Smart Contract Security Audit V1

xfuzion Zap Smart Contract Audit

https://xfuzion.finance/

Nov 21, 2023

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Pulse Chain
- **Name**: Zap
- **Language** : solidity
- **Contract Address**: 0x627ee38bfFE94692A11Bdd0B036c02b7fc2acd45
- **Code Source**: https://scan.pulsechain.com/address/0x627ee38bfFE94692A11Bdd0B036c02b7fc2acd45/contracts#address-tabs
- **Website**: https://xfuzion.finance/
- **Telegram**: https://t.me/XfuzionFinance
- **X**: https://x.com/xfuzionfinance?s=21&t=UppJadbfgCRdnVpOnFAakQ
- **Youtube:** https://youtube.com/@XfuzionFinance?si=nfNwrOVxVWNJgnq2
- **Discord**: https://discord.gg/FCcD8YTuxc

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well-Secured**.

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 1 low, 0 very low-level issues and 1 note in all solidity files of the contract
The files:

Zap.sol

Audit Score:

100% secure

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Zap.sol	d37415e89d9e7d1b21ef8bd8d8e2ef2fd1541cdd	0x627ee38bfFE94692A11Bdd0B036c02b7fc2acd45

- Contract: Zap
- Inherit: Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
estimateZapIn	✓	Read / public	**Passed**
estimateZapInToken	✓	Read / public	**Passed**
useNativeRouter	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
setTokenBridgeForRouter	✓	Write / public	**Passed**
setUseNativeRouter	✓	Write / public	**Passed**
swapToken	✓	Write / public	**Passed**
swapToNative	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
zapAcross	✓	Write / public	**Passed**
zapInToken	✓	Write / public	**Passed**
zapOut	✓	Write / public	**Passed**
zapIn	✓	Write / payable	**Passed**

transferOwnership	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**

Issues Checking Status

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) for more info check https://swcregistry.io/

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

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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found.

Low:

Use of block.timestamp for comparisons

The value of block.timestamp can be manipulated by the miner. And conditions with strict equality is difficult to achieve - block.timestamp.

```
function zapAcross(address _from, uint amount, address _toRouter, address
_recipient) external {
        IERC20(_from).safeTransferFrom(msg.sender, address(this), amount);

        IUniswapV2Pair pair = IUniswapV2Pair(_from);
        _approveTokenIfNeeded(pair.token0(), _toRouter);
        _approveTokenIfNeeded(pair.token1(), _toRouter);

        IERC20(_from).safeTransfer(_from, amount);
        uint amt0;
        uint amt1;
        (amt0, amt1) = pair.burn(address(this));
        IUniswapV2Router01(_toRouter).addLiquidity(pair.token0(), pair.token1(),
amt0, amt1, 0, 0, _recipient, block.timestamp);
    }
```

Recommendation

Avoid use of block.timestamp.

Status

Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Solidity compiler Bugs

Description

The smart contract uses 0.8.4 which isn't the latest one every upgrade solves low security issues; you can check the bugs from this link

