



SMART CONTRACT SECURITY AUDIT OF **SHADOWFI**



SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Audit Introduction

Auditing Firm	InterFi Network
Audit Architecture	InterFi Echelon Auditing Standard
Language	Solidity
Client Firm	ShadowFi
Website	https://shadowfi.com/
Telegram	https://t.me/shadowfiofficial
Report Date	August 22, 2022

About ShadowFi

InterFi

Safe, simple, private spending, online and in person.

Smart Contract
Security Audit



Audit Summary

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ ShadowFi's solidity source codes have **LOW RISK SEVERITY**
- ❖ ShadowFi's smart contracts have **ACTIVE OWNERSHIP**
- ❖ ShadowFi's centralization risk correlated to the active owner is **HIGH**
- ❖ Presale contract privileges – **SET COST, SET DISCOUNT, SET MAX BUY, WITHDRAW**
- ❖ Liquidity lock contract privileges – **WITHDRAW BNB AND TOKEN, SHADOW BURST AND STRIKE**
- ❖ Token contract privileges – **BLACKLIST, SET ALLOWED TRANSFER, BUYBACK, SET TXN LIMIT, SET FEES, AIRDROP**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, exploitability, and audit disclaimer, kindly refer to the audit.

🚫 Contract addresses: **Not deployed**

🔗 Blockchain: **Not chained**

✅ Verify the authenticity of this report on InterFi's GitHub: <https://github.com/interfinetwork>



Table Of Contents

Audit Information

Audit Scope.....	5
------------------	---

Echelon Audit Standard

Audit Methodology	6
Risk Classification	8
Centralization Risk.....	9

Smart Contract Risk Assessment

Static Analysis.....	10
Software Analysis.....	16
Manual Analysis.....	21
SWC Attacks	28
Risk Status & Radar Chart.....	30

Audit Summary

Auditor's Verdict	31
-------------------------	----

Legal Advisory

Important Disclaimer	32
About InterFi Network.....	33



Audit Scope

InterFi was consulted by ShadowFi to conduct the smart contract security audit of their solidity source codes. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

- ❖ ShadowFiPreSale.sol
- ❖ ShadowFiLPVault.sol
- ❖ ShadowFiToken.sol

Audit Hash

Solidity source code is audited at hash #6fc952928a0689f0dcaa12c5c699a38ddb71c776

InterFi

.....

Smart Contract
Security Audit



Audit Methodology

The scope of this report is to audit smart contract sources code of Velas Domain. InterFi has scanned contracts and reviewed codes for common vulnerabilities, exploits, hacks, and backdoors. Due to being out of scope, InterFi has not tested contracts on testnet to assess any functional flaws. Smart contract audits are conducted using a set of standards and procedures. A mutual collaboration is essential to perform an effective smart contract audit. Here's a brief overview of auditing process and methodology:

Connect

- ❖ Onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

Audit

- ❖ Automated analysis is performed to identify common contract vulnerabilities. We may use following third party frameworks and dependencies to perform the automated analysis:
 - Remix IDE Developer Tool
 - Open Zeppelin Code Analyzer
 - SWC Vulnerabilities Registry
 - DEX Dependencies, e.g., Pancakeswap, Uniswap
- ❖ Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- ❖ Manual line by line analysis is performed to identify contract issues and centralized privileges.



Report

- ❖ Auditing team provides a preliminary report specifying all the checks which have been performed and findings thereof.
- ❖ Client's development team reviews the report, and makes amendments in solidity codes.
- ❖ Auditing team provides the final comprehensive report with open and unresolved issues.

Publish

- ❖ Client may use the audit report internally or disclose it publicly.
- ❖ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

Automated 3P frameworks used to assess the smart contract vulnerabilities

- ❖ Remix IDE Developer Tool
- ❖ Open Zeppelin Code Analyzer
- ❖ SWC Vulnerabilities Registry
- ❖ DEX Dependencies, e.g., Pancakeswap, Uniswap



Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the “vulnerability” flagged by a tool is in a function that requires owning the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! High	This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
! Medium	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
! Low	This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy related vulnerabilities should be fixed to deter exploits.
! Informational	This level vulnerabilities may be ignored. They are code style violations and deviations from standard practises.



Centralization Risk

Centralization risk is the most common cause of decentralized finance hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as:

- ❖ Contract owner can be granted the power to `pause()` or `lock()` the contract in case of an external attack.
- ❖ Contract owner can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale, and to list on an exchange.

Authorizing a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. **Most important question to ask here is, how to mitigate centralization risk?** Here's InterFi's recommendation to lower the risks related to centralization hacks:

- ❖ Smart contract owner's private key must be carefully secured to avoid any potential hack.
- ❖ Smart contract ownership should be shared by multi-signature (multi-sig) wallets.
- ❖ Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

ShadowFi's Centralization Status














- ❖ ShadowFi's smart contracts have **active ownership**.



