



Versatile Finance

Smart Contract Security Audit



ETH: SquidGrow Staking

27 November 2022



Summary

Project Name: SquidGrowStaking

Contract Address: 0xB2dA5bD106Efe383840383CaF8cefA3D4b1d0C61

Client contact: Squid Grow Team

Blockchain: Ethereum smart chain

Language: Solidity

Project website: <https://squidgrow.wtf/>

Referral: 0xdb79debda30f876bf57c616e275b01ab7c212c9e

Fee address: 0x1cddea8931ecf9499296863d949aecdd23b41a47

Squid Grow: 0x88479186bac914e4313389a64881f5ed0153c765

Contract owner address: 0x502be06628b472a54760F7000E40B169BF51A645

Contract deployer address: 0x502be06628b472a54760F7000E40B169BF51A645

Background

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

<https://etherscan.io/address/0xb2da5bd106efe383840383caf8cefa3d4b1d0c61>

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. The Versatile Finance manages this a step 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.

The following techniques, methods, and tools were used 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 completely manually analyzed line by line, and the logic is checked and compared with what's mentioned in the whitepaper to make sure 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 the 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

No Centralisation Issues Found





Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
Ownable	Implementation	Context		
L		Public 		NO 
L	owner	Public 		NO 
L	renounceOwnership	Public 		onlyOwner
L	transferOwnership	Public 		onlyOwner
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		



L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	mod	Internal 		
Address	Library			
L	isContract	Internal 		
L	sendValue	Internal 		
L	functionCall	Internal 		
L	functionCall	Internal 		
L	functionCallWithValue	Internal 		
L	functionCallWithValue	Internal 		
L	_functionCallWithValue	Private 		
IERC20	Interface			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 

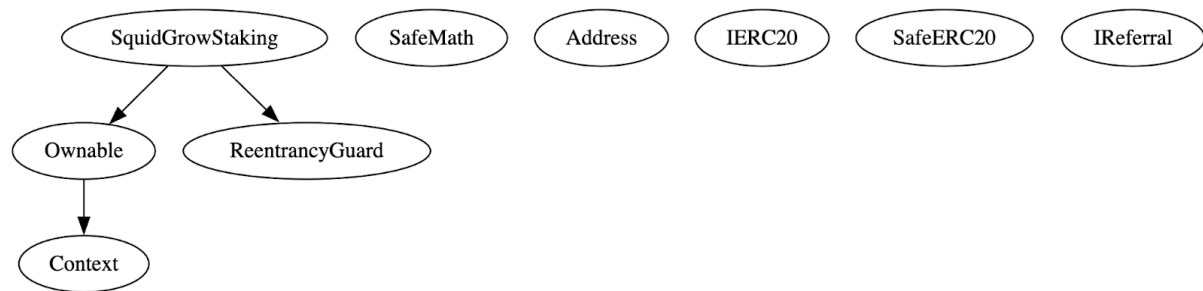
L	transferFrom	External !		NO !
SafeERC20	Library			
L	safeTransfer	Internal 		
L	safeTransferFrom	Internal 		
L	safeApprove	Internal 		
L	safeIncreaseAllowance	Internal 		
L	safeDecreaseAllowance	Internal 		
L	_callOptionalReturn	Private 		
ReentrancyGuard	Implementation			
L		Public !		NO !
IReferral	Interface			
L	recordReferral	External !		NO !
L	getReferrer	External !		NO !
SquidGrowStaking	Implementation	Ownable, ReentrancyGuard		
L		Public !		NO !

L	add	External !		onlyOwner
L	set	External !		onlyOwner
L	getMultiplier	Public !		NO !
L	pendingSquidgrow	External !		NO !
L	massUpdatePools	Public !		NO !
L	updatePool	Public !		NO !
L	deposit	Public !		nonReentrant
L	withdraw	Public !		nonReentrant
L	emergencyWithdraw	Public !		nonReentrant
L	safeSquidgrowTransfer	Internal 		
L	recoverUnsupportedToken	External !		onlyOwner
L	setFeeAddress	External !		onlyOwner
L	updateEmissionRate	External !		onlyOwner
L	whiteList	External !		onlyOwner
L	removeWhiteList	External !		onlyOwner
L	whiteListForWithdraw	External !		onlyOwner
L	removeWhiteListForWithdraw	External !		onlyOwner
L	blackList	External !		onlyOwner
L	removeBlackList	External !		onlyOwner

L	poolLength	External !		NO !
L	setReferralCommissionRate	External !		onlyOwner
L	setWithdrawFeeRate	External !		onlyOwner
L	payReferralCommission	Internal 		

