Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: CorkToken Protocol
Website: https://corkscrew.financial
Platform: AVAX
Language: Solidity
Date: April 16th, 2022

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Corkscrew team to perform the Security audit of the CorkToken Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 16th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

The Cork Token Contracts have functions like mint, burn, airdropTo, isBlacklisted, setBlacklisted, Blacklist, initialize, bailOutMint, claim, swap, skim, sync, _mintFee, etc. The Cork Token contract inherits the ERC20, Ownable, SafeMath, ERC1155Upgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable, SafeMathUpgradeable standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

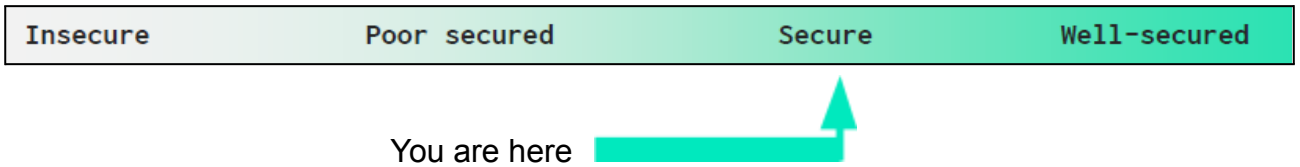
Name	**Code Review and Security Analysis Report for CorkToken Protocol Smart Contracts**
Platform	**AVAX / Solidity**
File 1	CorkToken.sol
File 1 MD5 Hash	B676FE4D9A7A3EE7A72BE997D0608677
File 2	Node.sol
File 2 MD5 Hash	ACFFDECDF4E6FFF628F4E8651F2F63AA
File 3	Pair.sol
File 3 MD5 Hash	E6EDED6794F2D9136FA9A99E45ACE3DB
Audit Date	April 16th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 CorkToken.sol • Name: Corkscrew • Symbole: CORK • Decimals: 18 • CorkToken contract has functions like: initialize, mint, burn, balanceOf, etc.	**YES, This is valid.**
File 2 Node.sol • Node contract has functions like: initialize, setBlacklisted, withdraw, nodeInit, etc.	**YES, This is valid.**
File 3 Pair.sol • Name: Joe LP Token • Symbol: JLP • Decimals: 18 • Minimum Liquidity: 1000	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts do contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 2 high, 2 medium and 1 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 3 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the CorkToken Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the CorkToken Protocol.

The CorkToken team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given a CorkToken Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://corkscrew.financial which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

CorkToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	owner	read	Passed	No Issue
5	onlyOwner	modifier	Passed	No Issue
6	renounceOwnership	write	access only Owner	No Issue
7	transferOwnership	write	access only Owner	No Issue
8	__ERC20_init	internal	access only Initializing	No Issue
9	__ERC20_init_unchained	internal	access only Initializing	No Issue
10	name	read	Passed	No Issue
11	symbol	read	Passed	No Issue
12	decimals	read	Passed	No Issue
13	totalSupply	read	Passed	No Issue
14	balanceOf	read	Passed	No Issue
15	transfer	write	Passed	No Issue
16	allowance	read	Passed	No Issue
17	approve	write	Passed	No Issue
18	transferFrom	write	Passed	No Issue
19	increaseAllowance	write	Passed	No Issue
20	decreaseAllowance	write	Passed	No Issue
21	_transfer	internal	Passed	No Issue
22	_mint	internal	Passed	No Issue
23	_burn	internal	Passed	No Issue
24	_approve	internal	Passed	No Issue
25	_spendAllowance	internal	Passed	No Issue
26	_beforeTokenTransfer	internal	Passed	No Issue
27	_afterTokenTransfer	internal	Passed	No Issue
28	initialize	write	Anyone can initialize contract	Refer Audit Findings
29	mint	write	Same function logic with different name	Refer Audit Findings
30	burn	write	The owner can burn anyone's token	Refer Audit Findings
31	balanceOf	read	Passed	No Issue
32	_transfer	internal	Passed	No Issue
33	transferFrom	write	Passed	No Issue
34	transfer	write	Passed	No Issue

