FBcoin.live Coin

Smart Contract Audit Report

SYSFIXED

18 Jan 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
FBcoin.live Coin	FB	Binance Smart Chain

Addresses

Contract address	0x770f030fdbf63ebf1c939de8bcff8943c2c2d454
Contract deployer address	0x189ACf59cCf0844C67058d80Ebf2f4D47560BfE5

Project Website

https://github.com/cubeforex/FBBank

Codebase

https://bscscan.com/address/0x770f030fdbf63ebf1c939de8bcff8943c2c2d454#code

SUMMARY

FBBank is the Web3 world's first decentralized crypto fund. It uses the DAO organization form of the blockchain to build an investment system, which is an unprecedented new financial organization. The FBBank project is committed to solving the shortcomings of ordinary investors, focusing on solving various pain points of ordinary investors, and helping investors to effectively avoid investment decision mistakes caused by various psychological factors. With its excellent results, FBBank has successfully obtained strategic cooperation support from well-known Web3 wallets and DeFi institutions such as Bitkeep, Onto, OKC, CoinHub, CherrySwap, HyperPay, and Ivy Market.

Contract Summary

Documentation Quality

FBcoin.live Coin provides a very poor documentation with standard of solidity base code.

- The technical description is provided unclear and disorganized.

Code Quality

The Overall quality of the basecode is poor.

- Solidity basecode and rules are unclear and disorganized by FBcoin.live Coin.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 71, 86, 97, 106, 119, 128, 137, 147 and 153.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10.
- SWC-111 | It is recommended to use alternatives to the deprecated constructions on lines 42, 87, 88, 89, 90, 99, 107, 108, 109, 110, 111, 120, 121, 129, 130, 138, 139 and 148.

CONCLUSION

We have audited the FBcoin.live Coin project released on January 2022 to find issues and identify potential security vulnerabilities in FBcoin.live Coin project. This process is used to find technical issues and security loopholes that may be found in smart contracts.

The security audit report yielded unsatisfactory results, discovering medium-risk and low-risk issues.

Writing a contract that does not follow the Solidity style guide can pose a significant risk. The serious and low problems we found in the smart contract are the built-in symbol "assert" shadowing and the definition "assert" using the same name as a built-in symbol. Reserved names should not be used to avoid confusion. Low-risk found are a floating pragma is set, and the "throw" keyword is deprecated. The current pragma Solidity directive is ""^0.4.12"". Specifying a fixed compiler version is recommended to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code. "throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

We were recommended to keep being aware of investing in this risky smart contract.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	PASS
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	PASS
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	ISSUE FOUND
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Monday Jan 17 2022 00:20:43 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Jan 18 2022 21:52:04 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	FB.sol

Detected Issues

ID	Title	Severity	Status
SWC-000	BUILTIN SYMBOL "ASSERT" SHADOWING	medium	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged

SWC-100	FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged
SWC-111	USE OF THE "THROW" KEYWORD IS DEPRECATED.	low	acknowledged

SWC-000 | BUILTIN SYMBOL "ASSERT" SHADOWING

LINE 40

medium SEVERITY

Definition "assert" uses the same name as a built-in symbol. Reserved names should not be used to avoid confusion.

