Coinsult

Advanced Manual Smart Contract Audit

Project: Green Baby Floki

Website: https://gbfloki.io/

Low-Risk	Medium-Risk	High-Risk
5 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x5f12cEFbd4C1312c728E83b14bec08c96EFD5C0a

Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0xe823b0957d8897135750fd08485ee7524a035ec7	1,000,000,000,000	100.0000%

Source Code

Coinsult was comissioned by Green Baby Floki to perform an audit based on the following smart contract:

https://bscscan.com/address/0x5f12cEFbd4C1312c728E83b14bec08c96EFD5C0a#code

Manual Code Review

In this audit report we will highlight all these issues:

Low-Risk

5 low-risk code
issues found

Medium-Risk

0 medium-risk code
issues found

High-Risk

0 high-risk code
issues found

The detailed report continues on the next page...

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Wrong require statement

```
function updateSellFees(uint256 _marketingFeeOnSell, uint256 _rewardFeeOnSell) external onlyOwner {
    require(
         _marketingFeeOnSell + _rewardFeeOnSell &amp;lt;= 25,
        &amp;quot;Fees must be less than 25%&amp;quot;
    );
    rewardFeeOnSell    = _rewardFeeOnSell;
    marketingFeeOnSell = _marketingFeeOnSell;
    _totalFeesOnSell   =  marketingFeeOnSell + rewardFeeOnSell;
    emit UpdateSellFees( _marketingFeeOnSell, _rewardFeeOnSell);
}
```

Recommendation

Currently it’s not ‘less than 25’ but less or equal to 25.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Too many digits

Literals with many digits are difficult to read and review.

```
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue &gt;= 200000 &amp;&amp; newValue &lt;= 500000, &quot;gasForProcessing must be b
    require(newValue != gasForProcessing, &quot;Cannot update gasForProcessing to same value&quot;);
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

While `1_ether` looks like `1 ether`, it is `10 ether`. As a result, it's likely to be used incorrectly.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

No zero address validation for some functions

Detect missing zero address validation.

```
function changeOperatorWallet(address newAddress) external onlyOperator{
    require(newAddress != operator,"Operator Address is already same");
    operator = newAddress;
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

Bob calls `updateOwner` without specifying the `newOwner`, soBob loses ownership of the contract.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
function sendBNB(address payable recipient, uint256 amount) internal {
    require(address(this).balance &gt;= amount, "Address: insufficient balance");

    (bool success, ) = recipient.call{value: amount}("");
    require(success, "Address: unable to send value, recipient may have reverted");
}
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