35	transferTax	external	Passed	No Issue
36	resetContract	external	access only Owner	No Issue
37	setApprove	external	Ambiguous Error Message	Refer Audit Findings
38	setApproveByOwner	external	access only Owner	No Issue
39	airdropTo	external	Owner can mint unlimited tokens, Same function logic with different name	Refer Audit Findings
40	setBlacklisted	write	access only Owner	No Issue
41	isBlacklisted	read	Passed	No Issue

Node.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	owner	read	Passed	No Issue
5	onlyOwner	modifier	Passed	No Issue
6	renounceOwnership	write	access only Owner	No Issue
7	transferOwnership	write	access only Owner	No Issue
8	_transferOwnership	internal	Passed	No Issue
9	__ReentrancyGuard_init	internal	access only Initializing	No Issue
10	__ReentrancyGuard_init_unchained	internal	access only Initializing	No Issue
11	nonReentrant	modifier	Passed	No Issue
12	__ERC1155_init	internal	access only Initializing	No Issue
13	__ERC1155_init_unchained	internal	access only Initializing	No Issue
14	supportsInterface	read	Passed	No Issue
15	uri	read	Passed	No Issue
16	balanceOf	read	Passed	No Issue
17	balanceOfBatch	read	Passed	No Issue
18	setApprovalForAll	write	Passed	No Issue
19	isApprovedForAll	read	Passed	No Issue
20	safeTransferFrom	write	Passed	No Issue
21	safeBatchTransferFrom	write	Passed	No Issue
22	_safeTransferFrom	internal	Passed	No Issue
23	_safeBatchTransferFrom	internal	Passed	No Issue
24	_setURI	internal	Passed	No Issue
25	_mint	internal	Passed	No Issue

26	_mintBatch	internal	Passed	No Issue
27	_burn	internal	Passed	No Issue
28	_burnBatch	internal	Passed	No Issue
29	_setApprovalForAll	internal	Passed	No Issue
30	_beforeTokenTransfer	internal	Passed	No Issue
31	_doSafeTransferAcceptanceCheck	write	Passed	No Issue
32	_doSafeBatchTransferAcceptanceCheck	write	Passed	No Issue
33	_asSingletonArray	write	Passed	No Issue
34	Blacklist	modifier	Passed	No Issue
35	setBlacklisted	write	access only Owner	No Issue
36	initialize	write	Anyone can initialize contract	Refer Audit Findings
37	setManager	write	access only Owner	No Issue
38	withdraw	write	Critical operation lacks event log	Refer Audit Findings
39	setClaimFeePercentage	write	access only Owner	No Issue
40	withdrawCork	write	access only Owner	No Issue
41	nodeInit	internal	Passed	No Issue
42	mint	write	Typing mistake	Refer Audit Findings
43	mintTo	write	access only Owner	No Issue
44	bailOutMint	write	Critical operation lacks event log	Refer Audit Findings
45	setTradeActivate	write	access only Owner	No Issue
46	safeTransferFrom	write	access only Owner	No Issue
47	claim	external	Critical operation lacks event log	Refer Audit Findings
48	claimPartial	external	Critical operation lacks event log	Refer Audit Findings
49	claimById	external	Critical operation lacks event log	Refer Audit Findings
50	getClaimableCork	read	Passed	No Issue
51	calculateClaimableAmount	read	Passed	No Issue
52	calculateMainAmount	write	Passed	No Issue
53	calculateSnowballAmount	read	Passed	No Issue
54	sellableCork	external	Passed	No Issue
55	getClaimFee	read	Passed	No Issue
56	getClaimableCorkById	read	Passed	No Issue
57	getClaimFeeById	read	Passed	No Issue
58	getClaimFeeByValue	read	Passed	No Issue
59	updateCollection	write	Critical operation lacks event log	Refer Audit Findings
60	addToCollection	write	Critical operation lacks event log	Refer Audit Findings
61	swapTokensForAVAX	write	Passed	No Issue