Source File

- FB.sol

Locations

```
39
40   function assert(bool assertion) internal {
41   if (!assertion) {
42   throw;
43   }
44
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 71

low SEVERITY

The function definition of "FB" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
70   /* Initializes contract with initial supply tokens to the creator of the contract */
71   function FB(
72   uint256 initialSupply,
73   string tokenName,
74   uint8 decimalUnits,
75
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 86

low SEVERITY

The function definition of "transfer" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
85   /* Send coins */
86   function transfer(address _to, uint256 _value) {
87   if (_to == 0x0) throw;                               // Prevent transfer to 0x0
address. Use burn() instead
88   if (_value <= 0) throw;
89   if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has
enough
90
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 97

low SEVERITY

The function definition of "approve" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
96   /* Allow another contract to spend some tokens in your behalf */
97   function approve(address _spender, uint256 _value)
98   returns (bool success) {
99   if (_value <= 0) throw;
100   allowance[msg.sender][_spender] = _value;
101
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 106

low SEVERITY

The function definition of "transferFrom" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
105   /* A contract attempts to get the coins */
106   function transferFrom(address _from, address _to, uint256 _value) returns (bool
success) {
107   if (_to == 0x0) throw;                                // Prevent transfer to 0x0
address. Use burn() instead
108   if (_value <= 0) throw;
109   if (balanceOf[_from] < _value) throw;                 // Check if the sender has
enough
110
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 119

low SEVERITY

The function definition of "burn" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
118
119   function burn(uint256 _value) returns (bool success) {
120   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
121   if (_value <= 0) throw;
122   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
123
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 128

low SEVERITY

The function definition of "freeze" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
127
128   function freeze(uint256 _value) returns (bool success) {
129   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
130   if (_value <= 0) throw;
131   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
132
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 137

low SEVERITY

The function definition of "unfreeze" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
136
137   function unfreeze(uint256 _value) returns (bool success) {
138   if (freezeOf[msg.sender] < _value) throw;            // Check if the sender has
enough
139   if (_value <= 0) throw;
140   freezeOf[msg.sender] = SafeMath.safeSub(freezeOf[msg.sender],
_value);                      // Subtract from the sender
141
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 147

low SEVERITY

The function definition of "withdrawEther" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
146   // transfer balance to owner
147   function withdrawEther(uint256 amount) {
148   if(msg.sender != owner)throw;
149   owner.transfer(amount);
150   }
151
```

SWC-100 | FUNCTION VISIBILITY IS NOT SET (PRIOR TO SOLIDITY 0.5.0)

LINE 153

low SEVERITY

The function definition of "" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source File

- FB.sol

Locations

```
152   // can accept ether
153   function() payable {
154   }
155   }
156
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

low SEVERITY

The current pragma Solidity directive is ""^0.4.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- FB.sol

Locations

```
9
10   pragma solidity ^0.4.12;
11
12   /**
13   * Math operations with safety checks
14
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 42

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
41   if (!assertion) {
42   throw;
43   }
44   }
45   }
46
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 87

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
86   function transfer(address _to, uint256 _value) {
87   if (_to == 0x0) throw;                               // Prevent transfer to 0x0
address. Use burn() instead
88   if (_value <= 0) throw;
89   if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has
enough
90   if (balanceOf[_to] + _value < balanceOf[_to]) throw; // Check for overflows
91
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 88

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
87   if (_to == 0x0) throw;                                // Prevent transfer to 0x0
address. Use burn() instead
88   if (_value <= 0) throw;
89   if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has
enough
90   if (balanceOf[_to] + _value < balanceOf[_to]) throw; // Check for overflows
91   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
92
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 89

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
88   if (_value <= 0) throw;
89   if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has
enough
90   if (balanceOf[_to] + _value < balanceOf[_to]) throw; // Check for overflows
91   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
92   balanceOf[_to] = SafeMath.safeAdd(balanceOf[_to],
_value);                             // Add the same to the recipient
93
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 90

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
89   if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has
enough
90   if (balanceOf[_to] + _value < balanceOf[_to]) throw; // Check for overflows
91   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
92   balanceOf[_to] = SafeMath.safeAdd(balanceOf[_to],
_value);                             // Add the same to the recipient
93   Transfer(msg.sender, _to, _value);                   // Notify anyone listening that
this transfer took place
94
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 99

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
98   returns (bool success) {
99   if (_value <= 0) throw;
100   allowance[msg.sender][_spender] = _value;
101   return true;
102   }
103
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 107

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
106   function transferFrom(address _from, address _to, uint256 _value) returns (bool
success) {
107   if (_to == 0x0) throw;                                // Prevent transfer to 0x0
address. Use burn() instead
108   if (_value <= 0) throw;
109   if (balanceOf[_from] < _value) throw;                 // Check if the sender has
enough
110   if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
111
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 108