Static Analysis

Symbol	Meaning
	Function can modify state
	Function is payable
	Function is locked
	Function can be accessed
!	Important functionality

```

| **ReentrancyGuard** | Implementation | |||
| L | <Constructor> | Public ! |  | NO ! |
| L | _nonReentrantBefore | Private  |  | |
| L | _nonReentrantAfter | Private  |  | |
| |||||
| **Context** | Implementation | |||
| L | _msgSender | Internal  | | |
| L | _msgData | Internal  | | |
| |||||
| **Ownable** | Implementation | Context |||
| L | <Constructor> | Public ! |  | NO ! |
| L | owner | Public ! | | NO ! |
| L | transferOwnership | Public ! |  | onlyOwner |
| L | _transferOwnership | Internal  |  | |
| |||||
| **IShadowFiToken** | Interface | |||
| L | totalSupply | External ! | | NO ! |
| L | balanceOf | External ! | | NO ! |
| L | transfer | External ! |  | NO ! |
| L | allowance | External ! | | NO ! |
| L | approve | External ! |  | NO ! |
| L | isAirdropped | External ! | | NO ! |
| L | transferFrom | External ! |  | NO ! |
| L | decimals | External ! | | NO ! |
| |||||
| **ShadowFiPresale** | Implementation | Ownable, ReentrancyGuard |||
| L | <Constructor> | Public ! |  | NO ! |
| L | depositTokens | Public ! |  | onlyOwner |
| L | withdrawTokens | Public ! |  | onlyOwner |
| L | setCost | Public ! |  | onlyOwner |

```



```

| L | setToken | Public ! | ● | onlyOwner |
| L | setDiscount | Public ! | ● | onlyOwner |
| L | setStartandStopTime | Public ! | ● | onlyOwner |
| L | setMax | Public ! | ● | onlyOwner |
| L | withdraw | Public ! | ● | onlyOwner |
| L | tokenAddress | Public ! | | NO ! |
| L | availableForSaleTokenAmount | Public ! | | NO ! |
| L | totalBoughtByUserTokenAmount | Public ! | | NO ! |
| L | totalSoldTokenAmount | Public ! | | NO ! |
| L | tokenCostBNB | Public ! | | NO ! |
| L | totalBNBRaisedSoFar | Public ! | | NO ! |
| L | discountPercentage | Public ! | | NO ! |
| L | maxBuyableTokenAmount | Public ! | | NO ! |
| L | getStartTime | Public ! | | NO ! |
| L | getStopTime | Public ! | | NO ! |
| L | buy | External ! | 🏠 | nonReentrant |

```

```

| **Context** | Implementation | |||
| L | _msgSender | Internal 🏠 | | |
| L | _msgData | Internal 🏠 | | |
| |||||

```

```

| **IPancakeRouter** | Interface | |||
| L | factory | External ! | | NO ! |
| L | WETH | External ! | | NO ! |
| L | addLiquidity | External ! | ● | NO ! |
| L | addLiquidityETH | External ! | 🏠 | NO ! |
| L | removeLiquidity | External ! | ● | NO ! |
| L | removeLiquidityETH | External ! | ● | NO ! |
| L | removeLiquidityWithPermit | External ! | ● | NO ! |
| L | removeLiquidityETHWithPermit | External ! | ● | NO ! |
| L | swapExactTokensForTokens | External ! | ● | NO ! |
| L | swapTokensForExactTokens | External ! | ● | NO ! |
| L | swapExactETHForTokens | External ! | 🏠 | NO ! |
| L | swapTokensForExactETH | External ! | ● | NO ! |
| L | swapExactTokensForETH | External ! | ● | NO ! |
| L | swapETHForExactTokens | External ! | 🏠 | NO ! |
| L | quote | External ! | | NO ! |
| L | getAmountOut | External ! | | NO ! |
| L | getAmountIn | External ! | | NO ! |
| L | getAmountsOut | External ! | | NO ! |
| L | getAmountsIn | External ! | | NO ! |
| L | removeLiquidityETHSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ! | 🏠 | NO ! |
| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External ! | ● | NO ! |

```

```

| |||||
| **IPancakeFactory** | Interface | |||
| L | getPair | External ! | | NO ! |
| |||||