62	resetContract	external	access only Owner	No Issue
63	setTraderJoeDivideSide	write	access only Owner	No Issue
64	getCorkPrice	read	Passed	No Issue
65	getOwnedNodeCountByType	read	Passed	No Issue
66	getNodeState	read	Passed	No Issue
67	_getNextTokenID	read	Passed	No Issue
68	_incrementTokenID	write	Passed	No Issue
69	_amount2cork	read	Passed	No Issue
70	setPresaleActive	write	Unused code	Refer Audit Findings
71	checkPresaleActive	read	Passed	No Issue
72	getNodeROI	read	Passed	No Issue
73	updateRewardInterval	write	access only Owner	No Issue
74	deleteTo	write	access only Owner	No Issue

Pair.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getReserves	read	Passed	No Issue
3	_safeTransfer	write	Passed	No Issue
4	initialize	external	Passed	No Issue
5	_update	write	Passed	No Issue
6	_mintFee	write	Passed	No Issue
7	mint	external	Passed	No Issue
8	burn	external	Passed	No Issue
9	swap	external	Passed	No Issue
10	skim	external	Passed	No Issue
11	sync	external	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token 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
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
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Owner can mint unlimited tokens: **CorkToken.sol**

```
function airdropTo(address to, uint256 amount) external onlyOwner(){
    _mint(to, amount);
}
```

There is no limit for minting tokens. Thus the owner can mint unlimited tokens to any account.

Resolution: There should be a limit for minting or need to confirm, if it is a part of the plan then disregard this issue.

(2) The owner can burn anyone's token: **CorkToken.sol**

```
function burn(address account, uint256 amount) public override onlyOwner {
    _burn(account, amount);
}
```

The owner can burn any users' tokens.

Resolution: We suggest changing the code so only token holders can burn their own tokens and not anyone else. Not even a contract creator.

Medium

(1) Claim Fee Percentage Limit is not set: **Node.sol**

The owner of the contract can set the individual percentage to any variable.This might deter investors as they could be wary that these fees might one day be set to 100% to force transfers to go to the contract owner.

Resolution: Consider adding a limit on fee percentage adjustment function.

Low

(1) Critical operation lacks event log: **Node.sol**

Missing event log for:

- claim
- claimPartial
- claimById
- withdraw
- bailOutMint
- addToCollection
- updateCollection

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Anyone can initialize contract:

CorkToken.sol

```
function initialize(uint256 initialSupply) initializer public {
    __Ownable_init();
    __ERC20_init("Corkscrew", "CORK");
    _mint(msg.sender, initialSupply * 10**decimals());
}
```

Node.sol

```
function initialize() initializer public {
    __Ownable_init();
    __ERC1155_init("https://example.com/{id}.json");
    nodeInit();
    _percentRate = 10**8;
    _rewardInterval = 1 days;
    _periodDays = 30;
    claimFeePercentage = 10;
}
```

Anyone can initialize() function and make the owner itself.

Resolution: We suggest executing the initialize() function just after the deploy on mainnet so that the deployer will be the owner..

(2) Same function logic with different name: **CorkToken.sol**

```
function airdropTo(address to, uint256 amount) external onlyOwner(){
    _mint(to, amount);
}
```

```
function mint(address to, uint256 amount) public override onlyOwner {
    _mint(to, amount);
}
```

There are two functions "mint()" and "airdropTo()", In both functions internal logic is the same, it's doing a mint token.

Resolution: Deployer has to confirm before deploying the contract to production.

(3) Multiple pragma: **Pair.sol**

There are multiple pragmas with different compiler versions.

Resolution: We suggest using only one pragma and removing the other.

(4) Unused code: **Node.sol**

```
function setPresaleActive(bool _isPresaleActive) public onlyOwner {
    //require(!isPresaleActive, "Presale was already activated");
    isPresaleActive = _isPresaleActive;
}
```

There is an unused code comment.

Resolution: Remove unused commented code.

