Wynaut

Security Code Review

https://twitter.com/VidarTheAuditor - 16 February 2021

Overview

Project Summary

Project Name	Wynaut
Description	Meme coin
Platform	**Binance Smart Chain**, Solidity
Contracts	
	• **https://bscscan.com/address/0x067a5ad3f0f91AcF512fFE66Ea77f37b4DcaaF18#code**

Executive Summary

Binance Smart Chain contracts were provided.

We have run extensive static analysis of the codebase as well as standard security assessment utilising industry approved tools.

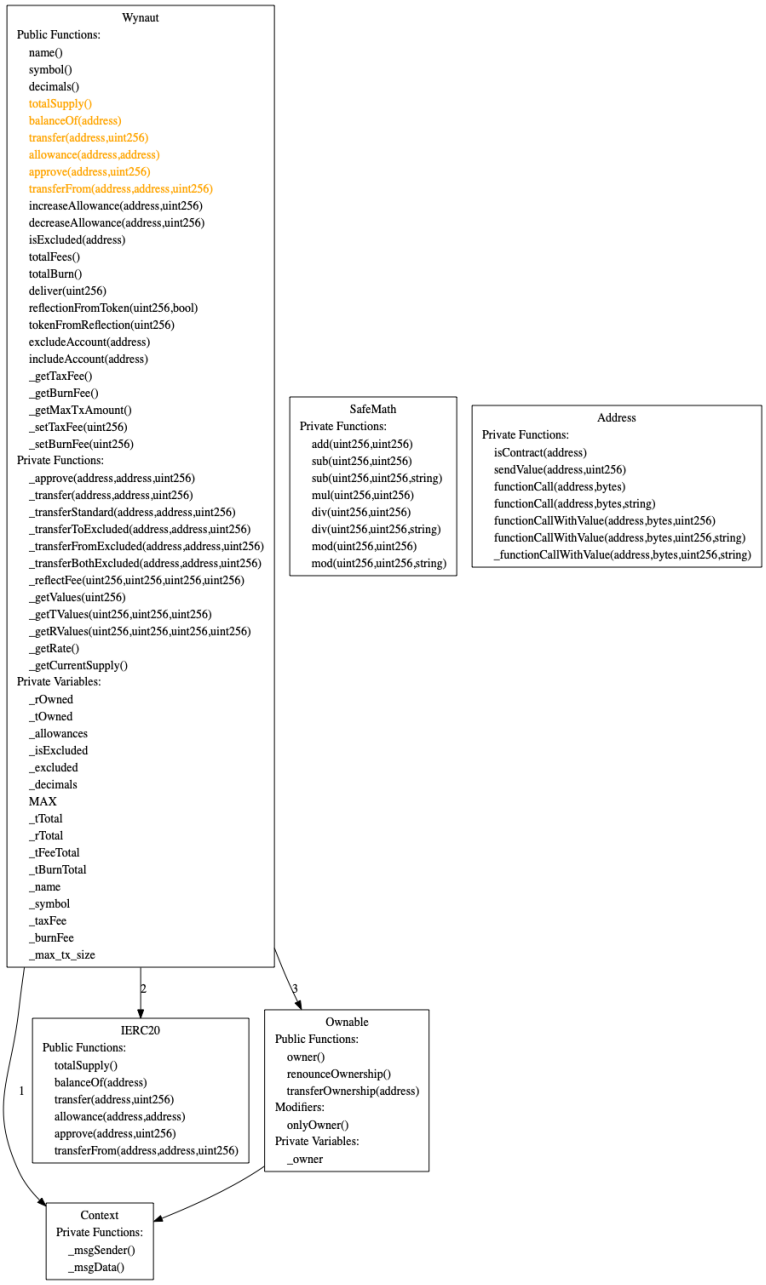
We have not found any critical vulnerabilities arising from a third party involvement.

Some recommendations were issued regarding the ownership structure of the currently deployed contract (more info can be found in Deployment section).

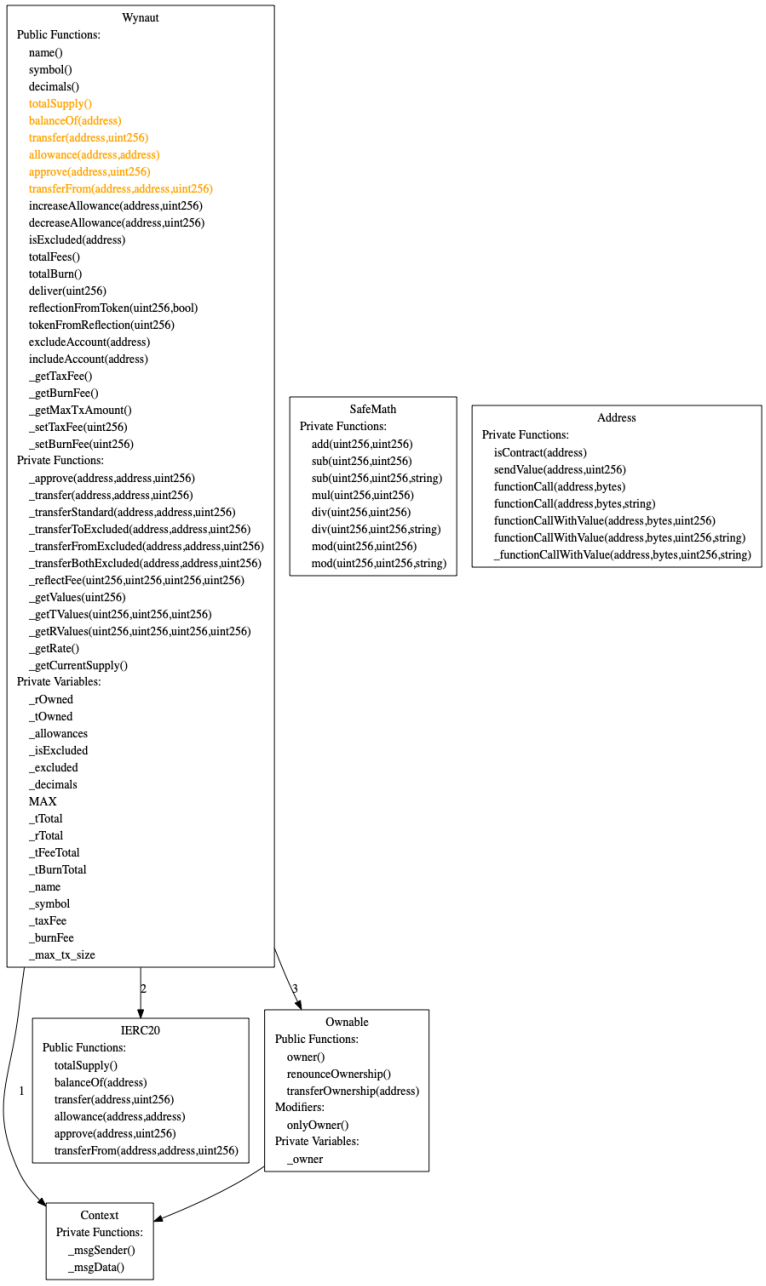
Disclaimer: The analysis did not include any tokenomics analysis (e.g. APY rates etc).