```













```







| **IPancakePair** | Interface | |||
| L | name | External ! | |NO ! |
| L | symbol | External ! | |NO ! |
| L | decimals | External ! | |NO ! |
| L | totalSupply | External ! | |NO ! |
| L | balanceOf | External ! | |NO ! |
| L | allowance | External ! | |NO ! |
| L | approve | External ! | ● |NO ! |
| L | transfer | External ! | ● |NO ! |
| L | transferFrom | External ! | ● |NO ! |
| L | DOMAIN_SEPARATOR | External ! | |NO ! |
| L | PERMIT_TYPEHASH | External ! | |NO ! |
| L | nonces | External ! | |NO ! |
| L | permit | External ! | ● |NO ! |
| L | MINIMUM_LIQUIDITY | External ! | |NO ! |
| L | factory | External ! | |NO ! |
| L | token0 | External ! | |NO ! |
| L | token1 | External ! | |NO ! |
| L | getReserves | External ! | |NO ! |
| L | price0CumulativeLast | External ! | |NO ! |
| L | price1CumulativeLast | External ! | |NO ! |
| L | kLast | External ! | |NO ! |
| L | mint | External ! | ● |NO ! |
| L | burn | External ! | ● |NO ! |
| L | swap | External ! | ● |NO ! |
| L | skim | External ! | ● |NO ! |
| L | sync | External ! | ● |NO ! |
| L | initialize | External ! | ● |NO ! |
| |||||
| **IShadowFiToken** | Interface | |||
| L | totalSupply | External ! | |NO ! |
| L | balanceOf | External ! | |NO ! |
| L | transfer | External ! | ● |NO ! |
| L | allowance | External ! | |NO ! |
| L | approve | External ! | ● |NO ! |
| L | isAirdropped | External ! | |NO ! |
| L | transferFrom | External ! | ● |NO ! |
| L | decimals | External ! | |NO ! |
| L | burn | External ! | ● |NO ! |
| |||||
| **IERC20** | Interface | |||
| L | transfer | External ! | ● |NO ! |
| L | balanceOf | External ! | |NO ! |
| L | approve | External ! | ● |NO ! |
| |||||
| **IWETH** | Interface | |||
| L | deposit | External ! | 🏠 |NO ! |
| L | transfer | External ! | ● |NO ! |
| L | withdraw | External ! | ● |NO ! |
| |||||




```









```

| **ShadowFiLiquidityLock** | Implementation | Ownable, ReentrancyGuard |||
| L | <Constructor> | Public ! |  | NO ! |
| L | endLock | Public ! |  | onlyOwner |
| L | extendLockTime | Public ! |  | onlyOwner |
| L | withdrawBNB | Public ! |  | onlyOwner |
| L | withdrawTokens | Public ! |  | onlyOwner |
| L | shadowStrike | Public ! |  | onlyOwner |
| L | shadowBurst | Public ! |  | onlyOwner |
| L | getLPOwnershipPercent | Public ! | | NO ! |
| L | getLiquidTokens | Public ! | | NO ! |
| L | getLiquidPercent | Public ! | | NO ! |
| L | getRemoveAmountForBuyAndBurnExcess | Public ! | | NO ! |
| L | getRemoveAmountForBuyAndBurnExcessAmount | Public ! | | NO ! |
| L | addLiquidity | External ! |  | onlyOwner nonReentrant |
| L | getLockTime | Public ! | | NO ! |
| L | <Receive Ether> | External ! |  | NO ! |
| L | <Fallback> | External ! |  | NO ! |

| **SafeMath** | Library | |||
| L | add | Internal  | | |
| L | sub | Internal  | | |
| L | sub | Internal  | | |
| L | mul | Internal  | | |
| L | div | Internal  | | |
| L | div | Internal  | | |
| |||||

| **IBEP20** | Interface | |||
| L | totalSupply | External ! | | NO ! |
| L | decimals | External ! | | NO ! |
| L | symbol | External ! | | NO ! |
| L | name | External ! | | NO ! |
| L | getOwner | External ! | | NO ! |
| L | balanceOf | External ! | | NO ! |
| L | transfer | External ! |  | NO ! |
| L | allowance | External ! | | NO ! |
| L | approve | External ! |  | NO ! |
| L | transferFrom | External ! |  | NO ! |
| |||||

| **IDEXFactory** | Interface | |||
| L | createPair | External ! |  | NO ! |
| |||||

| **IDEXRouter** | Interface | |||
| L | factory | External ! | | NO ! |
| L | WETH | External ! | | NO ! |
| L | addLiquidity | External ! |  | NO ! |
| L | addLiquidityETH | External ! |  | NO ! |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! |  | NO ! |
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ! |  | NO ! |
| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External ! |  | NO ! |
| |||||