https://etherscan.io/solcbuginfo

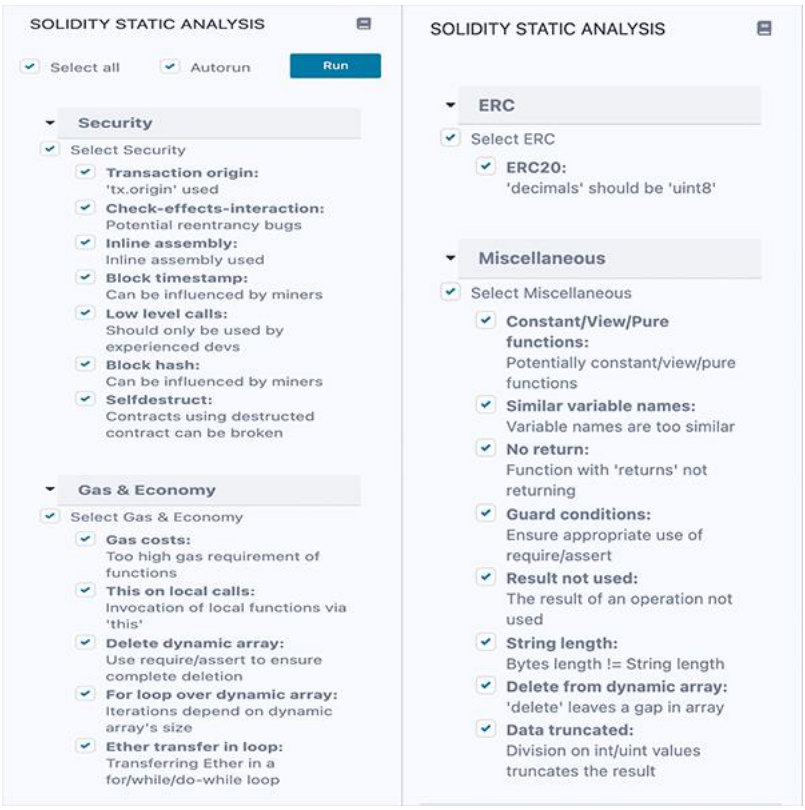
Remediation

Use 0.8.23 instead of 0.8.4

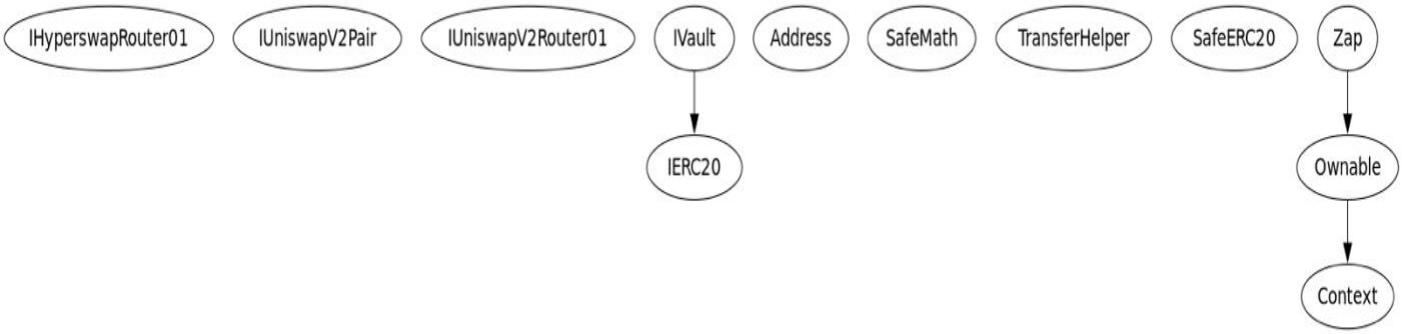
Status: Acknowledged.