(5) Use latest solidity version: **Pair.sol**

```
pragma solidity = 0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.

Resolution: We suggest using version > 0.8.0.

(6) Ambiguous Error Message: **CorkToken.sol**

```
require(from == swapAddress, "hmmm... what doing?");

require(
    ISwapCork(swapAddress).getSwapAvailable(),
    "hmmm... what doing?"
);
```

```
function setApprove(
    address owner,
    address spender,
    uint256 amount
) external override {
    require(msg.sender == swapAddress, "hmmm... what doing?");
    _approve(owner, spender, amount);
}
```

The mentioned error message does not explain exactly the error of the operation.

Resolution: As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system.

(7) Typing mistake: **Node.sol**

```
function mint(uint256 _nodeType, uint256 _amount, string calldata _uri) public Blacklist {
    require(
        collection[_nodeType].currentSupply <=
            collection[_nodeType].maxSupply,
        "all of this collection are purchased"
    );

    require(
        collection[_nodeType].currentSupply + _amount <=
            collection[_nodeType].maxSupply,
        "there is not enought nodes to sell"
    );
```

There is a typing mistake in requiring "enought".

Resolution: Correct the spelling in the error message.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setBlacklisted: The Node owner can set the address in blacklisted.
- setManager: The Node owner can set the manager address.
- setClaimFeePercentage: The Node owner can set claim fee percentage value.
- withdrawCork: The Node owner can withdraw cork from this contract.
- mintTo: The Node owner can mint cork from wallet.
- setTradeActivate: The Node owner can set trade active status.
- updateCollection: The Node owner can update collection values like: id, title, price, maxSupply, firstRun, trueYield, snowball, maxSnowball, maxDailySell, currentSupply, purchaseLimit.
- addToCollection: The Node owner can add a new collection.
- resetContract: The Node owner can reset pair address, cork address, swap address.
- setTraderJoeDivideSide: The Node owner can update trader joe divideside status.
- setPresaleActive: The Node owner can update presale active status.
- updateRewardInterval: The Node owner can update the reward interval value.
- mint: The CorkToken owner can mint an amount from the address.
