Coinsult

Advanced Manual Smart Contract Audit

Bork Finance

Project: Bork Token

Website: http://borkfinance.dog

🟢 Low-Risk	🟡 Medium-Risk	🔴 High-Risk
4 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract Address

–

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

Not available

Source Code

Coinsult was comissioned by Bork Token to perform an audit based on the following smart contract:

https://github.com/GambleFinance/borkcontract/blob/main/BorkToken.sol

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

4 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.

Change boolean name

```
function updateFee(
    bool feeType,
    uint256 fee
) external onlyOwner {
    require(fee &lt;= 1000,
            &quot;Bork: Fee cannot be more then 10%&quot;);

    if (feeType) { // TRUE == BUY FEE
        BUY_FEE = fee;
    } else { // FALSE == SELL FEE
        SELL_FEE = fee;
    }

    emit FeeUpdated(_msgSender(), feeType, fee);
}
```

Recommendation

For readability it would be better to change ‘feeType’ to ‘isBuy’. True for buys, false for sells.

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

Too many digits

Literals with many digits are difficult to read and review.

```
uint256 feeAmount = amount.mul(fee).div(10000);
```

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 updateDevAddress(
    address payable _dev
) external onlyOwner {
    isExcludedFromFee[developmentAddress] = false;
    emit AddressExcluded(_msgSender(), developmentAddress, false);

    developmentAddress = _dev;
    isExcludedFromFee[developmentAddress] = true;

    emit AddressExcluded(_msgSender(), developmentAddress, true);
}
```

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.

Costly operations inside a loop

Costly operations inside a loop might waste gas, so optimizations are justified.

```
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) external onlyOwner {
    for (uint256 i = 0; i &lt; _accounts.length; i++) {
        isExcludedFromFee[_accounts[i]] = _excluded;

        emit AddressExcluded(_msgSender(), _accounts[i], _excluded);
    }
}
```

Recommendation

Use a local variable to hold the loop computation result.

Exploit scenario

```
contract CostlyOperationsInLoop{

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTORE`s, which might lead to an `out-of-gas`.

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 can mint new tokens

Extra notes by the team

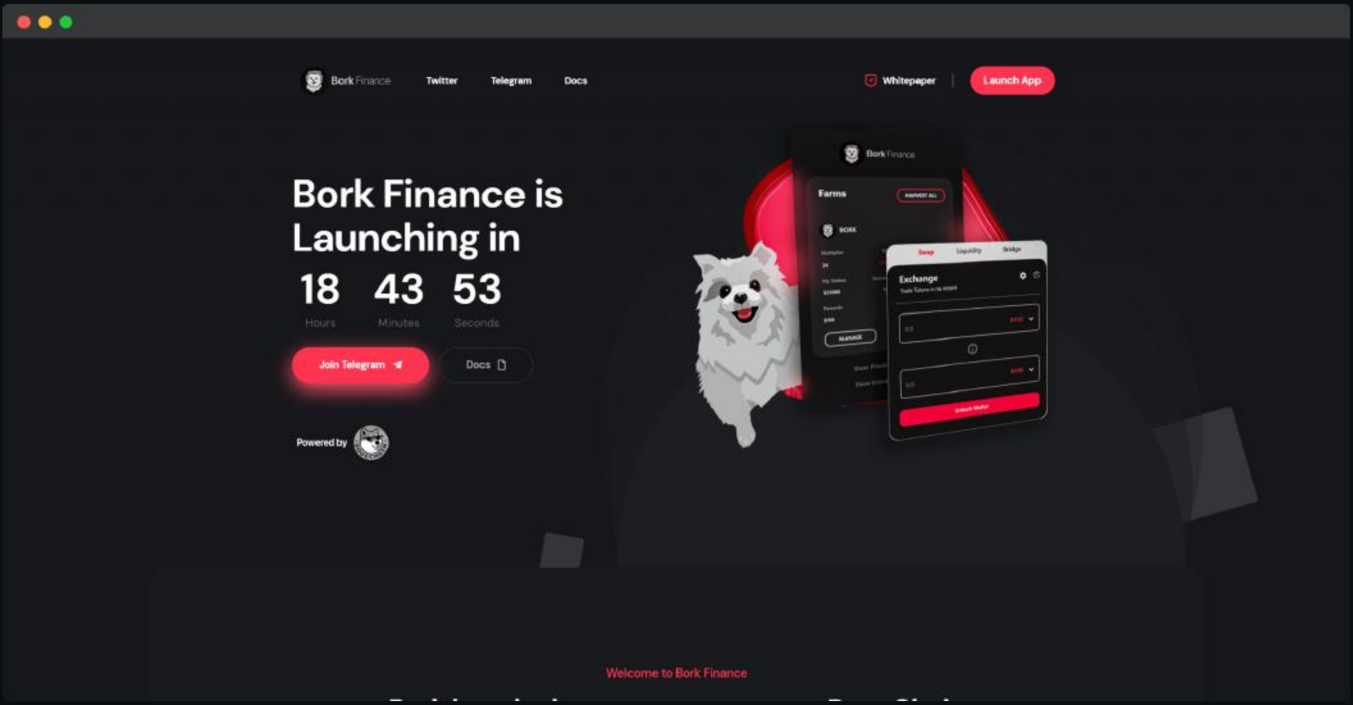
No notes

Contract Snapshot

```
contract Bork is ERC20Burnable, Ownable {
using SafeMath for uint256;

mapping(address =&gt; bool) public isExcludedFromFee;
mapping(address =&gt; bool) public isMinter;
mapping(address =&gt; bool) public whiteListedPair;

uint256 immutable public MAX_SUPPLY;
uint256 public BUY_FEE = 10;
uint256 public SELL_FEE = 10;

address payable public developmentAddress;

event TokenRecoverd(address indexed _user, uint256 _amount);
event FeeUpdated(address indexed _user, bool _feeType, uint256 _fee);
event ToggleV2Pair(address indexed _user, address indexed _pair, bool _flag);
event AddressExcluded(address indexed _user, address indexed _account, bool _flag);
event MinterRoleAssigned(address indexed _user, address indexed _account);
event MinterRoleRevoked(address indexed _user, address indexed _account);

constructor(string memory __name, string memory __symbol, uint256 _maxSupply, uint256 _initialSupply

    require(_initialSupply  0) {
        _mint(_msgSender(), _initialSupply);
    }
}
```

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

● Not KYC verified by Coinsult

Bork Token

Audited by Coinsult.net

Date: 19 August 2022

✓ Advanced Manual Smart Contract Audit