Automatic Testing

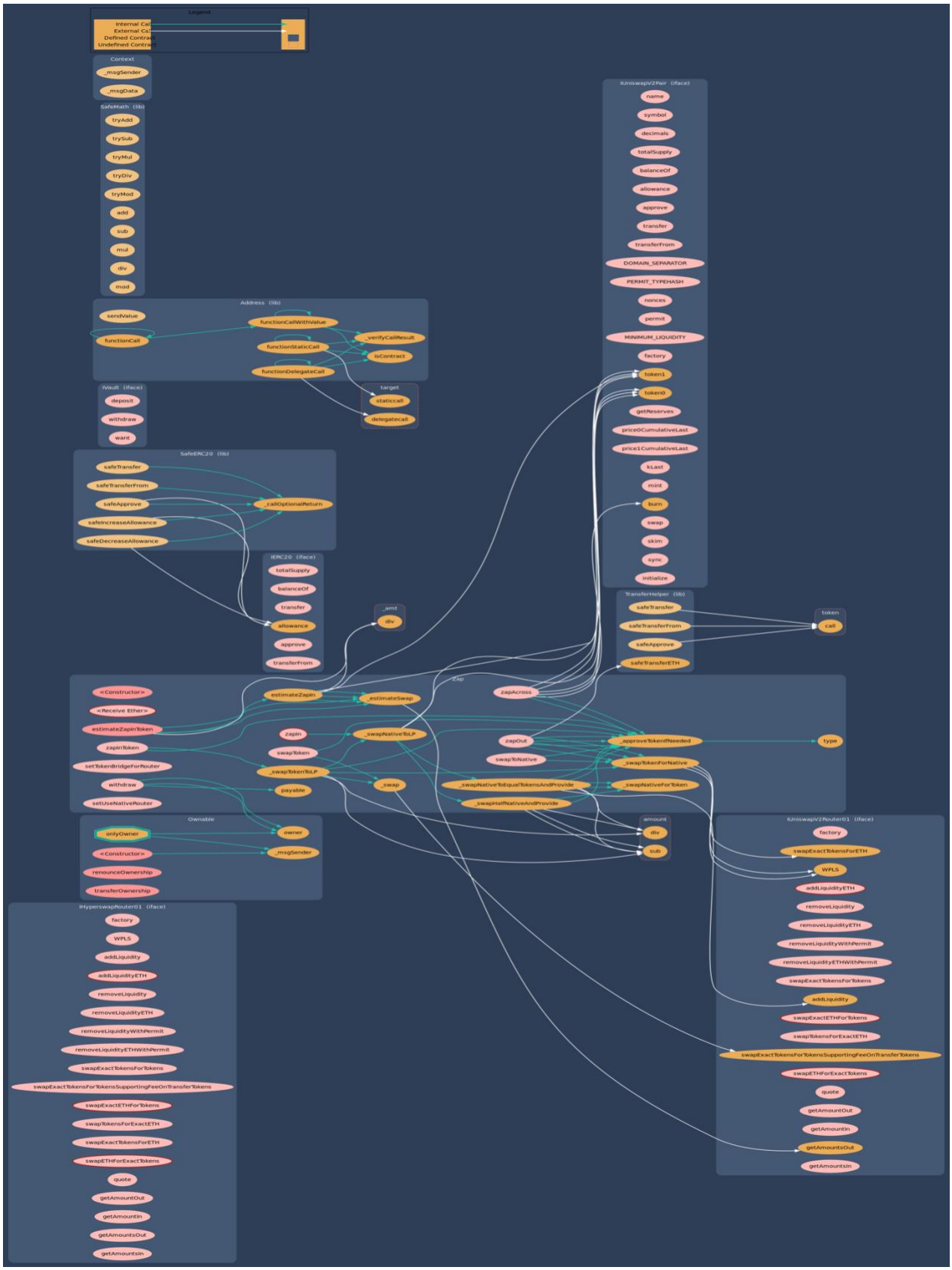
1- SOLIDITY STATIC ANALYSIS



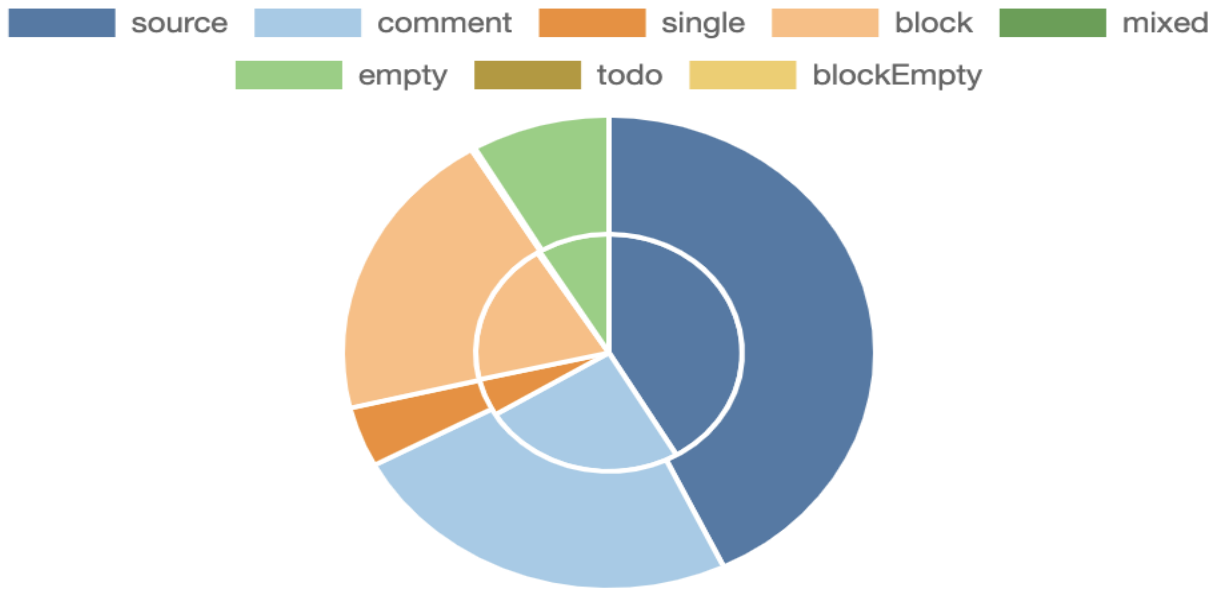
2- Inheritance graph



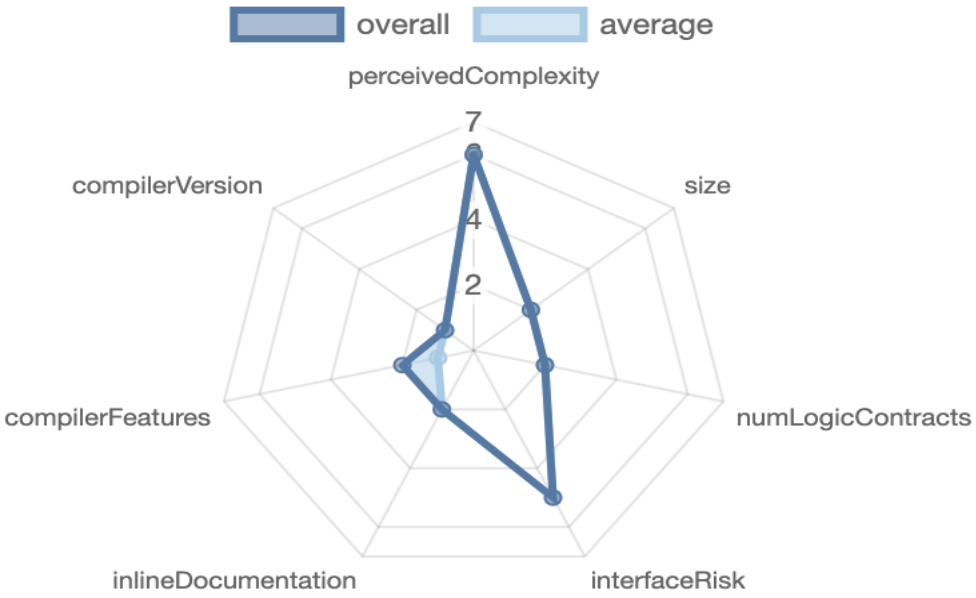
3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Context
_msgSender
_msgData
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
Address (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
_verifyCallResult
isContract
IVault (iface)
deposit
withdraw
want
target
staticcall
delegatecall
SafeERC20 (lib)
safeTransfer
safeTransferFrom
safeApprove
safeIncreaseAllowance
safeDecreaseAllowance
_callOptionalReturn
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
_amt
div
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token1
token0
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize
TransferHelper (lib)
safeTransfer
safeTransferFrom
safeApprove
safeTransferETH
token
call
Zap
<Constructor>
<Receive Ether>
estimateZapIn
estimateZapInToken
zapInToken
setTokenBridgeForRouter
withdraw
setUseNativeRouter
_estimateSwap
zapIn
swapToken
_swapTokenToLP
payable
_swapNativeToLP
_swap
zapAcross
zapOut
swapToNative
_swapNativeToEqualTokensAndProvide
_swapHalfNativeAndProvide
_approveTokenIfNeeded
_swapTokenForNative
_swapNativeForToken
type
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
owner
_msgSender
amount
div
sub
IHyperswapRouter01 (iface)
factory
WPLS
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IUniswapV2Router01 (iface)
factory
swapExactTokensForETH
WPLS
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
addLiquidity
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn

Source lines



Risk level



Source units in scope

Source Units in Scope

Source Units Analyzed: **1**
Source Units in Scope: **1** (**100%**)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	xfuzion Zap.sol	7	5	1318	994	553	424	742	💻💰📩👥🔆Σ
📝📚🔍🎨	**Totals**	**7**	**5**	**1318**	**994**	**553**	**424**	**742**	💻💰📩👥🔆Σ

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
1	4	5	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
89	8

External	Internal	Private	Pure	View
84	104	11	30	28

StateVariables

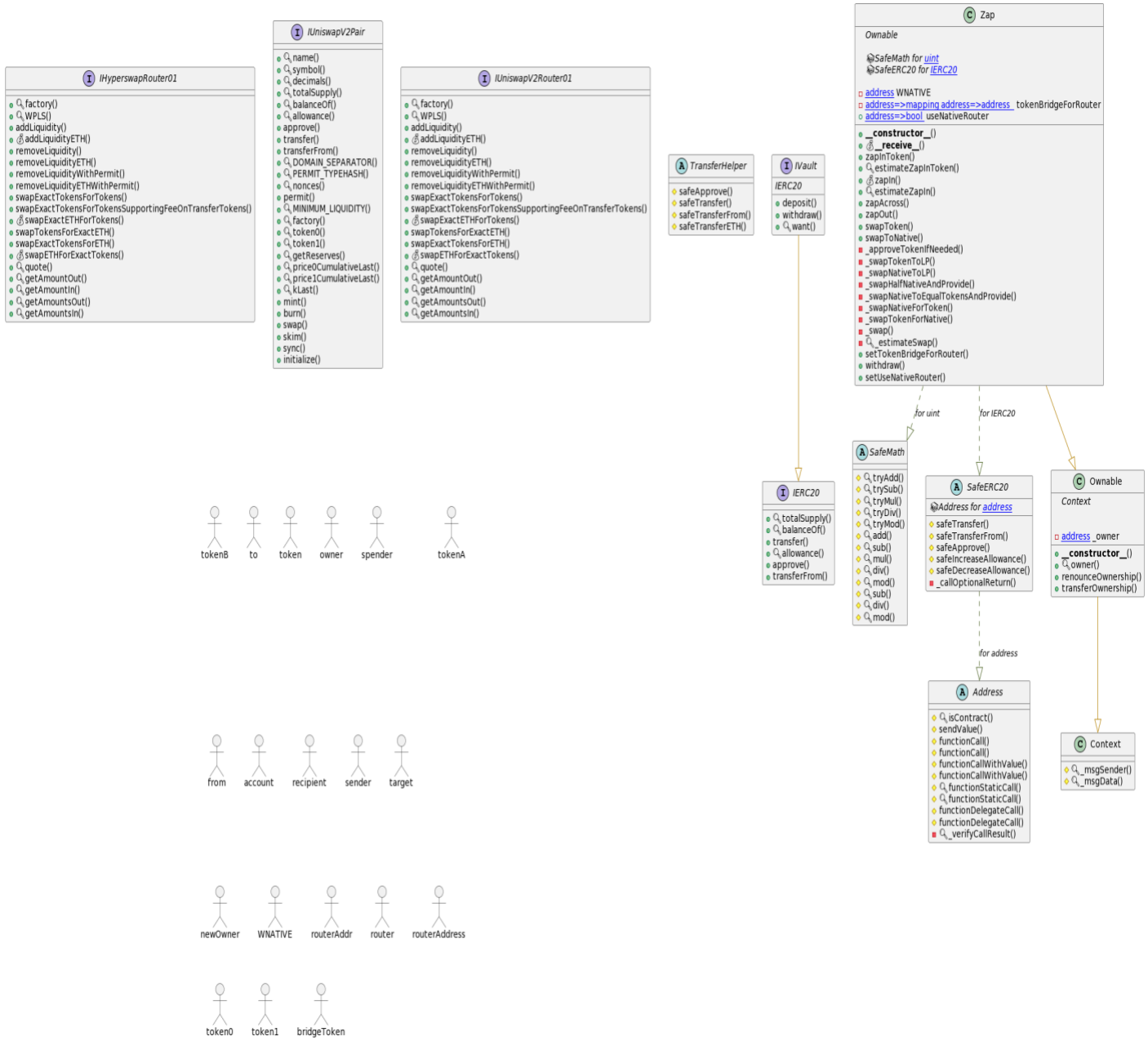
Total	🌐Public
4	1

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.8.4` `^0.8.0`		`yes`	`yes` (2 asm blocks)	

