# Findings Breakdown




| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 1 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 7 | 0 | 0 | 0 |

## ST - Stops Transactions

| | |
|---|---|
| **Criticality** | Critical |
| **Location** | BUTANE.sol#L445 |
| **Status** | Unresolved |

### Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled, the owner will not be able to disable them again.

```
require(
    tradingEnabled ||
        _isExcludedFromFee[sender] ||
        _isExcludedFromFee[recipient],
    "Trading not yet enabled!"
);
```

### Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

# RVD - Redundant Variable Declaration

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L226 |
| **Status** | Unresolved |

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares some variables that are used in a meaningful way by the contract. As a result, these variables are redundant.

```
address currentRouter;
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

# RED - Redundant Event Declaration

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L217 |
| **Status** | Unresolved |

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares some events that are not used by the contract. As a result, these events are redundant.

```
event Log(string, uint256);
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

## IDI - Immutable Declaration Improvement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L241 |
| **Status** | Unresolved |

### Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
currentRouter
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L223,224,225 |
| **Status** | Unresolved |

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private tTotal = 30_000_000 ether
string private _name = "BUTANE Token"
string private _symbol = "WBBC"
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L217,218,219 |
| **Status** | Unresolved |

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
string public constant contractVersion = "4"
string public constant contractDev = "CFG"
string public constant contractEdition = "SAFU"
```

### Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

# L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L496 |
| **Status** | Unresolved |

## Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero
address");

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

        uint256 accountBalance = _balances[account];
...
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

## Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L11 - Unnecessary Boolean equality

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L589,602 |
| **Status** | Unresolved |

### Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(
            _isExcludedFromFee[account] != true,
            "The wallet is already excluded!"
        )

require(
            _isExcludedFromFee[account] != false,
            "The wallet is already included!"
        )
```

### Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

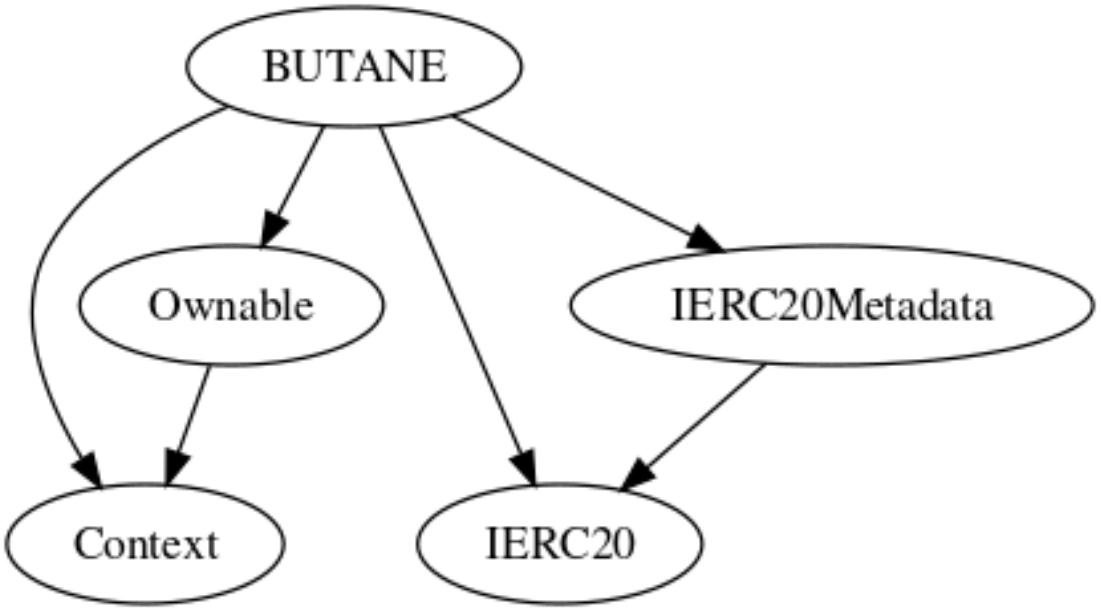
# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | _checkOwner | Internal | | |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |

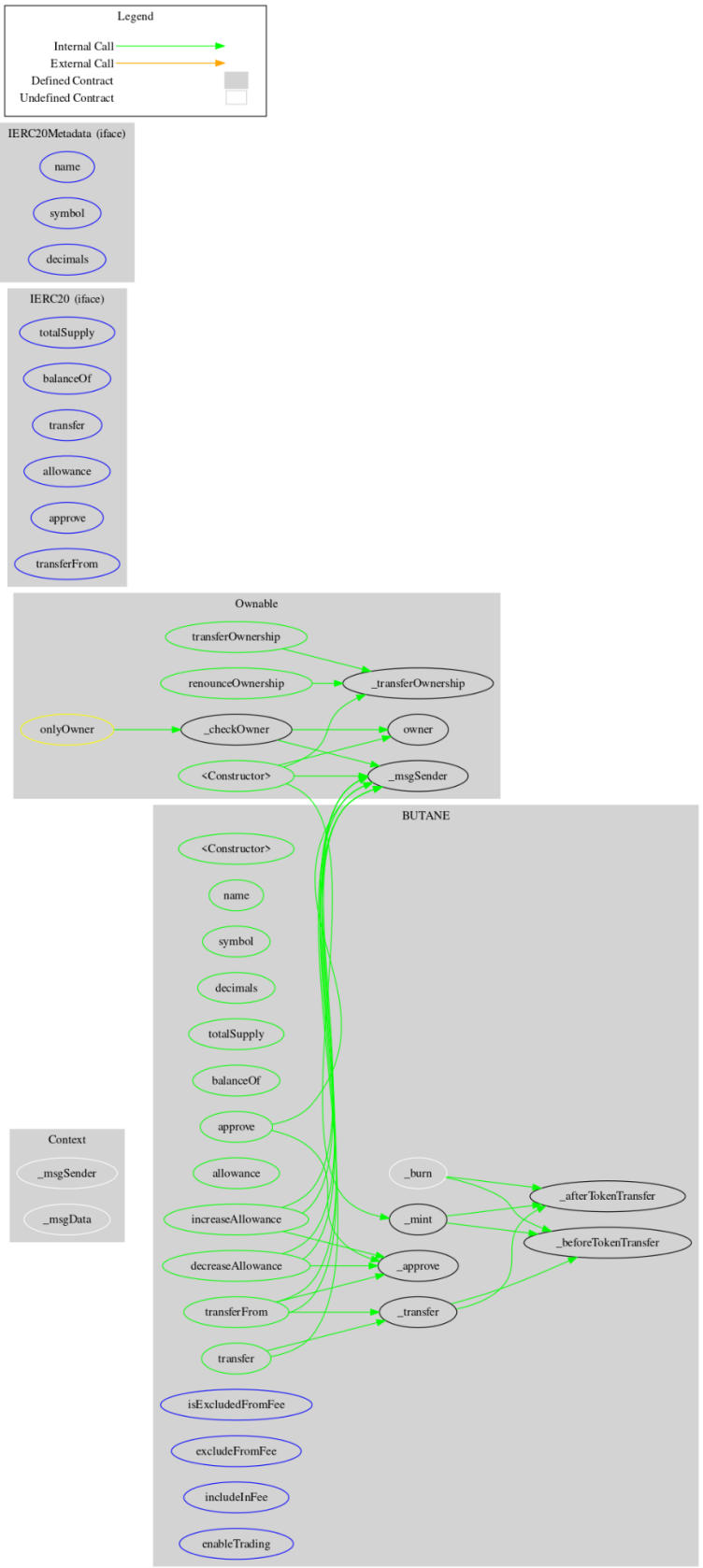
| | | | | |
|---|---|---|---|---|
| | transferFrom | External | ✓ | - |
| | | | | |
| **IERC20Metadata** | Interface | IERC20 | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| **BUTANE** | Implementation | Context, IERC20, IERC20Metadata, Ownable | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |

| | | | | |
|---|---|---|---|---|
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | _afterTokenTransfer | Internal | ✓ | |
| | isExcludedFromFee | External | | - |
| | excludeFromFee | External | ✓ | onlyOwner |
| | includeInFee | External | ✓ | onlyOwner |
| | enableTrading | External | ✓ | onlyOwner |