```



```

| **ShadowAuth** | Implementation | |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | authorizeFor | Public ! | ● | authorizedFor |
| L | authorizeForMultiplePermissions | Public ! | ● | authorizedFor |
| L | unauthorizeFor | Public ! | ● | authorizedFor |
| L | unauthorizeForMultiplePermissions | Public ! | ● | authorizedFor |
| L | isOwner | Public ! | | NO ! |
| L | isAuthorizedFor | Public ! | | NO ! |
| L | isAuthorizedFor | Public ! | | NO ! |
| L | transferOwnership | Public ! | ● | onlyOwner |
| L | getPermissionNameToIndex | Public ! | | NO ! |
| L | getPermissionUnlockTime | Public ! | | NO ! |
| L | isLocked | Public ! | | NO ! |
| L | lockPermission | Public ! | ● | authorizedFor |
| L | unlockPermission | Public ! | ● | NO ! |
| |||||
| **IDividendDistributor** | Interface | |||
| L | setDistributionCriteria | External ! | ● | NO ! |
| L | setShare | External ! | ● | NO ! |
| L | deposit | External ! | 🚫 | NO ! |
| L | process | External ! | ● | NO ! |
| L | claimDividend | External ! | ● | NO ! |
| |||||
| **DividendDistributor** | Implementation | IDividendDistributor |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | setDistributionCriteria | External ! | ● | onlyToken |
| L | setShare | External ! | ● | onlyToken |
| L | deposit | External ! | 🚫 | onlyToken |
| L | process | External ! | ● | onlyToken |
| L | shouldDistribute | Internal 🔒 | | |
| L | distributeDividend | Internal 🔒 | ● | |
| L | claimDividend | External ! | ● | NO ! |
| L | getUnpaidEarnings | Public ! | | NO ! |
| L | getCumulativeDividends | Internal 🔒 | | |
| L | addShareholder | Internal 🔒 | ● | |
| L | removeShareholder | Internal 🔒 | ● | |
| |||||
| **ShadowFi** | Implementation | IBEP20, ShadowAuth |||
| L | <Constructor> | Public ! | ● | ShadowAuth |
| L | <Receive Ether> | External ! | 🚫 | NO ! |
| L | totalSupply | External ! | | NO ! |
| L | decimals | External ! | | NO ! |
| L | symbol | External ! | | NO ! |
| L | name | External ! | | NO ! |
| L | getOwner | External ! | | NO ! |
| L | balanceOf | Public ! | | NO ! |
| L | allowance | External ! | | NO ! |
| L | approve | Public ! | ● | NO ! |
| L | approveMax | External ! | ● | NO ! |
| L | transfer | External ! | ● | NO ! |

```



```

| L | transferFrom | External ! | ● | NO ! |
| L | _transferFrom | Internal 🔒 | ● | |
| L | _basicTransfer | Internal 🔒 | ● | |
| L | checkTxLimit | Internal 🔒 | | |
| L | shouldTakeFee | Internal 🔒 | | |
| L | getTotalFee | Public ! | | NO ! |
| L | getMultipliedFee | Public ! | | NO ! |
| L | takeFee | Internal 🔒 | ● | |
| L | isSell | Internal 🔒 | | |
| L | shouldSwapBack | Internal 🔒 | | |
| L | swapBack | Internal 🔒 | ● | swapping |
| L | triggerBuyback | External ! | ● | authorizedFor |
| L | clearBuybackMultiplier | External ! | ● | authorizedFor |
| L | buyTokens | Internal 🔒 | ● | swapping |
| L | setBuybackMultiplierSettings | External ! | ● | authorizedFor |
| L | launched | Internal 🔒 | | |
| L | launch | Internal 🔒 | ● | |
| L | setTxLimit | External ! | ● | authorizedFor |
| L | setIsDividendExempt | External ! | ● | authorizedFor |
| L | setIsFeeExempt | External ! | ● | authorizedFor |
| L | setIsTxLimitExempt | External ! | ● | authorizedFor |
| L | setFees | External ! | ● | authorizedFor |
| L | setFeeReceivers | External ! | ● | authorizedFor |
| L | setSwapBackSettings | External ! | ● | authorizedFor |
| L | setTargetLiquidity | External ! | ● | authorizedFor |
| L | setDistributionCriteria | External ! | ● | authorizedFor |
| L | setDistributorSettings | External ! | ● | authorizedFor |
| L | getCirculatingSupply | Public ! | | NO ! |
| L | getLiquidityBacking | Public ! | | NO ! |
| L | isOverLiquified | Public ! | | NO ! |
| L | claimDividend | External ! | ● | NO ! |
| L | addPair | External ! | ● | authorizedFor |
| L | removeLastPair | External ! | ● | authorizedFor |
| L | setFeesOnNormalTransfers | External ! | ● | authorizedFor |
| L | setLaunchedAt | External ! | ● | authorizedFor |
| L | setAllowedAddress | External ! | ● | onlyOwner |
| L | burn | Public ! | ● | NO ! |
| L | airdrop | External ! | ● | onlyOwner |
| L | isAirdropped | External ! | | NO ! |
| L | setBlackListed | External ! | ● | onlyOwner |

