SquidGrow Token

Smart Contract Security Audit Report

15.08.2022

Versatile Finance Audit

Helping Businesses Incubate Ideas Into Reality

info@versatile.finance

Summary

Project Name: SquidGrow

Contract Address: 0x88479186bac914e4313389a64881f5ed0153c765

Client contact: SquidGrow Team

Blockchain: Binance smart chain

Language: Solidity

Project website: https://squidgrow.wtf

Buy Tax: 0-10 %

Sell Tax: 0-10 %

Token supply: 1,000,000,000,000,000

Token ticker: SquidGrow

Decimals: 19

Contract deployer address: 0x0f25bF0F93C094fE01bB26bb00670aA8Bdcafa8d

Contract's current owner address: 0x0f25bf0f93c094fe01bb26bb00670aa8bdcafa8d

Background

Versatile Finance was commissioned by Squid Grow Team to perform an audit of the smart contract.

https://bscscan.com/token/0x88479186bac914e4313389a64881f5ed0153c765

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

What is an audit

A smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. Versatile Finance manages this further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members.

Techniques and Methods

- The code quality
- Use of best practices
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.
- Code risk issue analysis and recommendations
- Ownership privileges
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.

used the following techniques, methods, and tools to review all the smart contracts.

Structural Analysis

We analyze the design patterns and structure of smart contracts. A thorough check is done to ensure the smart contract is structured in a way that will not have any issues.

Static Analysis

A static Analysis of Smart Contracts is done to identify contract vulnerabilities. In this step, a series of automated tools and manual testings are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code is done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts are manually analyzed line by line, and the logic is checked and compared with what's mentioned in the whitepaper to ensure everything's functioned as intended.

Gas Consumption

We check the behavior of smart contracts in production. Manual testings are done in DEXs to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Issue Categories

Every issue in this report has been assigned a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

NO High severity issues found

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues be fixed before moving to a live environment.

Medium-level severity issues

NO Medium severity issues found

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they can still be fixed. This can put users' funds at risk and has a medium to high probability of exploitation.

Low-level severity issues

NO Low severity issues found

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future. These issues have a low probability of occurring or may have a minimal impact.

Informational

NO informational issues found

These are severity four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Centralization

1 Centralization issue found

The owner can mark and unmark wallets as bot

```
ftrace | funcSig
function setisBot(bool _bool⬆, address _address⬆) external authorized {
    isBot[_address⬆] = _bool⬆;
}

ftrace | funcSig
```

Owner privileges

The owner can exclude wallets from fees

The owner can mark and unmark wallets as bot

The owner can add/remove internal wallets

The owner can enable disable antibots

The owner can distribute contract bnb balance with team wallets

```
ftrace | funcSig
function approvals(uint256 _na⬆, uint256 _da⬆) external authorized {
    performapprovals(_na⬆, _da⬆);
}
```

The owner can change auto LP receiver wallet

```
ftrace | funcSig
function setPairReceiver(address _address⬆) external authorized {
    liquidity_receiver = _address⬆;
}
```

The owner can start swapping

```
ftrace | funcSig
function setstartSwap(uint256 _input⬆) external authorized {
    startSwap = true;
    botOn = true;
    startedTime = block.timestamp.add(_input⬆);
}
```

The owner can change swap back settings

```
ftrace | funcSig
function setSwapBackSettings(bool enabled⬆, uint256 _threshold⬆)
    external
    authorized
{
    swapEnabled = enabled⬆;
    swapThreshold = _threshold⬆;
}
```

The owner can send contract BNB balance to default receiver address

```
ftrace | funcSig
function approval(uint256 percentage⬆) external authorized {
    uint256 amountBNB = address(this).balance;
    payable(default_receiver).transfer(amountBNB.mul(percentage⬆).div(100));
}
```

The owner can change max transaction and max wallet token amount minimum up to 0.5%

```
ftrace | funcSig
function setMaxes(uint256 _transaction⬆, uint256 _wallet⬆)
    external
    authorized
{
    uint256 newTx = (_totalSupply * _transaction⬆) / 10000;
    uint256 newWallet = (_totalSupply * _wallet⬆) / 10000;
    _maxTxAmount = newTx;
    _maxWalletToken = newWallet;
    require(
        newTx >= _totalSupply.mul(5).div(1000) &&
            newWallet >= _totalSupply.mul(5).div(1000),
        "Max TX and Max Wallet cannot be less than .5%"
    );
}
```

The owner can send any BEP20 tokens in contract to any wallet

```
ftrace | funcSig
function rescueBEP20(
    address _tadd⬆,
    address _rec⬆,
    uint256 _amt⬆
) external authorized {
    uint256 tamt = IBEP20(_tadd⬆).balanceOf(address(this));
    IBEP20(_tadd⬆).transfer(_rec⬆, tamt.mul(_amt⬆).div(100));
}
```