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🧼 ECRecover	🌀 New/Create/Create2
`yes`		`yes`			

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
factory	c45a0155	factory()
WPLS	ef8ef56f	WPLS()
addLiquidity	e8e33700	addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
addLiquidityETH	f305d719	addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity	baa2abde	removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH	02751cec	removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit	2195995c	removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit	ded9382a	removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokens	38ed1739	swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapExactTokensForTokensSupportingFeeOnTransferTokens	5c11d795	swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens	7ff36ab5	swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH	4a25d94a	swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH	18cbafe5	swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens	fb3bdb41	swapETHForExactTokens(uint256,address[],address,uint256)
quote	ad615dec	quote(uint256,uint256,uint256)
getAmountOut	054d50d4	getAmountOut(uint256,uint256,uint256)
getAmountIn	85f8c259	getAmountIn(uint256,uint256,uint256)
getAmountsOut	d06ca61f	getAmountsOut(uint256,address[])
getAmountsIn	1f00ca74	getAmountsIn(uint256,address[])
name	06fdde03	name()
symbol	95d89b41	symbol()
decimals	313ce567	decimals()
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transfer	a9059cbb	transfer(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
DOMAIN_SEPARATOR	3644e515	DOMAIN_SEPARATOR()
PERMIT_TYPEHASH	30adf81f	PERMIT_TYPEHASH()
nonces	7ecebe00	nonces(address)
permit	d505accf	permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
MINIMUM_LIQUIDITY	ba9a7a56	MINIMUM_LIQUIDITY()
factory	c45a0155	factory()
token0	0dfe1681	token0()
token1	d21220a7	token1()
getReserves	0902f1ac	getReserves()
price0CumulativeLast	5909c0d5	price0CumulativeLast()

price1CumulativeLast	5a3d5493	price1CumulativeLast()
kLast	7464fc3d	kLast()
mint	6a627842	mint(address)
burn	89afcb44	burn(address)
swap	022c0d9f	swap(uint256,uint256,address,bytes)
skim	bc25cf77	skim(address)
sync	fff6cae9	sync()
initialize	c4d66de8	initialize(address)
factory	c45a0155	factory()
WPLS	ef8ef56f	WPLS()
addLiquidity	e8e33700
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
addLiquidityETH	f305d719
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity	baa2abde
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH	02751cec
removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit	2195995c
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,b
ool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit	ded9382a
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,u
int8,bytes32,bytes32)
swapExactTokensForTokens	38ed1739
swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapExactTokensForTokensSupportingFeeOnTransferTokens	5c11d795
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
swapExactETHForTokens	7ff36ab5
swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH	4a25d94a
swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH	18cbafe5
swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens	fb3bdb41
swapETHForExactTokens(uint256,address[],address,uint256)
quote	ad615dec	quote(uint256,uint256,uint256)
getAmountOut	054d50d4	getAmountOut(uint256,uint256,uint256)
getAmountIn	85f8c259	getAmountIn(uint256,uint256,uint256)
getAmountsOut	d06ca61f	getAmountsOut(uint256,address[])
getAmountsIn	1f00ca74	getAmountsIn(uint256,address[])
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
transfer	a9059cbb	transfer(address,uint256)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
deposit	b6b55f25	deposit(uint256)
withdraw	2e1a7d4d	withdraw(uint256)
want	1f1fcd51	want()
owner	8da5cb5b	owner()
renounceOwnership	715018a6	renounceOwnership()
transferOwnership	f2fde38b	transferOwnership(address)
zapInToken	cee6202c	zapInToken(address,uint256,address,address,address)
estimateZapInToken	8e8d8152
estimateZapInToken(address,address,address,uint256)
zapIn	35e8b7d5	zapIn(address,address,address)
estimateZapIn	d9059f1b	estimateZapIn(address,address,uint256)