# Inheritance Graph



# Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
transferOwnership
renounceOwnership
_transferOwnership
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
BUTANE
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
allowance
increaseAllowance
decreaseAllowance
transferFrom
transfer
isExcludedFromFee
excludeFromFee
includeInFee
enableTrading
_burn
_mint
_approve
_transfer
_afterTokenTransfer
_beforeTokenTransfer
Context
_msgSender
_msgData

# Summary

BUTANE contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions. A multi-wallet signing pattern will provide security against potential hacks.

# Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. 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 Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



**The Cyberscope team**

https://www.cyberscope.io

Cyberscope

# Audit Report

# BUTANE

May 2023

Network BSC

Address 0x23c38197AcAA488084C61A30b2FAdd4956CE30d5

Audited by © cyberscope

# Analysis

Critical Medium Minor / Informative Pass

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | ST | Stops Transactions | Unresolved |
| ● | OCTD | Transfers Contract's Tokens | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | ULTW | Transfers Liquidity to Team Wallet | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

# Diagnostics

Critical Medium Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | RVD | Redundant Variable Declaration | Unresolved |
| ● | RED | Redundant Event Declaration | Unresolved |
| ● | IDI | Immutable Declaration Improvement | Unresolved |
| ● | L02 | State Variables could be Declared Constant | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L11 | Unnecessary Boolean equality | Unresolved |

# Table of Contents

**Analysis** 1
**Diagnostics** 2
**Table of Contents** 3
**Review** 4
Audit Updates 4
Source Files 4
**Findings Breakdown** 5
ST - Stops Transactions 6
Description 6
Recommendation 6
RVD - Redundant Variable Declaration 7
Description 7
Recommendation 7
RED - Redundant Event Declaration 8
Description 8
Recommendation 8
IDI - Immutable Declaration Improvement 9
Description 9
Recommendation 9
L02 - State Variables could be Declared Constant 10
Description 10
Recommendation 10
L04 - Conformance to Solidity Naming Conventions 11
Description 11
Recommendation 11
L09 - Dead Code Elimination 12
Description 12
Recommendation 13
L11 - Unnecessary Boolean equality 14
Description 14
Recommendation 14
**Functions Analysis** 15
**Inheritance Graph** 18
**Flow Graph** 19
**Summary** 20
**Disclaimer** 21
**About Cyberscope** 22

# Review

| | |
|---|---|
| **Contract Name** | BUTANE |
| **Compiler Version** | v0.8.19+commit.7dd6d404 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0x23c38197acaa488084c61a30b2fadd4956ce30d5 |
| **Address** | 0x23c38197acaa488084c61a30b2fadd4956ce30d5 |
| **Network** | BSC |
| **Symbol** | WBBC |
| **Decimals** | 18 |
| **Total Supply** | 30,000,000 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 28 May 2023 |

## Source Files

| | |
|---|---|
| **Filename** | SHA256 |
| **BUTANE.sol** | fc86b128c02a0466b425b15cc32edbf26c693a3d42cbdcf6fef879997e0f8118 |

# Findings Breakdown




| | |
|---|---|
| Critical | 1 |
| Medium | 0 |
| Minor / Informative | 7 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 1 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 7 | 0 | 0 | 0 |

## ST - Stops Transactions

| | |
|---|---|
| **Criticality** | Critical |
| **Location** | BUTANE.sol#L440 |
| **Status** | Unresolved |

### Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled, the owner will not be able to disable them again.

```
require(
    tradingEnabled ||
        _isExcludedFromFee[sender] ||
        _isExcludedFromFee[recipient],
    "Trading not yet enabled!"
);
```

### Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

# RVD - Redundant Variable Declaration

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L222 |
| **Status** | Unresolved |

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares some variables that are used in a meaningful way by the contract. As a result, these variables are redundant.

```
address currentRouter;
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

## RED - Redundant Event Declaration

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L213 |
| **Status** | Unresolved |

### Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares some events that are not used by the contract. As a result, these events are redundant.