The owner can change all fee receiver address

```
ftrace | funcSig
function setDivisors(
    uint256 _distributor,
    uint256 _staking,
    uint256 _liquidity,
    uint256 _marketing
) external authorized {
    distributor_divisor = _distributor;
    staking_divisor = _staking;
    liquidity_divisor = _liquidity;
    marketing_divisor = _marketing;
}
```

The owner can change all fees maximum up to 10%

```
ftrace | funcSig
function setStructure(
    uint256 _liq,
    uint256 _mark,
    uint256 _stak,
    uint256 _burn,
    uint256 _tran
) external authorized {
    liquidityFee = _liq;
    marketingFee = _mark;
    stakingFee = _stak;
    burnFee = _burn;
    transferFee = _tran;
    totalFee = liquidityFee.add(marketingFee).add(stakingFee).add(burnFee);
    require(
        totalFee <= feeDenominator.div(10),
        "Tax cannot be more than 10%"
    );
}
```

The owner can change all team wallets address

Audit Results

Vulnerability Category	Status
Arbitrary Jump/Storage Write	pass
BRC20 Token standards	pass
Compiler errors	pass
Latest compiler version	pass
Authorization of function call to untrusted contract	pass
Dependence on Predictable Variables	pass
Ether/Token Theft	pass
Gas consumption	pass
Safemath features	pass
Fallback usage	pass
Deprecated items	pass
Redundant code	pass
Overriding variables	pass
Flash Loans	pass
Front Running	pass
Improper Events	pass
Improper Authorization Scheme	pass
Integer Over/Underflow	pass
Business logic issues	pass

Orcle issues	pass
Race Conditions	pass
Reentrancy	pass
Signature Issues	pass
Unbounded Loops	pass
Unused Code	pass
Pseudo random number generator (PRNG)	pass
Fake deposit	pass
Centralization of control	High centralization issue

Contract Description Table

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		

└	div	Internal 🔒		
└	mod	Internal 🔒		
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗

└	name	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Auth	Implementation			
└		Public ❗	🛑	NO❗
└	authorize	Public ❗	🛑	authorized
└	unauthorize	Public ❗	🛑	authorized
└	isOwner	Public ❗		NO❗
└	isAuthorized	Public ❗		NO❗
└	transferOwnership	Public ❗	🛑	authorized
└	renounceOwnership	External ❗	🛑	authorized
IFactory	Interface			

└	createPair	External ❗	🛑	NO❗
└	getPair	External ❗		NO❗
IRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
SquidGrow	Implementation	IBEP20, Auth		
└		Public ❗	🛑	Auth
└		External ❗	💵	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗

└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	viewisBot	Public ❗		NO❗
└	isCont	Internal 🔒		
└	approve	Public ❗	🛑	NO❗
└	getCirculatingSupply	Public ❗		NO❗
└	setFeeExempt	External ❗	🛑	authorized
└	setisBot	External ❗	🛑	authorized
└	setisInternal	External ❗	🛑	authorized
└	setbotOn	External ❗	🛑	authorized
└	syncContractPair	External ❗	🛑	authorized
└	approvals	External ❗	🛑	authorized
└	setPairReceiver	External ❗	🛑	authorized
└	setstartSwap	External ❗	🛑	authorized
└	setSwapBackSettings	External ❗	🛑	authorized