zapAcross	88f61f9f	zapAcross(address,uint256,address,address)
zapOut	5cc7647c	zapOut(address,uint256,address,address)
swapToken	ba2ebf96	swapToken(address,uint256,address,address,address)
swapToNative	89fecbd5	swapToNative(address,uint256,address,address)
setTokenBridgeForRouter	5424888b	setTokenBridgeForRouter(address,address,address)
withdraw	51cff8d9	withdraw(address)
setUseNativeRouter	88fbe419	setUseNativeRouter(address)

Automatic general report

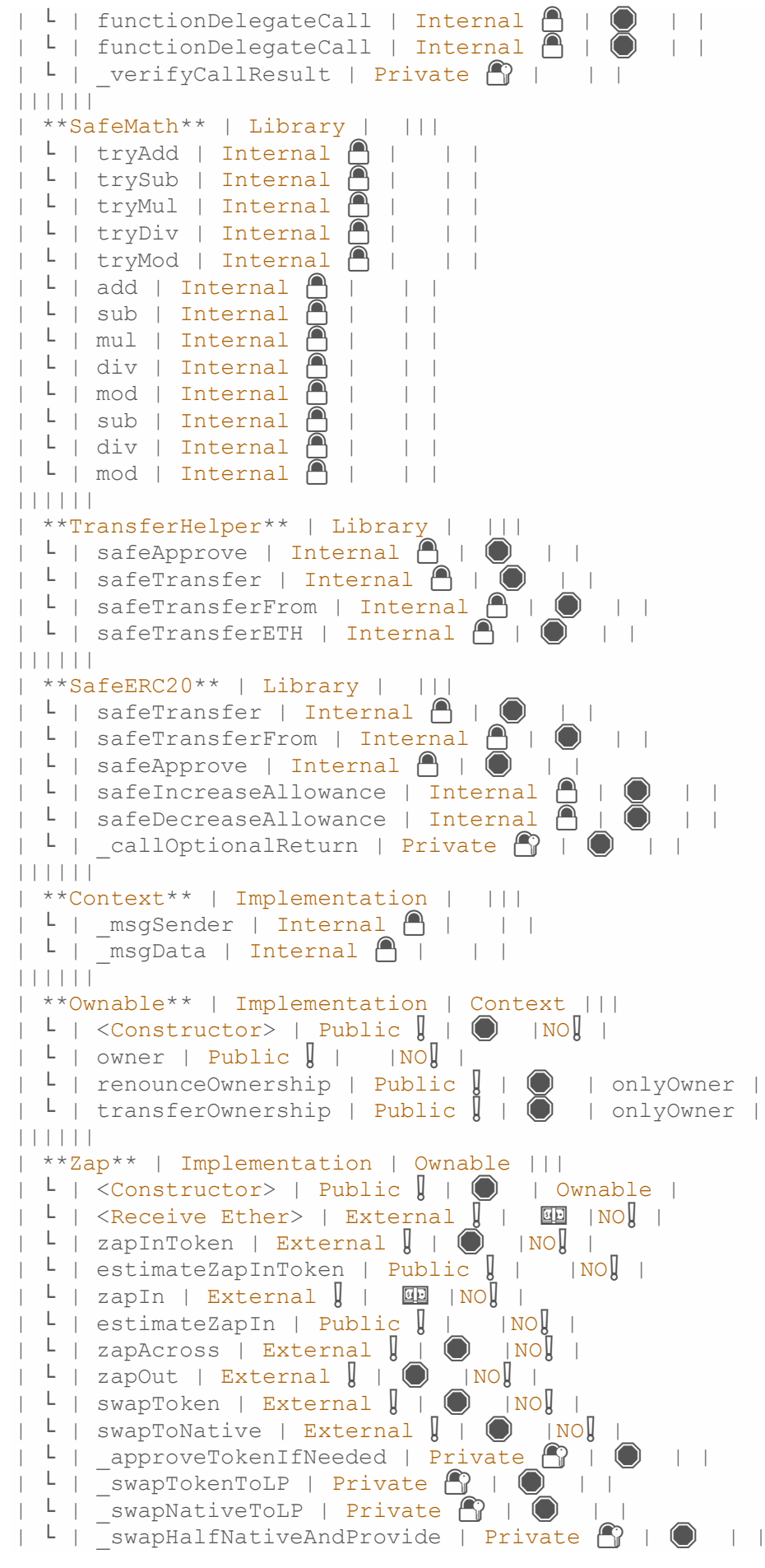
Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/xfuzion Zap.sol	d37415e89d9e7d1b21ef8bd8d8e2ef2fd1541cdd