```
event Log(string, uint256);
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

## IDI - Immutable Declaration Improvement

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L237 |
| **Status** | Unresolved |

### Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
currentRouter
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L219,220,221 |
| **Status** | Unresolved |

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private tTotal = 30_000_000 ether
string private _name = "BUTANE Token"
string private _symbol = "WBBC"
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L209,210,211 |
| **Status** | Unresolved |

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
string public constant contractVersion = "4"
string public constant contractDev = "CFG"
string public constant contractEdition = "SAFU"
```

### Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

# L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L492 |
| **Status** | Unresolved |

## Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero
address");

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

        uint256 accountBalance = _balances[account];
...
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

## Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L11 - Unnecessary Boolean equality

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | BUTANE.sol#L585,598 |
| **Status** | Unresolved |

### Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(
            _isExcludedFromFee[account] != true,
            "The wallet is already excluded!"
        )

require(
            _isExcludedFromFee[account] != false,
            "The wallet is already included!"
        )
```

### Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

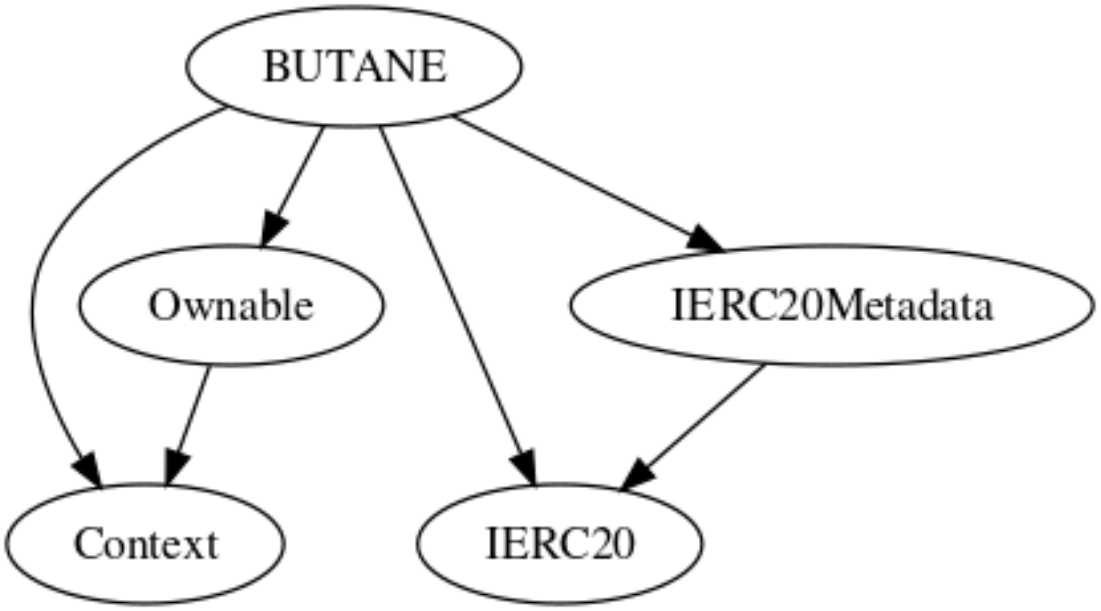
# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | _checkOwner | Internal | | |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |

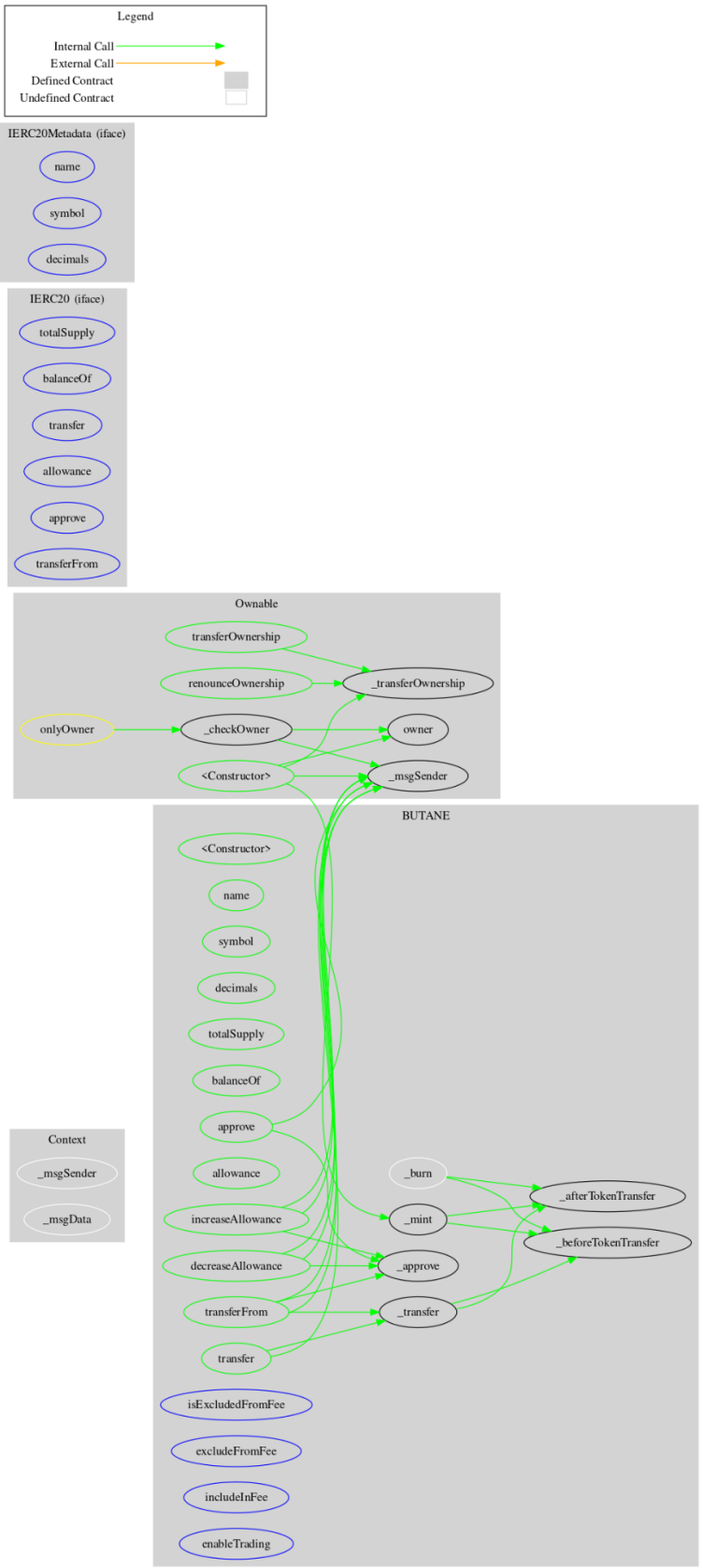
| | | | | |
|---|---|---|---|---|
| | transferFrom | External | ✓ | - |
| | | | | |
| **IERC20Metadata** | Interface | IERC20 | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| **BUTANE** | Implementation | Context, IERC20, IERC20Metadata, Ownable | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |

| | | | | |
|---|---|---|---|---|
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | _afterTokenTransfer | Internal | ✓ | |
| | isExcludedFromFee | External | | - |
| | excludeFromFee | External | ✓ | onlyOwner |
| | includeInFee | External | ✓ | onlyOwner |
| | enableTrading | External | ✓ | onlyOwner |

# Inheritance Graph



# Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
transferOwnership
renounceOwnership
_transferOwnership
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
BUTANE
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
allowance
increaseAllowance
decreaseAllowance
transferFrom
transfer
isExcludedFromFee
excludeFromFee
includeInFee
enableTrading
_burn
_mint
_approve
_transfer
_afterTokenTransfer
_beforeTokenTransfer
Context
_msgSender
_msgData

# Summary

BUTANE contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions. A multi-wallet signing pattern will provide security against potential hacks.

# Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. 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 Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



**The Cyberscope team**

https://www.cyberscope.io