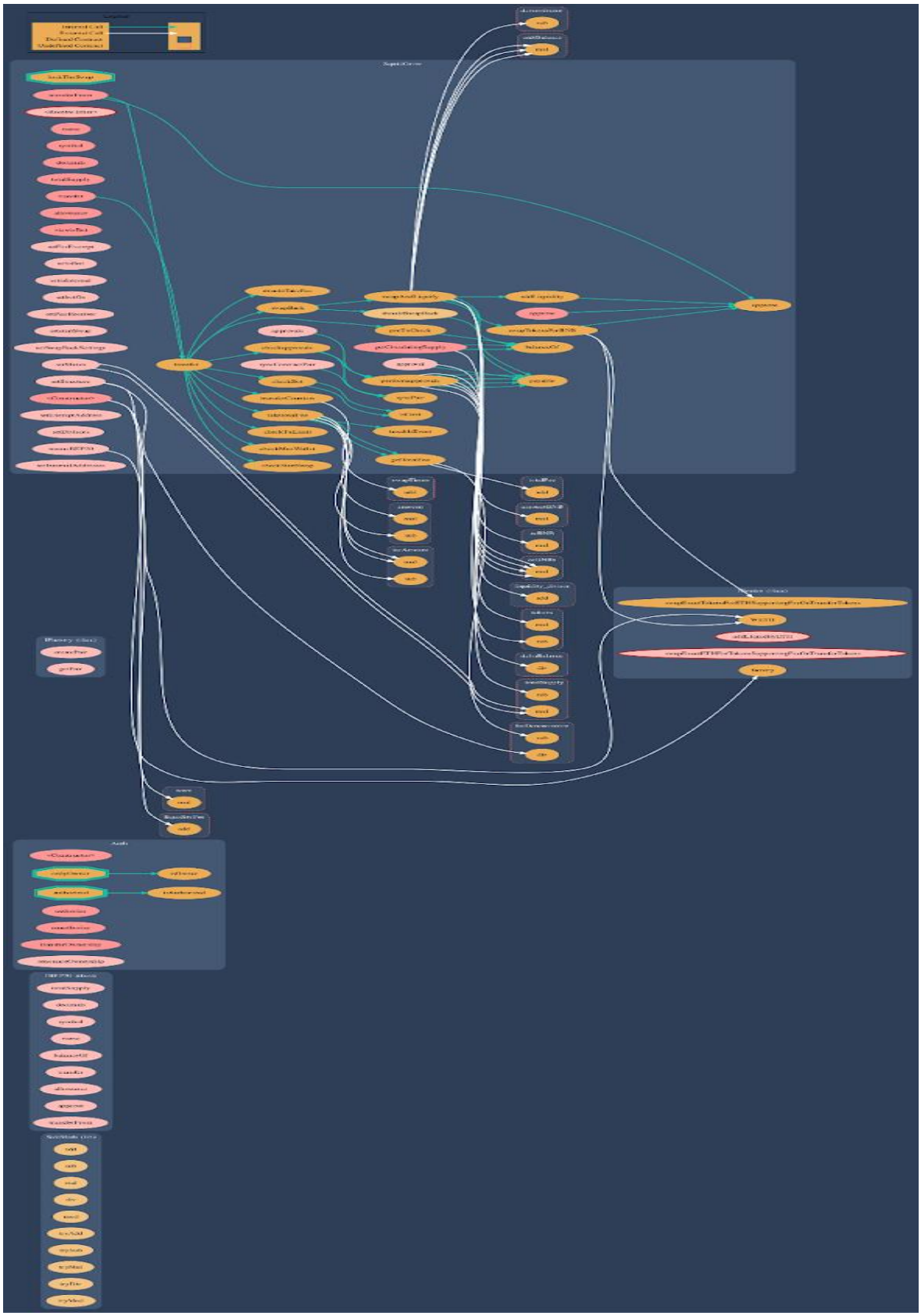
└	transferFrom	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	preTxCheck	Internal 🔒		
└	checkStartSwap	Internal 🔒		
└	checkMaxWallet	Internal 🔒		
└	transferCounters	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		
└	taxableEvent	Internal 🔒		
└	taketotalFee	Internal 🔒	🛑	
└	getTotalFee	Public ❗		NO❗
└	checkTxLimit	Internal 🔒		
└	checkBot	Internal 🔒	🛑	
└	approval	External ❗	🛑	authorized
└	checkapprovals	Internal 🔒	🛑	

└	setMaxes	External ❗	🛑	authorized
└	syncPair	Internal 🔒	🛑	
└	rescueBEP20	External ❗	🛑	authorized
└	setExemptAddress	External ❗	🛑	authorized
└	setDivisors	External ❗	🛑	authorized
└	performapprovals	Internal 🔒	🛑	
└	setStructure	External ❗	🛑	authorized
└	setInternalAddresses	External ❗	🛑	authorized
└	shouldSwapBack	Internal 🔒		
└	swapBack	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	

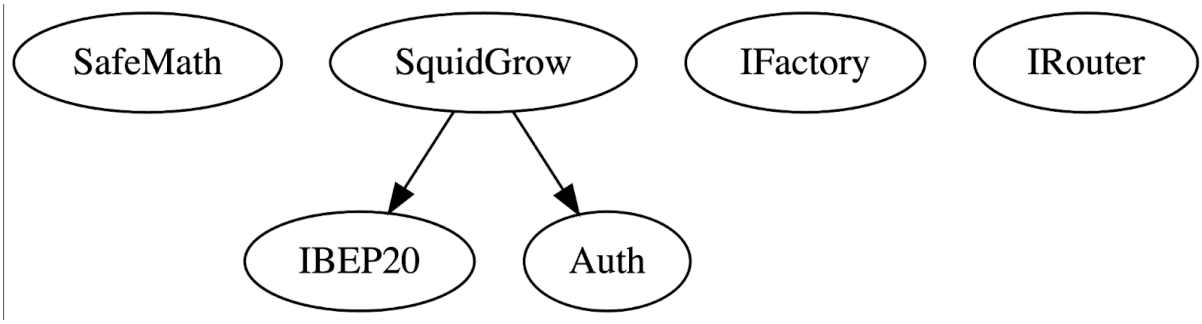
Legend

Symbol	Meaning
	Function can modify state
	Function is payable

Function Graph



Inheritance chart



Audit conclusion

Versatile Finance team has performed in-depth testing, line by line manual code review, and automated audit of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASS**

Number of risk issues: **1**

Solidity code functional issue level: **PASS**

Number of owner privileges: **14**

Centralization risk correlated to the active owner: **HIGH**

Smart contract active ownership: **YES**

Disclaimer

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 disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. 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 Versatile Finance and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Versatile Finance) owe no duty of care towards you or any other person, nor does Versatile Finance make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and Versatile Finance hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Versatile Finance hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Versatile Finance, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.