Smart Contract

Security Assessment

For SupChain

1 May 2024

Ascendant

Ascendant

@ascendantfi

www.ascendant.finance

Ascendant

Table of Contents

3	Disclaimer
4	Executive Summary
5	Overview
6	Findings Summary & Legend
8	Manual Review • Issue Checking Status • Audit Findings • Functional Test Status • Omitted Results
17	Automated Review • Unified Model Language
19	Conclusion

DISCLAIMER

This independent audit has been conducted to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by the auditor.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and the auditor is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will the auditor or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Auditor retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Auditor is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. The auditor may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
🔴 High	0
🟠 Medium	0
🟡 Low	7
🟣 Informational	24
Total	31

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Hardhat
- Remix IDE
- Slither
- Sol2UML

Overview

This report has been prepared for SupChain for the Ethereum Network. This audit provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	SupChain
Platform	Ethereum
Language	Solidity

Contracts Assessed

Name	**Location**
Supchain	Not Published
IERC20Errors	In Supchain contract
IERC721Errors	In Supchain contract
IERC1155Errors	In Supchain contract
IERC20	In Supchain contract
Context	In Supchain contract
Ownable	In Supchain contract

Name	Location
IERC20Metadata	In Supchain contract
IRouter	In Supchain contract
IFactory	In Supchain contract

Findings Summary

Severity	Found
🔴 High	0
🟠 Medium	0
🟡 Low	7
🟣 Informational	24
Total	31

Classification of Issues

🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	LOWx5
Contract	Supchain
Description	Multiplication on the result of a division

Code Snippet	
	```
	if (isBuy) {
	uint256 tax = value * BUY_TAX / 100;
	super._update(from, address(this), tax);
	super._update(address(this), address(0), tax * BURN_FEE / 100);
	super._update(address(this), _ecosystemAddress, tax * ECOSYSTEM_FEE / 100);
	super._update(address(this), _stakingAddress, tax * POOL_LIQUIDITY_FEE / 100);

	value = value - tax;
	}
	if (isSell) {
	uint256 tax = value * SELL_TAX / 100;
	super._update(from, address(this), tax);
	super._update(address(this), address(0), tax * BURN_FEE / 100);
	super._update(address(this), _ecosystemAddress, tax * ECOSYSTEM_FEE / 100);

	value = value - tax;
	}
	```

Recommendation	Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss. Consider multiplying before dividing.
Status	ACKNOWLEDGED

Audit Findings

Severity	Lowx2
Contract	Supchain
Description	Lack of zero check
Code Snippet	setEcosystemAddress() setStakingAddress
Recommendation	A require statement should be added prior to the assignment of the admin variable to ensure that the input address is not address(0).
Status	ACKNOWLEDGED

Functional Test Status

Function Name	Type/Return Type	Score
Context		
_contextSuffixLength	internal	PASS
_msgData	internal	PASS
_msgSender	internal	PASS
Ownable		
_checkOwner	internal	PASS
_transferOwnership	internal	PASS
constructor	internal	PASS
owner	public	PASS
renounceOwnership	public	PASS
transferOwnership	public	PASS
IRouter		
WETH	external	PASS
addLiquidityETH	external	PASS
factory	external	PASS
swapTokensForETHSupportingFeeOnTransferTokens	external	PASS
IFactory		
createPair	external	PASS