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IHyperswapRouter01	Interface			
└	factory	External	NO	
└	WPLS	External	NO	
└	addLiquidity	External	NO	
└	addLiquidityETH	External	NO	
└	removeLiquidity	External	NO	
└	removeLiquidityETH	External	NO	
└	removeLiquidityWithPermit	External	NO	
└	removeLiquidityETHWithPermit	External	NO	
└	swapExactTokensForTokens	External	NO	
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	NO	
└	swapExactETHForTokens	External	NO	
└	swapTokensForExactETH	External	NO	
└	swapExactTokensForETH	External	NO	
└	swapETHForExactTokens	External	NO	
└	quote	External	NO	
└	getAmountOut	External	NO	
└	getAmountIn	External	NO	
└	getAmountsOut	External	NO	
└	getAmountsIn	External	NO	
IUniswapV2Pair	Interface			
└	name	External	NO	
└	symbol	External	NO	
└	decimals	External	NO	
└	totalSupply	External	NO	
└	balanceOf	External	NO	
└	allowance	External	NO	
└	approve	External	NO	
└	transfer	External	NO	
└	transferFrom	External	NO	
└	DOMAIN_SEPARATOR	External	NO	
└	PERMIT_TYPEHASH	External	NO	
└	nonces	External	NO	
└	permit	External	NO	
└	MINIMUM_LIQUIDITY	External	NO	


└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WPLS	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IVault	Interface	IERC20		
└	deposit	External ❗	🛑	NO❗
└	withdraw	External ❗	🛑	NO❗
└	want	External ❗		NO❗
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		


└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	_verifyCallResult	Private 🔐		
SafeMath	Library		
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
TransferHelper	Library		
└	safeApprove	Internal 🔒	🛑	
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeTransferETH	Internal 🔒	🛑	
SafeERC20	Library		
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeApprove	Internal 🔒	🛑	
└	safeIncreaseAllowance	Internal 🔒	🛑	
└	safeDecreaseAllowance	Internal 🔒	🛑	
└	_callOptionalReturn	Private 🔐	🛑	
Context	Implementation		
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context	
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Zap	Implementation	Ownable	
└	<Constructor>	Public ❗	🛑	Ownable
└	<Receive Ether>	External ❗	💸	NO❗
└	zapInToken	External ❗	🛑	NO❗
└	estimateZapInToken	Public ❗		NO❗
└	zapIn	External ❗	💸	NO❗
└	estimateZapIn	Public ❗		NO❗
└	zapAcross	External ❗	🛑	NO❗
└	zapOut	External ❗	🛑	NO❗
└	swapToken	External ❗	🛑	NO❗
└	swapToNative	External ❗	🛑	NO❗
└	_approveTokenIfNeeded	Private 🔐	🛑	
└	_swapTokenToLP	Private 🔐	🛑	
└	_swapNativeToLP	Private 🔐	🛑	
└	_swapHalfNativeAndProvide	Private 🔐	🛑	

└	_swapNativeToEqualTokensAndProvide	Private 🔐	🛑		
└	_swapNativeForToken	Private 🔐	🛑		
└	_swapTokenForNative	Private 🔐	🛑		
└	_swap	Private 🔐	🛑		
└	_estimateSwap	Private 🔐			
└	setTokenBridgeForRouter	External ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	onlyOwner
└	setUseNativeRouter	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
🆎	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “Well Secured”.

✓ No volatile code.
✓ No high severity issues were found.

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 the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

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 Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico 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 Saferico 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, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, 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 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.