Architecture & Standards

Please find below the calling architecture of the reviewed contracts.



Wynaut contracts are fully BEP-20 compatible.



Findings

Number of contracts: 6 (including inherited ones)

Use: SafeMath

Name	# functions	ERCS	ERC20 info	Complex code	Features
SafeMath	8			No	
Address	7			No	Send ETH Assembly
Wynaut	51	ERC20	No Minting Approve Race Cond.	No	

For more info see Deployment section.

Static Analysis Findings

High issues: None

Medium issues: None

Low/Informational issues:

State variable could be declared constant:

```
Wynaut._max_tx_size (Wynaut.sol#483) should be constant
```

State variable could be declared constant.

Dynamic Tests

We have run fuzzing/property-based testing of Solidity smarts contracts. It was using sophisticated grammar-based fuzzing campaigns based on a contract ABI to falsify user-defined predicates or Solidity assertions.

There were also dynamic tests run on EVM byte code to detect common vulnerabilities including integer underflows, owner-overwrite-to-Ether-withdrawal, and others.

```
The analysis was completed successfully. No issues were detected.
```

No Issues were found.

Manual Checks

We found the BEP20 compatible token with reflect capabilities (functionality of https://etherscan.io/address/0xA1AFFfE3F4D611d252010E3EAf6f4D77088b0cd7#code is included)

The following functions are currently controlled by contract owner:

```
579     function excludeAccount(address account) external onlyOwner() {
580         require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap router.');
581         require(!_isExcluded[account], "Account is already excluded");
582         if(_rOwned[account] > 0) {
583             _tOwned[account] = tokenFromReflection(_rOwned[account]);
584         }
585         _isExcluded[account] = true;
586         _excluded.push(account);
587     }
588
        ftrace | funcSig
589     function includeAccount(address account) external onlyOwner() {
590         require(_isExcluded[account], "Account is already excluded");
591         for (uint256 i = 0; i < _excluded.length; i++) {
592             if (_excluded[i] == account) {
593                 _excluded[i] = _excluded[_excluded.length - 1];
594                 _tOwned[account] = 0;
595                 _isExcluded[account] = false;
596                 _excluded.pop();
597                 break;
598             }
599         }
600     }
```

```
        ftrace | funcSig
733     function _setTaxFee(uint256 taxFee) external onlyOwner() {
734         _taxFee = taxFee;
735     }
736
        ftrace | funcSig
737     function _setBurnFee(uint256 burnFee) external onlyOwner() {
738         _burnFee = burnFee;
739     }
```

The contract owner can set the critical parameters like taxFee and burnFee to any arbitrary numbers. Please see Deployment section for recommendations.

Automatic Tests

The project lacks any automatic testing and tests scripts. We did not run any functional tests provided by the team, due to lack of such scripts provided. Hence the full business logic functionality was not tested.

[**Recommendation**]: Create comprehensive test cases and implement them as scripts or mocha tests using the hardhat infrastructure.

[**Disclaimer**] There were no tests conducted testing full system functionality due to lack of proper test cases and/or test scripts.

Deployment & Contract Ownership

The contracts are currently deployed on BSC Mainnet:

- https://bscscan.com/address/0x067a5ad3f0f91AcF512fFE66Ea77f37b4DcaaF18#code

The current (block #4929934) is an ordinary address
0xda393CbcaEdb555E19B803Fd6ec5dD51BE67f521

As the owner can change critical contract parameters (**_taxFee & _burnFee**) it is strongly advised to put some governance on top of the contract ownership.

Recommendations:

- Deploy a governance on top of the ownership.
    - It could be a proper multi-sig controlled setup or full Compound like governance system - https://medium.com/compound-finance/compound-governance-5531f524cf68
    - There is also the possibility of controlling those functions via **TimeLock** contract with significant delay to make sure users are notified in advanced of any changes.

Disclaimer

The information appearing in this report is for general purposes only and is not intended to provide any legal security guarantees to any individual or entity. As one review is not enough to provide 100% security against any attacks or bugs, it is **strongly advisable** to conduct **more reviews or/and audits**.

The report does not provide personalised investment advice or recommendations, especially does not provide advice to conclude any transactions and it does not provide investment, financial, legal or tax advice.

We are not responsible or liable for any loss which results from the report.

The report should not be considered as an investment advice.