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
107   if (_to == 0x0) throw;                                // Prevent transfer to 0x0
address. Use burn() instead
108   if (_value <= 0) throw;
109   if (balanceOf[_from] < _value) throw;                 // Check if the sender has
enough
110   if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
111   if (_value > allowance[_from][msg.sender]) throw;     // Check allowance
112
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 109

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
108   if (_value <= 0) throw;
109   if (balanceOf[_from] < _value) throw;                 // Check if the sender has
enough
110   if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
111   if (_value > allowance[_from][msg.sender]) throw;     // Check allowance
112   balanceOf[_from] = SafeMath.safeSub(balanceOf[_from],
_value);                            // Subtract from the sender
113
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 110

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
109   if (balanceOf[_from] < _value) throw;                // Check if the sender has
enough
110   if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
111   if (_value > allowance[_from][msg.sender]) throw;     // Check allowance
112   balanceOf[_from] = SafeMath.safeSub(balanceOf[_from],
_value);                            // Subtract from the sender
113   balanceOf[_to] = SafeMath.safeAdd(balanceOf[_to],
_value);                              // Add the same to the recipient
114
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 111

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
110   if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
111   if (_value > allowance[_from][msg.sender]) throw;     // Check allowance
112   balanceOf[_from] = SafeMath.safeSub(balanceOf[_from],
_value);                          // Subtract from the sender
113   balanceOf[_to] = SafeMath.safeAdd(balanceOf[_to],
_value);                            // Add the same to the recipient
114   allowance[_from][msg.sender] = SafeMath.safeSub(allowance[_from][msg.sender],
_value);
115
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 120

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
119   function burn(uint256 _value) returns (bool success) {
120   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
121   if (_value <= 0) throw;
122   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
123   totalSupply = SafeMath.safeSub(totalSupply,_value);
// Updates totalSupply
124
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 121

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
120   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
121   if (_value <= 0) throw;
122   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
123   totalSupply = SafeMath.safeSub(totalSupply,_value);
// Updates totalSupply
124   Burn(msg.sender, _value);
125
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 129

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
128   function freeze(uint256 _value) returns (bool success) {
129   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
130   if (_value <= 0) throw;
131   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                      // Subtract from the sender
132   freezeOf[msg.sender] = SafeMath.safeAdd(freezeOf[msg.sender],
_value);                                // Updates totalSupply
133
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 130

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
129   if (balanceOf[msg.sender] < _value) throw;            // Check if the sender has
enough
130   if (_value <= 0) throw;
131   balanceOf[msg.sender] = SafeMath.safeSub(balanceOf[msg.sender],
_value);                     // Subtract from the sender
132   freezeOf[msg.sender] = SafeMath.safeAdd(freezeOf[msg.sender],
_value);                                // Updates totalSupply
133   Freeze(msg.sender, _value);
134
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 138

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
137   function unfreeze(uint256 _value) returns (bool success) {
138   if (freezeOf[msg.sender] < _value) throw;            // Check if the sender has
enough
139   if (_value <= 0) throw;
140   freezeOf[msg.sender] = SafeMath.safeSub(freezeOf[msg.sender],
_value);                      // Subtract from the sender
141   balanceOf[msg.sender] = SafeMath.safeAdd(balanceOf[msg.sender], _value);
142
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 139

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
138   if (freezeOf[msg.sender] < _value) throw;            // Check if the sender has
enough
139   if (_value <= 0) throw;
140   freezeOf[msg.sender] = SafeMath.safeSub(freezeOf[msg.sender],
_value);                      // Subtract from the sender
141   balanceOf[msg.sender] = SafeMath.safeAdd(balanceOf[msg.sender], _value);
142   Unfreeze(msg.sender, _value);
143
```

SWC-111 | USE OF THE "THROW" KEYWORD IS DEPRECATED.

LINE 148

low SEVERITY

"throw" is disallowed as of Solidity version 0.5.0. Use one of "revert()", "require()" or "assert()" instead

Source File

- FB.sol

Locations

```
147   function withdrawEther(uint256 amount) {
148   if(msg.sender != owner)throw;
149   owner.transfer(amount);
150   }
151
152
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Sysfixed's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.