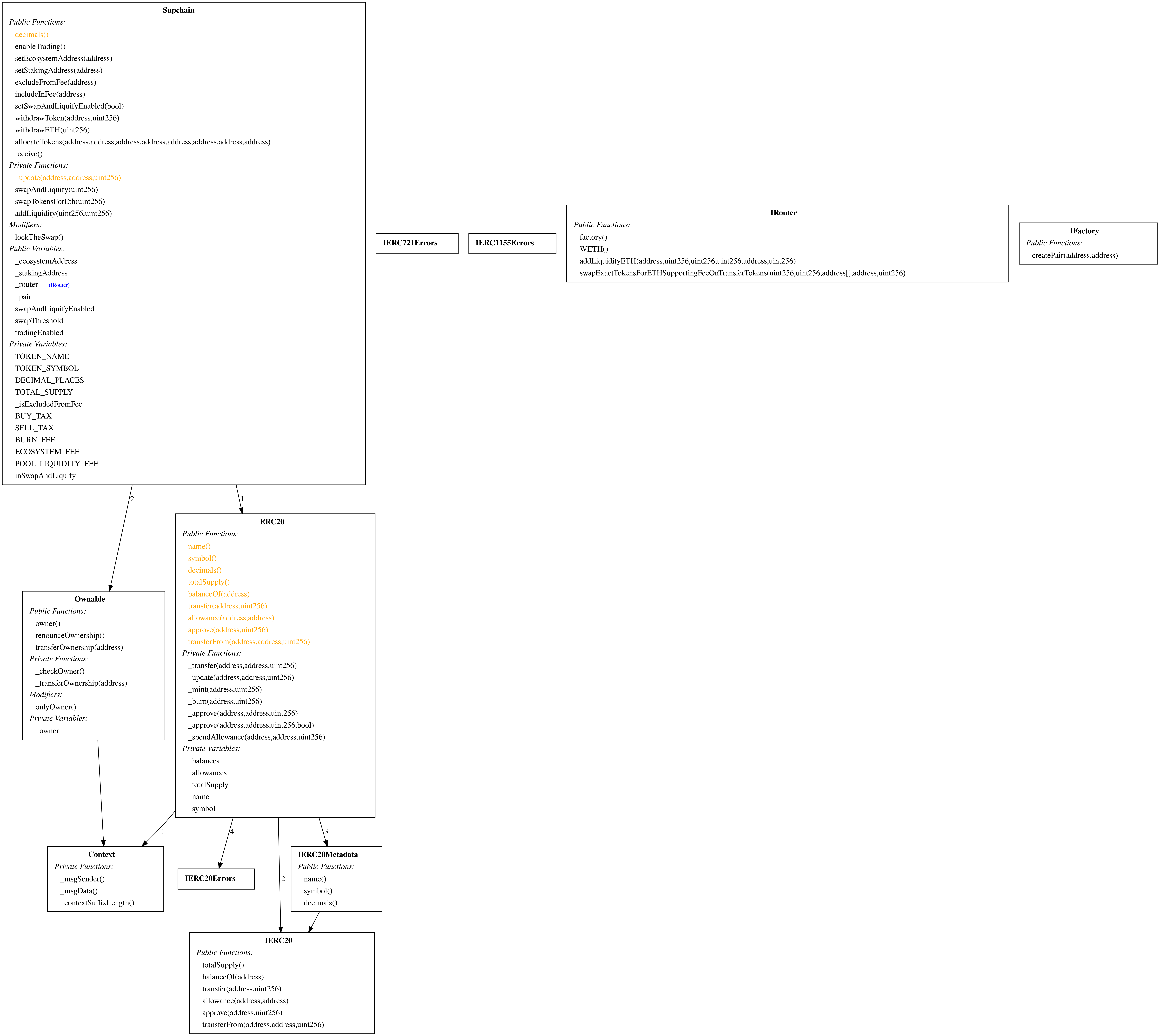
Supchain		
_update	internal	PASS
addLiquidity	private	PASS
allocateTokens	external	PASS
constructor	public	PASS
decimals	public	PASS
enableTrading	public	PASS
excludeFromFee	public	PASS
includeInFee	public	PASS
receive	external	PASS
setEcosystemAddress	external	PASS
setStakingAddress	external	PASS
setSwapAndLiquifyEnabled	public	PASS
swapAndLiquify	private	PASS
swapTokensForEth	private	PASS
withdrawETH	external	PASS
withdrawToken	external	PASS
IERC20		
allowance	external	PASS
approve	external	PASS
balanceOf	external	PASS
totalSupply	external	PASS

transfer	external	PASS
transferFrom	external	PASS
IERC20Metadata		
decimals	external	PASS
name	external	PASS
symbol	external	PASS
ERC20		
_approve	internal	PASS
_burn	internal	PASS
_mint	internal	PASS
_spendAllowance	internal	PASS
_transfer	internal	PASS
_update	internal	PASS
allowance	public	PASS
approve	public	PASS
balanceOf	public	PASS
constructor	internal	PASS
decimals	public	PASS
name	public	PASS
symbol	public	PASS
totalSupply	public	PASS
transfer	public	PASS
transferFrom	public	PASS

Omitted Results

Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.

Automated Review


Supchain
Public Functions:
decimals()
enableTrading()
setEcosystemAddress(address)
setStakingAddress(address)
excludeFromFee(address)
includeInFee(address)
setSwapAndLiquifyEnabled(bool)
withdrawToken(address,uint256)
withdrawETH(uint256)
allocateTokens(address,address,address,address,address,address,address,address)
receive()
Private Functions:
_update(address,address,uint256)
swapAndLiquify(uint256)
swapTokensForEth(uint256)
addLiquidity(uint256,uint256)
Modifiers:
lockTheSwap()
Public Variables:
_ecosystemAddress
_stakingAddress
_router (IRouter)
_pair
swapAndLiquifyEnabled
swapThreshold
tradingEnabled
Private Variables:
TOKEN_NAME
TOKEN_SYMBOL
DECIMAL_PLACES
TOTAL_SUPPLY
_isExcludedFromFee
BUY_TAX
SELL_TAX
BURN_FEE
ECOSYSTEM_FEE
POOL_LIQUIDITY_FEE
inSwapAndLiquify
IERC721Errors
IERC1155Errors
IRouter
Public Functions:
factory()
WETH()
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
IFactory
Public Functions:
createPair(address,address)
2
1
ERC20
Public Functions:
name()
symbol()
decimals()
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
Private Functions:
_transfer(address,address,uint256)
_update(address,address,uint256)
_mint(address,uint256)
_burn(address,uint256)
_approve(address,address,uint256)
_approve(address,address,uint256,bool)
_spendAllowance(address,address,uint256)
Private Variables:
_balances
_allowances
_totalSupply
_name
_symbol
Ownable
Public Functions:
owner()
renounceOwnership()
transferOwnership(address)
Private Functions:
_checkOwner()
_transferOwnership(address)
Modifiers:
onlyOwner()
Private Variables:
_owner
1
4
3
2
Context
Private Functions:
_msgSender()
_msgData()
_contextSuffixLength()
IERC20Errors
IERC20Metadata
Public Functions:
name()
symbol()
decimals()
IERC20
Public Functions:
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)

Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantfi

www.ascendant.finance

Ascendant