Exploit scenario

```
contract ArbitrarySend{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

Owner privileges

- 🟢 Owner cannot set fees higher than 25%
- 🟢 Owner cannot pause trading
- 🟢 Owner cannot change max transaction amount
- 🟡 Owner can exclude from fees
- ⚠️ Owner is able to transfer without fees
- ⚠️ Owner is able to exclude addresses from dividend
- ⚠️ Owner can set max wallet balance (cannot be lower than 1% of the total supply)
- ⚠️ Owner can set max transaction limit (cannot be lower than 0.1% of the total supply)
- ⚠️ Owner can exclude addresses from the max transaction limit and max wallet balance.
- ⚠️ Owner can update minimum holding balance to be eligible to dividend

Extra notes by the team

No notes

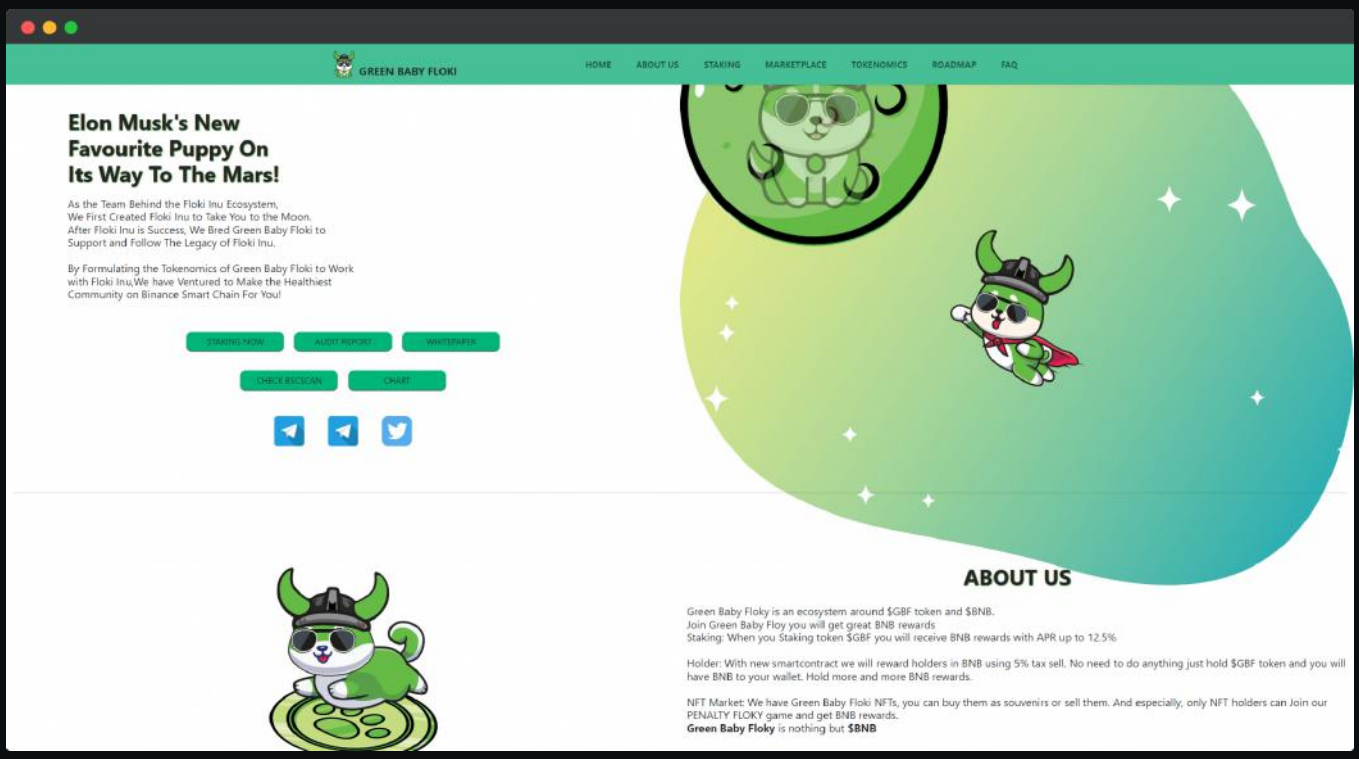
Contract Snapshot

```
constructor() ERC20("Green Baby Floki", "GBF")
{
    address newOwner = 0xe823B0957d8897135750fd08485eE7524A035eC7;
    transferOwnership(newOwner);
    operator=msg.sender;
    _mint(owner(), 1_000_000_000_000 * (10 ** 18));
    swapTokensAtAmount = totalSupply() / 5000;

    maxTransactionAmountBuy    = totalSupply() / 20;
    maxTransactionAmountSell   = totalSupply() / 20;
    maxWalletAmount            = totalSupply() / 10;
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



- Mobile Friendly
- Does not contain jQuery errors
- SSL Secured
- No major spelling errors

Project Overview

KYC verified by Coinsult partner

Not KYC verified by Coinsult

Green Baby Floki

Completed KYC Verification at a Coinsult partner

✔ Project Owner Identified

✔ Contract: 0x5f12cEFbd4C1312c728E83b14bec08c96EFD5C0a

Green Baby Floki

Audited by Coinsult.net

Date: 11 August 2022

✔ Advanced Manual Smart Contract Audit