```



Software Analysis

Function Signatures

```

18160ddd => totalSupply()
313ce567 => decimals()
95d89b41 => symbol()
06fdde03 => name()
893d20e8 => getOwner()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
c9c65396 => createPair(address,address)
c45a0155 => factory()
ad5c4648 => WETH()
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
1e8f5283 => authorizeFor(address,string)
af05cf93 => authorizeForMultiplePermissions(address,string[])
8101cae3 => unauthorizeFor(address,string)
1ad6084f => unauthorizeForMultiplePermissions(address,string[])
2f54bf6e => isOwner(address)
c41235a6 => isAuthorizedFor(address,string)
7a511001 => isAuthorizedFor(address,Permission)
f2fde38b => transferOwnership(address)
06fbdc8 => getPermissionNameToIndex(string)
e75cae79 => getPermissionUnlockTime(string)
8032eccb => isLocked(string)
971563fc => lockPermission(string,uint64)
07858b02 => unlockPermission(string)
2d48e896 => setDistributionCriteria(uint256,uint256)
14b6ca96 => setShare(address,uint256)
d0e30db0 => deposit()
ffb2c479 => process(uint256)
f0fc6bca => claimDividend()
8c21cd52 => shouldDistribute(address)
5319504a => distributeDividend(address)
28fd3198 => getUnpaidEarnings(address)
e68af3ac => getCumulativeDividends(uint256)
db29fe12 => addShareholder(address)
9babdad6 => removeShareholder(address)
571ac8b0 => approveMax(address)

```




```

cb712535 => _transferFrom(address,address,uint256)
f0774e71 => _basicTransfer(address,address,uint256)
4afa518a => checkTxLimit(address,uint256)
332402f8 => shouldTakeFee(address,address)
f1f3bca3 => getTotalFee(bool)
d806d12f => getMultipliedFee()
20cb7bce => takeFee(address,address,uint256)
bff09df1 => isSell(address)
0d5c6cea => shouldSwapBack()
6ac5eeee => swapBack()
6a6e3cbe => triggerBuyback(uint256,bool)
b210b06d => clearBuybackMultiplier()
c625e9b1 => buyTokens(uint256,address)
2375ce40 => setBuybackMultiplierSettings(uint256,uint256,uint256)
8091f3bf => launched()
01339c21 => launch()
5c85974f => setTxLimit(uint256)
f708a64f => setIsDividendExempt(address,bool)
658d4b7f => setIsFeeExempt(address,bool)
f84ba65d => setIsTxLimitExempt(address,bool)
86f6c3c1 => setFees(uint256,uint256,uint256,uint256,uint256,uint256)
a4b45c00 => setFeeReceivers(address,address)
df20fd49 => setSwapBackSettings(bool,uint256)
201e7991 => setTargetLiquidity(uint256,uint256)
9d1944f5 => setDistributorSettings(uint256)
2b112e49 => getCirculatingSupply()
d51ed1c8 => getLiquidityBacking(uint256)
1161ae39 => isOverLiquified(uint256,uint256)
c2b7bbb6 => addPair(address)
0093dc14 => removeLastPair()
f3a54f2c => setFeesOnNormalTransfers(bool)
39e67c8a => setLaunchedAt(uint256)
fae36986 => setAllowedAddress(address,bool)
42966c68 => burn(uint256)
8ba4cc3c => airdrop(address,uint256)
bfa51df9 => isAirdropped(address)
5cd8c072 => setBlackListed(address,bool)
62898eb8 => _nonReentrantBefore()
c7443eb2 => _nonReentrantAfter()
119df25f => _msgSender()
8b49d47e => _msgData()
8da5cb5b => owner()
f2fde38b => transferOwnership(address)
d29d44ee => _transferOwnership(address)
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
20b57614 => isAirdropped(address)

```



```

23b872dd => transferFrom(address,address,uint256)
313ce567 => decimals()
66168bd7 => depositTokens(address,uint256)
06b091f9 => withdrawTokens(address,uint256)
44a0d68a => setCost(uint256)
144fa6d7 => setToken(address)
dabd2719 => setDiscount(uint256)
0924068e => setStartandStopTime(uint32,uint32)
1fe9eabc => setMax(uint256)
3ccfd60b => withdraw()
9d76ea58 => tokenAddress()
887d0a12 => availableForSaleTokenAmount()
0dda4650 => totalBoughtByUserTokenAmount(address)
962f05e4 => totalSoldTokenAmount()
704261e1 => tokenCostBNB()
9a46e66f => totalBNBRaisedSoFar()
b3fa9f24 => discountPercentage()
54761b8d => maxBuyableTokenAmount()
c828371e => getStartTime()
4af899a2 => getStopTime()
d96a094a => buy(uint256)
c45a0155 => factory()
ad5c4648 => WETH()
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde => removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
ded9382a =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
38ed1739 => swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee => swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5 => swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a => swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5 => swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41 => swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
af2979eb =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)