- burn: The CorkToken owner can burn an amount from the address.
- resetContract: The CorkToken owner can reset pair address, swap address.
- setApproveByOwner: The CorkToken owner can update approved status by owner.
- airdropTo: The CorkToken owner can airdrop the amount from the address.
- setBlacklisted: The CorkToken owner can update addresses in blacklist.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the airdrop smart contract once its function is completed.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts. **So, the smart contracts will be ready for the mainnet deployment after fixing or acknowledging those issues**.

Since possible test cases can be unlimited for such smart contracts 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.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

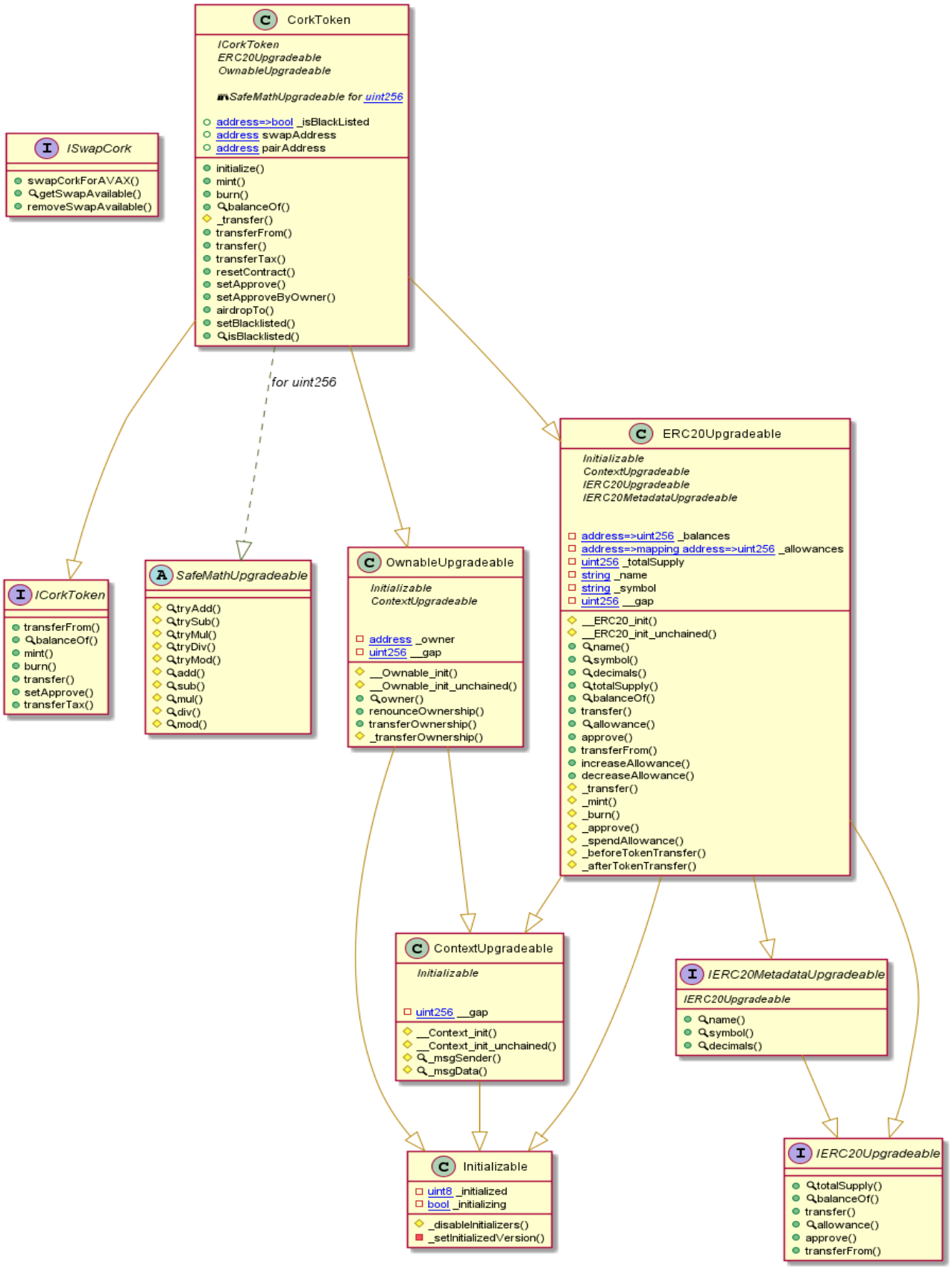
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

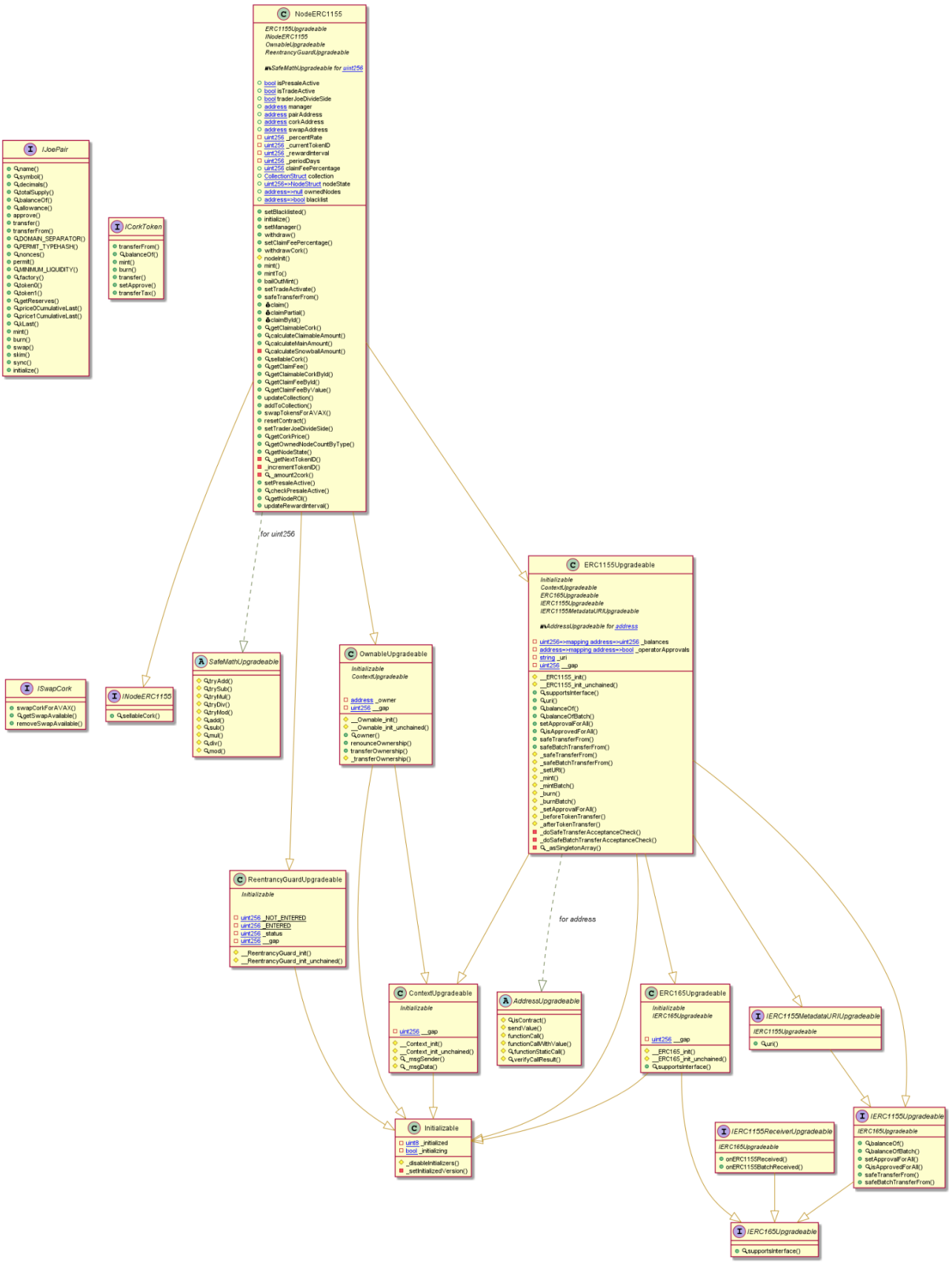
Appendix

Code Flow Diagram - CorkToken Protocol

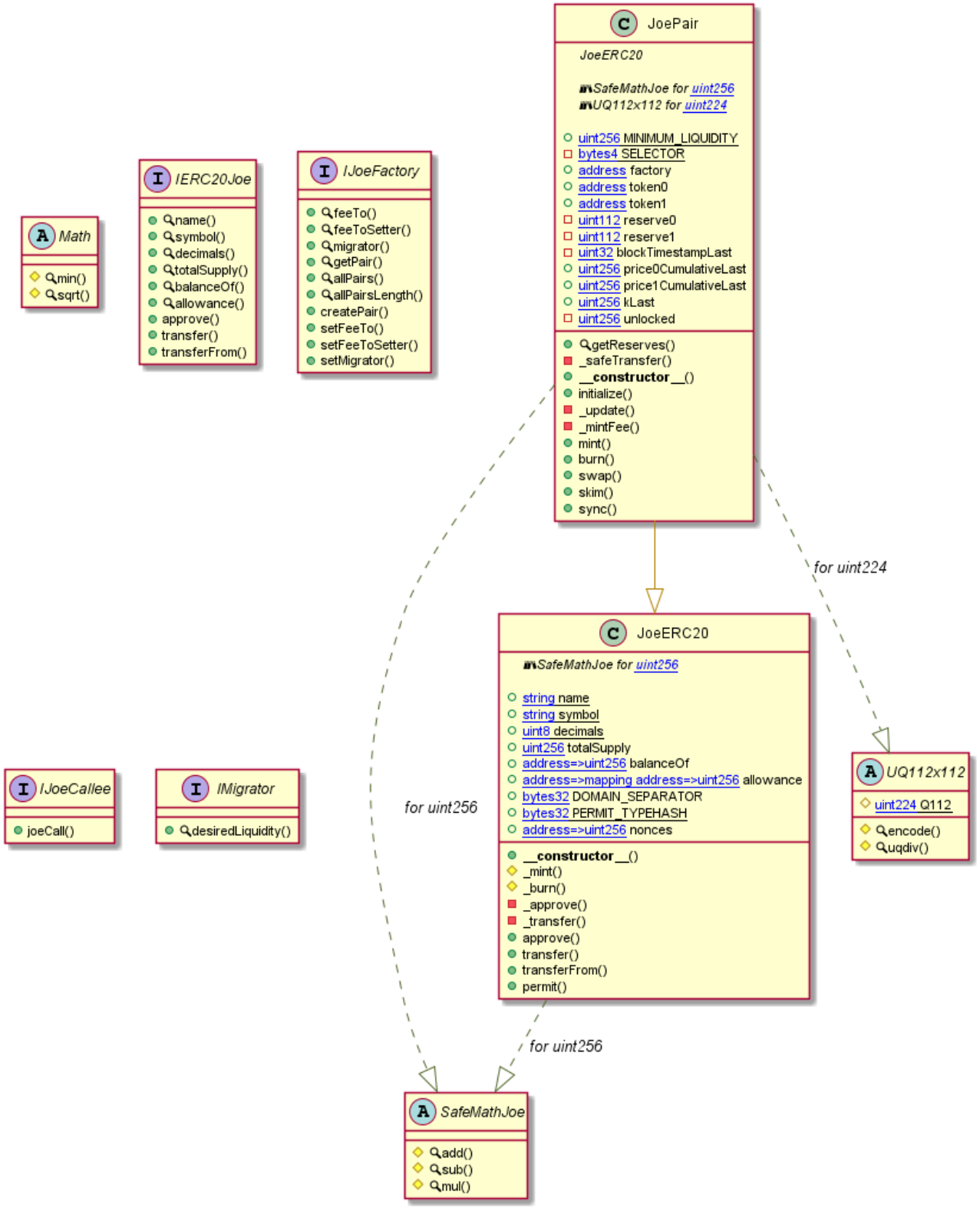
CorkToken Diagram



Node Diagram



Pair Diagram



Slither Results Log

Slither log >> CorkToken.sol

