Versatile Finance

Smart Contract Security Audit

ETH: SquidGrow Bridge

25 December 2022

Summary

Project Name: SquidGrowBSCBridgeContract

Contract Address: 0xB4056Fe94DCa4dcFcB6B9d830434D4AbC98496AD

Client contact: Squid Grow Team

Blockchain: Ethereum smart chain

Language: Solidity

Project website: http://squidgrow.wtf/

Governor: 0xd07054481051086511 4ad5a0b6a821a5bd2e7c49

System: 0x7B9f65e1B5F7a8031cBe25A78815A65244898260

Contract deployer address: 0x240f809D2F33AA3044Fbee70f18ED901Bd30c277

Contract's current owner address: 0x240f809d2f33aa3044fbee70f18ed901bd30c277

Background

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

https://etherscan.io/address/0xb4056fe94dca4dcfcb6b9d830434d4abc98496ad#code

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 is 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 centralization 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	_checkOwner	Internal 🔒		
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	

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❗
IERC20Permit	Interface			
L	permit	External ❗	🛑	NO❗
L	nonces	External ❗		NO❗
L	DOMAIN_SEPARATOR	External ❗		NO❗
Address	Library			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	

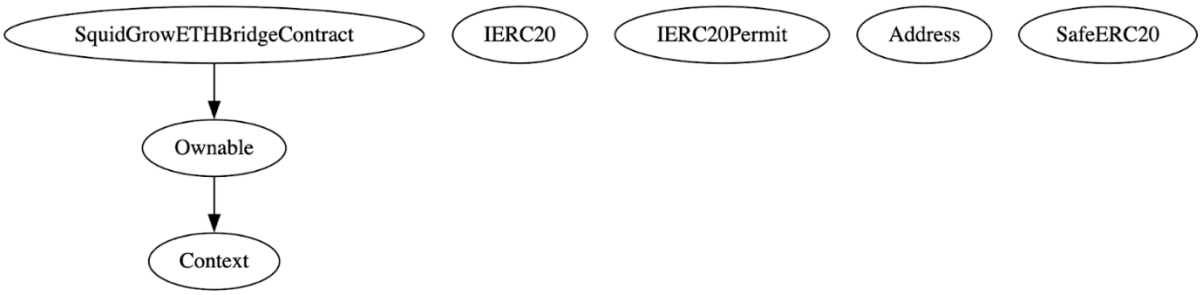
L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionStaticCall	Internal 🔒		
L	functionStaticCall	Internal 🔒		
L	functionDelegateCall	Internal 🔒	🛑	
L	functionDelegateCall	Internal 🔒	🛑	
L	verifyCallResultFromTarget	Internal 🔒		
L	verifyCallResult	Internal 🔒		
L	_revert	Private 🔒		
SafeERC20	Library			
L	safeTransfer	Internal 🔒	🛑	
L	safeTransferFrom	Internal 🔒	🛑	

L	safeApprove	Internal 🔒	🛑	
L	safeIncreaseAllowance	Internal 🔒	🛑	
L	safeDecreaseAllowance	Internal 🔒	🛑	
L	safePermit	Internal 🔒	🛑	
L	_callOptionalReturn	Private 🔐	🛑	
SquidGrowBSCBridgeContract	Implementation	Ownable		
L		Public ❗	🛑	NO❗
L	addTargetChain	External ❗	🛑	onlyOwner
L	removeTargetChain	External ❗	🛑	onlyOwner
L	excludeFromFees	External ❗	🛑	onlyGovernance
L	updateBridgeFee	External ❗	🛑	onlyGovernance
L	updateGovernor	External ❗	🛑	onlyGovernance
L	getBridgeFee	External ❗		NO❗
L	updateBridgeFeesAddress	External ❗	🛑	onlyGovernance

L	updateSystem	External ❗	🛑	onlyOwner
L	setProcessFees	External ❗	🛑	onlyOwner
L	getProcessFees	External ❗		NO❗
L	getBridgeStatus	External ❗		NO❗
L	updateBridgingState	External ❗	🛑	onlyOwner
L	calculateFees	Public ❗		NO❗
L	bridge	Public ❗	💵	whenNotPaused
L	bridgeBack	External ❗	🛑	whenNotPaused onlySystem
L	_sendBSC	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable


SquidGrowETHBridgeContract
IERC20
IERC20Permit
Address
SafeERC20
Ownable
Context

Owner privileges

The owner can add or remove new chains from the bridge.

```
ftrace | funcSig
function addTargetChain(uint256 _targetChain) external onlyOwner {
    require(
        supportedTargetChains[_targetChain] != true,
        "Already supported"
    );
    supportedTargetChains[_targetChain] = true;
    emit TargetChainAdded(_targetChain, block.timestamp);
}

ftrace | funcSig
function removeTargetChain(uint256 _targetChain) external onlyOwner {
    require(
        supportedTargetChains[_targetChain] != false,
        "Already not supported"
    );
    supportedTargetChains[_targetChain] = false;
    emit TargetChainRemoved(_targetChain, block.timestamp);
}
```

The owner can include/exclude wallets from fees.

```
ftrace | funcSig
function excludeFromFees(address account, bool exclude)
    external
    onlyGovernance
{
    require(_isExcludedFromFees[account] != exclude, "Already set");
    _isExcludedFromFees[account] = exclude;
    emit ExcludedFromFees(account, exclude);
}
```

The owner can update the bridge fee maximum of up to 10%

```
ftrace | funcSig
function updateBridgeFee(uint256 bridgeFee) external onlyGovernance {
    require(
        _bridgeFee <= MAX_BRIDGE_FEES,
        "Cannot update to more than MAX_BRIDGE_FEES"
    );
    _bridgeFee = bridgeFee;
    emit BridgeFeesUpdated(bridgeFee);
}
```

The owner can change the bridge fee address (the address that receives fees collected from the bridge contract)

```
ftrace | funcSig
function updateBridgeFeesAddress(address _bridgeFeesAddress)
    external
    onlyGovernance
{
    emit BridgeFeeAddressUpdated(bridgeFeesAddress, _bridgeFeesAddress);
    bridgeFeesAddress = _bridgeFeesAddress;
}
```

The owner can change the system address (Only the system address can call the bridge function)

```
ftrace | funcSig
function updateSystem(address payable _system) external onlyOwner {
    emit SytemUpdated(system, _system);
    system = _system;
}
```

The owner can change process fee to each chain

```
ftrace | funcSig
function setProcessFees(uint256 _targetChain, uint256 processFees)
    external
    onlyOwner
{
    _processFees[_targetChain] = processFees;
    emit ProcessFeesUpdated(_targetChain, processFees);
}
```

The owner can enable/disable bridge

```
ftrace | funcSig
function updateBridgingState(bool paused) external onlyOwner {
    require(_isBridgingPaused != paused, "Already set");
    _isBridgingPaused = paused;
    emit BridgingStateUpdated(paused);
}
```

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

Audit conclusion

Versatile Finance team has performed in-depth testings, 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: **7**

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

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.