```



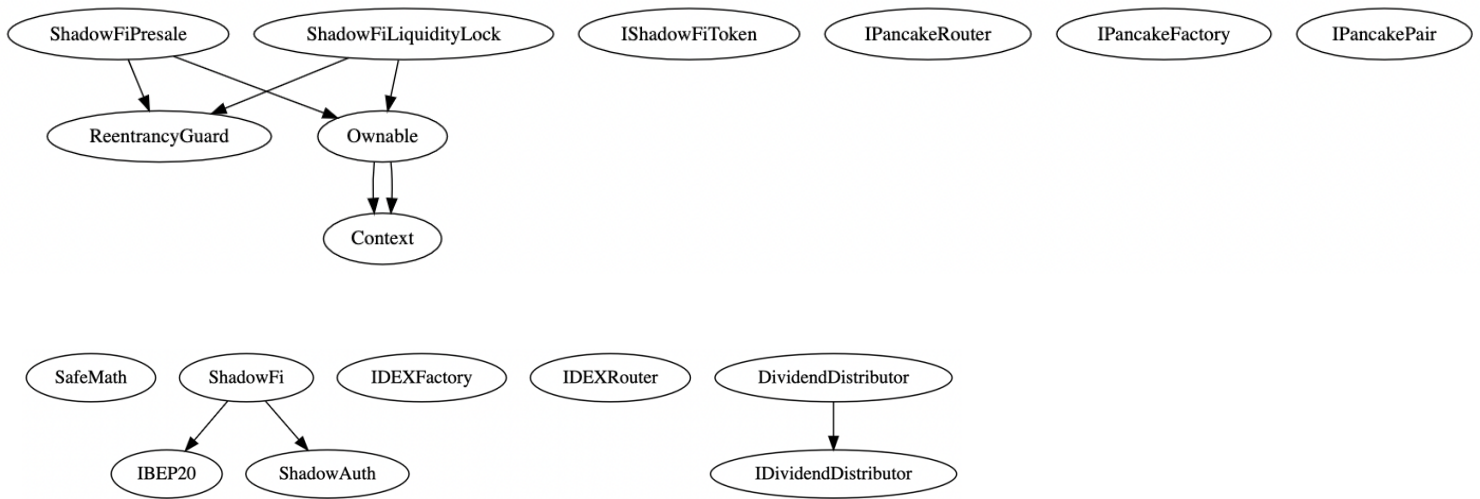
```

5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
e6a43905 => getPair(address,address)
06fdde03 => name()
95d89b41 => symbol()
3644e515 => DOMAIN_SEPARATOR()
30adf81f => PERMIT_TYPEHASH()
7ecebe00 => nonces(address)
d505accf => permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56 => MINIMUM_LIQUIDITY()
0dfe1681 => token0()
d21220a7 => token1()
0902f1ac => getReserves()
5909c0d5 => price0CumulativeLast()
5a3d5493 => price1CumulativeLast()
7464fc3d => kLast()
6a627842 => mint(address)
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address)
fff6cae9 => sync()
485cc955 => initialize(address,address)
9dc29fac => burn(address,uint256)
d0e30db0 => deposit()
2e1a7d4d => withdraw(uint256)
077e6334 => endLock()
828047a5 => extendLockTime(uint256)
1d111d13 => withdrawBNB()
49df728c => withdrawTokens(address)
adb4fa84 => shadowStrike()
5651d706 => shadowBurst(uint256)
235d1019 => getLPOwnershipPercent()
72c5eaec => getLiquidTokens()
e3443876 => getLiquidPercent()
e528777c => getRemoveAmountForBuyAndBurnExcess()
0babcd7 => getRemoveAmountForBuyAndBurnExcessAmount(uint256)
51c6590a => addLiquidity(uint256)
c0a4d64d => getLockTime()

```



Inheritance Graph



InterFi

Smart Contract Security Audit



Manual Analysis

Function	Description	Available	Status
Total Supply	provides information about the total token supply	Yes	Passed
Balance Of	provides account balance of the owner's account	Yes	Passed
Transfer	executes transfers of a specified number of tokens to a specified address	Yes	Passed
Approve	allow a spender to withdraw a set number of tokens from a specified account	Yes	Passed
Allowance	returns a set number of tokens from a spender to the owner	Yes	Passed
Buy Back	is an action in which the project buys back its tokens from the existing holders usually at a market price	Yes	Passed
Burn	executes transfers of a specified number of tokens to a burn address	Yes	Passed
Mint	executes the creation of a specified number of tokens and adds it to the total supply	NA	NA
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	NA	NA
Dividend	executes transfers of a specified dividend token to a specified address	Yes	Passed
Lock	locks permitted access to all or some function modules of the smart contract	Yes	Passed



Function	Description	Tested	Verdict
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	! Low
Airdrop	executes transfers of a specified number of tokens to a specified address	Yes	Passed
Max Transaction	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	Passed
Contract Fees	executes fee collection from swap events and/or transfer events	Yes	Passed
Transfer Ownership	executes transfer of contract ownership to a specified wallet	Yes	Passed
Renounce Ownership	executes transfer of contract ownership to a dead address	Yes	Passed