Legend

Symbol	Meaning
	Function can modify state
	Function is payable



Owner privileges

The owner can add new pools to the contract

```
ftrace | funcSig
function add(
    IERC20 _lpToken↑,
    uint256 _allocPoint↑,
    uint16 _depositFeeBP↑
) external onlyOwner {
    require(
        _depositFeeBP↑ <= MAXIMUM_FEE_RATE,
        "depositFee mustn't be greater than 25%"
    );
    _lpToken↑.balanceOf(address(this));
    uint256 lastRewardBlock = block.number;
    totalAllocPoint = totalAllocPoint.add(_allocPoint↑);
    poolInfo.push(
        PoolInfo({
            lpToken: _lpToken↑,
            allocPoint: _allocPoint↑,
            lastRewardBlock: lastRewardBlock,
            accSquidgrowPerShare: 0,
            depositFeeBP: _depositFeeBP↑,
            lpSupply: 0
        })
    );
}
```

The owner can change pool details

```
ftrace | funcSig
function set(
    uint256 _pid↑,
    uint256 _allocPoint↑,
    uint16 _depositFeeBP↑
) external onlyOwner {
    require(
        _depositFeeBP↑ <= MAXIMUM_FEE_RATE,
        "depositFee mustn't be greater than 25%"
    );
    totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid↑].allocPoint).add(
        _allocPoint↑
    );
    if (poolInfo[_pid↑].allocPoint != 0) {
        poolInfo[_pid↑].allocPoint = _allocPoint↑;
    }
    poolInfo[_pid↑].depositFeeBP = _depositFeeBP↑;
}
```

The owner can get any BEP20 tokens from the contract

```
// Recover unsupported tokens that are sent accidentally
ftrace | funcSig
function recoverUnsupportedToken(address _addr↑, uint256 _amount↑)
    external
    onlyOwner
{
    require(_addr↑ != address(0), "non-zero address");
    IERC20(_addr↑).safeApprove(msg.sender, _amount↑);
    uint256 balance = IERC20(_addr↑).balanceOf(address(this));
    if (_amount↑ > 0 && _amount↑ <= balance) {
        IERC20(_addr↑).safeTransfer(msg.sender, _amount↑);
    }
}
```

The owner can change the fee address

```
// set fee address
ftrace | funcSig
function setFeeAddress(address _feeAddress↑) external onlyOwner {
    require(_feeAddress↑ != address(0), "non-zero");
    feeAddress = _feeAddress↑;
}
```

The owner can change per block reward rate

```
// update reward per block
ftrace | funcSig
function updateEmissionRate(uint256 _SquidgrowPerBlock↑) external onlyOwner {
    massUpdatePools();
    SquidgrowPerBlock = _SquidgrowPerBlock↑;
}
```

The owner can whitelist and blacklist wallets, (whitelisted wallets can stake without any fees and blacklisted wallets cannot get any rewards)

```
// add people to whitelist for deposit fee
ftrace | funcSig
function whiteList(address _addr↑) external onlyOwner {
    | isWhiteListed[_addr↑] = true;
}

// remove people from whitelist for deposit fee
ftrace | funcSig
function removeWhiteList(address _addr↑) external onlyOwner {
    | isWhiteListed[_addr↑] = false;
}
```

The owner can change referral commission maximum up to 5%

```
// Update referral commission rate by the owner
ftrace | funcSig
function setReferralCommissionRate(uint16 _referralCommissionRate↑)
    external
    onlyOwner
{
    require(
        _referralCommissionRate↑ <= MAXIMUM_REFERRAL_COMMISSION_RATE,
        "setReferralCommissionRate: invalid referral commission rate basis points"
    );
    referralCommissionRate = _referralCommissionRate↑;
}
```

The owner can add/remove wallets from blacklist

```
// add people to blacklist for reward
ftrace | funcSig
function blackList(address _addr↑) external onlyOwner {
    | isBlackListed[_addr↑] = true;
}

// remove people from blacklist
ftrace | funcSig
function removeBlackList(address _addr↑) external onlyOwner {
    | isBlackListed[_addr↑] = false;
}
```

The owner can change withdrawal fee up to 25%

```
// Update referral commission rate by the owner
ftrace | funcSig
function setWithdrawFeeRate(uint16 _withdrawFeeRate↑) external onlyOwner {
    require(
        _withdrawFeeRate↑ <= MAXIMUM_FEE_RATE,
        "WithdrawFeeRate mustn't be greater than 25%"
    );
    withdrawFeeRate = _withdrawFeeRate↑;
}
```

The owner can whitelist and remove wallets from withdrawal fee

```
// add people to whitelist for withdraw fee
ftrace | funcSig
function whiteListForWithdraw(address _addr↑) external onlyOwner {
    isWhiteListedForWithdraw[_addr↑] = true;
}

// remove people from whitelist for withdraw fee
ftrace | funcSig
function removeWhiteListForWithdraw(address _addr↑) external onlyOwner {
    isWhiteListedForWithdraw[_addr↑] = false;
}
```

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

Oracle 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

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 analyzed 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: **0**

Solidity code functional issue level: **PASS**

Number of owner privileges: **10**

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

Smart contract active ownership: **ACTIVE**

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.