```
INFO:Detectors:
CorkToken.setApprove(address,address,uint256).owner (CorkToken.sol#978) shadows:
        - OwnableUpgradeable.owner() (CorkToken.sol#468-470) (function)
CorkToken.setApproveByOwner(address,address,uint256).owner (CorkToken.sol#987) shadows:
        - OwnableUpgradeable.owner() (CorkToken.sol#468-470) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in CorkToken._transfer(address,address,uint256) (CorkToken.sol#921-945):
        External calls:
        - ISwapCork(swapAddress).removeSwapAvailable() (CorkToken.sol#943)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (CorkToken.sol#730)
                - super._transfer(from,to,amount) (CorkToken.sol#944)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ContextUpgradeable.__Context_init() (CorkToken.sol#427-428) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (CorkToken.sol#430-431) is never used and should be removed
ContextUpgradeable._msgData() (CorkToken.sol#436-438) is never used and should be removed
Initializable._disableInitializers() (CorkToken.sol#407-409) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (CorkToken.sol#198-200) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (CorkToken.sol#240-242) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (CorkToken.sol#296-305) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (CorkToken.sol#256-258) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (CorkToken.sol#322-331) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (CorkToken.sol#226-228) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (CorkToken.sol#212-214) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (CorkToken.sol#273-282) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (CorkToken.sol#127-133) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (CorkToken.sol#169-174) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (CorkToken.sol#181-186) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (CorkToken.sol#152-162) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (CorkToken.sol#140-145) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (CorkToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (CorkToken.sol#427-428) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (CorkToken.sol#430-431) is not in mixedCase
Variable ContextUpgradeable.__gap (CorkToken.sol#445) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (CorkToken.sol#457-459) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (CorkToken.sol#461-463) is not in mixedCase
Variable OwnableUpgradeable.__gap (CorkToken.sol#515) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (CorkToken.sol#537-539) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (CorkToken.sol#541-544) is not in mixedCase
Variable ERC20Upgradeable.__gap (CorkToken.sol#876) is not in mixedCase
Event CorkTokenblacklist(address,bool) (CorkToken.sol#882) is not in CapWords
Parameter CorkToken.resetContract(address,address)._pairAddress (CorkToken.sol#969) is not in mixedCase
Parameter CorkToken.resetContract(address,address)._swapAddress (CorkToken.sol#969) is not in mixedCase
Variable CorkToken._isBlackListed (CorkToken.sol#881) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
OwnableUpgradeable.__gap (CorkToken.sol#515) is never used in CorkToken (CorkToken.sol#879-1006)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (CorkToken.sol#487-489)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (CorkToken.sol#495-498)
name() should be declared external:
        - ERC20Upgradeable.name() (CorkToken.sol#549-551)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (CorkToken.sol#557-559)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (CorkToken.sol#581-583)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (CorkToken.sol#623-627)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (CorkToken.sol#668-672)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (CorkToken.sol#688-697)
initialize(uint256) should be declared external:
        - CorkToken.initialize(uint256) (CorkToken.sol#892-896)
```