Smart Contract Security Audit



Notable Information

- ❖ Token contract **authorizes** wallets to modify some contract privileged functions. When ownership is transferred, previous authorizations should be voided. There's an elevated risk of out-of-gas, and potential resource exhaustion errors with multi wallet calls.

```

constructor(address owner_) {
    owner = owner_;
    for (uint256 i; i < NUM_PERMISSIONS; i++) {
        authorizations[owner_][i] = true;

    function authorizeFor(address adr, string memory permissionName) public
    authorizedFor(Permission.Authorize) {
        uint256 permIndex = permissionNameToIndex[permissionName];
        authorizations[adr][permIndex] = true;

    function authorizeForMultiplePermissions(address adr, string[] calldata permissionNames) public
    authorizedFor(Permission.Authorize) {
        for (uint256 i; i < permissionNames.length; i++) {
            uint256 permIndex = permissionNameToIndex[permissionNames[i]];
            authorizations[adr][permIndex] = true;

    function transferOwnership(address payable adr) public onlyOwner {
        address oldOwner = owner;
        owner = adr;
        for (uint256 i; i < NUM_PERMISSIONS; i++) {
            authorizations[oldOwner][i] = false;
            authorizations[owner][i] = true;

```

- ❖ Token contract authorized role can change **buyback multiplier settings**.

```

function triggerBuyback(uint256 amount, bool triggerBuybackMultiplier) external
    authorizedFor(Permission.Buyback) {
        // buyTokens(amount, DEAD);
        burn(msg.sender, amount);
        if(triggerBuybackMultiplier){
            buybackMultiplierTriggeredAt = block.timestamp;

```

- ❖ In presale contract, function **buy()** is protected against re-entrancy.
- ❖ In liquidity lock contract, function **addLiquidity()** is protected against re-entrancy.



- ❖ Presale contract owner can **set token to sell, set sell cost, and set sell discount.**

```
function setCost(uint256 _cost) public onlyOwner {
    tokenCost = _cost;
}
function setToken(address _tokenAddress) public onlyOwner {
    require(address(token) != _tokenAddress, "Already set!");
    token = IShadowFiToken(_tokenAddress);
}
function setDiscount(uint256 _discountPercent) public onlyOwner {
    require(
        _discountPercent > 1 && _discountPercent < 1001,
        "Invalid percent is provided."
    );
}
```

- ❖ Presale contract owner can **set max token buyable amount.**

```
function setMax(uint256 _maxAmount) public onlyOwner {
    maxAmount = _maxAmount;
}
```

- ❖ Liquidity lock contract owner can **add liquidity.**

```
function addLiquidity(uint256 _amountToken) external payable onlyOwner {
    require(!lockEnded, "Contract is locked.");
    require(_amountToken > 0, "Invalid parameter is provided.");
    require(msg.value > 0, "You should fund this contract with BNB.");
}
```

- ❖ Liquidity lock contract owner can **end lock** once the unlock time is reached.

```
function endLock() public onlyOwner {
    require(!lockEnded, "You already claimed all LP tokens.");
    require(block.timestamp >= lockTime, "LP tokens are still locked.");
}
```

- ❖ Liquidity lock contract utilizes **shadowStrike()**, and **shadowBurst()**. Amount of ShadowFi tokens in liquidity is at least 8% of the total supply.

```
function shadowStrike() public onlyOwner {
    require(liquidPercent > 800, "The amount of ShadowFi tokens in liquidity should be 8%+ of the totalSupply.");
}
function shadowBurst(uint256 percent) public onlyOwner {
    require(percent >= 1 && percent <= 9200, "Invalid parameter is provided");
    require(liquidPercent - percent >= 800, "The amount compared to liquidity tokens should be less than 8% of the totalSupply.");
}
```



- ❖ Token Smart contract utilizes **safemath** function to avoid common smart contract vulnerabilities.

```
string private _name = "ShadowFiToken";
library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");
    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
```

- ❖ Smart contracts call `receive()` for fallbacks. It is executed on a call to the contract with empty call data. Make sure the contract can receive token through a regular transaction, and does not throw an exception.

```
.....
receive() external payable {}
fallback() external payable {}
```

- ❖ Token contract owner can **airdrop** tokens to specified wallets.

```
function airdrop(address _user, uint256 _amount) external onlyOwner {
    _transferFrom(msg.sender, _user, _amount);
    airdropped[_user] = true;
```

- ❖ Token contract owner can **blacklist** certain wallets from interacting with the contract function modules.

```
function setBlackListed(address user, bool flag) external onlyOwner {
    blacklist[user] = flag;
```



- ❖ Token contract authorized role can **change transaction fees**. Maximum fee limits are set to allow the value change within the set parameters.

```
function setFees(uint256 _liquidityFee, uint256 _buybackFee, uint256 _reflectionFee, uint256
_marketingFee, uint256 _feeDenominator, uint256 _totalSellFee) external
authorizedFor(Permission.AdjustContractVariables) {
    totalBuyFee = _liquidityFee.add(_buybackFee).add(_reflectionFee).add(_marketingFee);
    require(totalBuyFee <= feeDenominator / 10, "Buy fee too high");
    require(totalSellFee <= feeDenominator / 5, "Sell fee too high");

    uint256 liquidityFee = 200;
    uint256 buybackFee = 0;
    uint256 reflectionFee = 600;
    uint256 marketingFee = 100;
    uint256 totalBuyFee = 900;
    uint256 totalSellFee = 1400;
    uint256 feeDenominator = 10000;
```

- ❖ Token contract authorized role can **change max transaction limit**. Minimum limit is set to allow the value change within the set parameters.

```
function setTxLimit(uint256 amount) external authorizedFor(Permission.AdjustContractVariables)
    require(amount >= _maxSupply / 2000);
```

- ❖ Token contract function **addLiquidity()** should send auto liquidity to an inaccessible address. Token contract sends LP tokens to autoLiquidityReceiver.

```
if(amountToLiquify > 0){
    try router.addLiquidityETH{ value: amountBNBLiquidity }(
        address(this),
        amountToLiquify,
        0,
        0,
        autoLiquidityReceiver,
        block.timestamp

        autoLiquidityReceiver = owner_;
```

- ❖ Token contract has an **informational severity issue** which may or may not create any functional vulnerability.

"Irrelevant Code"



- ❖ Token contract has an **informational severity issue** which may or may not create any functional vulnerability.

“Unknown Externally Owned Accounts”

- ❖ Smart contracts have **informational severity issue** which may or may not create any functional vulnerability.

“Expected pragma, import directive or contract/interface/library definition”

- ❖ Smart contracts have **low severity issue** which may or may not create any functional vulnerability.

“Utilization of block.timestamp”

InterFi

.....

Smart Contract Security Audit



SWC Attacks

SWC ID	Description	Status
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Informational
SWC-103	Floating Pragma	! Informational
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	! Low
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	! Low
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

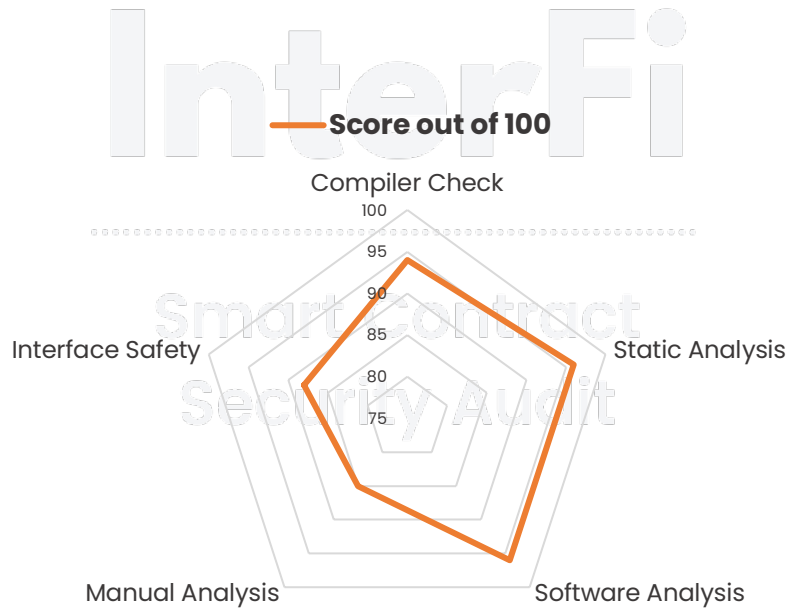


SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with the hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	! Informational
SWC-136	Unencrypted Private Data On-Chain	Passed



Risk Status & Radar Chart

Risk Severity	Status
High	No high severity issues identified
Medium	No medium severity issues identified
Low	3 low severity issues identified
Informational	3 informational severity issues identified
Centralization Risk	Active contract ownership identified



Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ ShadowFi's solidity source codes have **LOW RISK SEVERITY**
- ❖ ShadowFi's smart contracts have **ACTIVE OWNERSHIP**
- ❖ ShadowFi's centralization risk correlated to the active owner is **HIGH**

InterFi

.....

Note for stakeholders

Smart Contract Security Audit

- ❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.
- ❖ If the smart contract is not deployed on any blockchain at the time of the audit, the contract can be modified or altered before blockchain development. Verify contract's deployment status in the audit report.
- ❖ Make sure that the project team's KYC/identity is verified by an independent firm.
- ❖ Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
- ❖ Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period.



Important Disclaimer

InterFi Network provides contract development, testing, auditing and project evaluation services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purpose without InterFi's prior written consent.**

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only – we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant to external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security. Therefore, InterFi does not guarantee the explicit security of the audited smart contract.**

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.

This report should not be considered as an endorsement or disapproval of any project or team.

The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your due diligence and consult your financial advisor before making any investment decisions.



About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy to use.**

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit <https://interfi.network>

To view our audit portfolio, visit <https://github.com/interfinetwork>

To book an audit, message <https://t.me/interfiaudits>





RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