```
initialize(uint256) should be declared external:
        - CorkToken.initialize(uint256) (CorkToken.sol#892-896)
mint(address,uint256) should be declared external:
        - CorkToken.mint(address,uint256) (CorkToken.sol#903-905)
burn(address,uint256) should be declared external:
        - CorkToken.burn(address,uint256) (CorkToken.sol#907-909)
setApproveByOwner(address,address,uint256) should be declared external:
        - CorkToken.setApproveByOwner(address,address,uint256) (CorkToken.sol#986-992)
setBlacklisted(address,bool) should be declared external:
        - CorkToken.setBlacklisted(address,bool) (CorkToken.sol#997-1001)
isBlacklisted(address) should be declared external:
        - CorkToken.isBlacklisted(address) (CorkToken.sol#1003-1005)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CorkToken.sol analyzed (10 contracts with 75 detectors), 53 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Node.sol

```
INFO:Detectors:
NodeERC1155.mint(uint256,uint256,string)._uri (Node.sol#1638) shadows:
        - ERC1155Upgradeable._uri (Node.sol#981) (state variable)
NodeERC1155.mintTo(address[],uint256[],uint256[],uint256[],string[])._uri (Node.sol#1709) shadows:
        - ERC1155Upgradeable._uri (Node.sol#981) (state variable)
NodeERC1155.bailOutMint(uint256,uint256,uint256,string)._uri (Node.sol#1745) shadows:
        - ERC1155Upgradeable._uri (Node.sol#981) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
NodeERC1155.setManager(address)._manager (Node.sol#1545) lacks a zero-check on :
                - manager = address(_manager) (Node.sol#1546)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
NodeERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (Node.sol#1806-1834) has external calls inside a loop: c
orkToken.transfer(from,claimableCork) (Node.sol#1823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Variable 'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (Node.so
l#1425)' in ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (Node.sol#1416-
1435) potentially used before declaration: response != IERC1155ReceiverUpgradeable.onERC1155Received.selector (Node.sol#1426
)
Variable 'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (Node.sol#
1429)' in ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (Node.sol#1416-14
35) potentially used before declaration: revert(string)(reason) (Node.sol#1430)
Variable 'ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response
 (Node.sol#1447)' in ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],byte
s) (Node.sol#1437-1458) potentially used before declaration: response != IERC1155ReceiverUpgradeable.onERC1155BatchReceived.
selector (Node.sol#1449)
Variable 'ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (
Node.sol#1452)' in ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes)
 (Node.sol#1437-1458) potentially used before declaration: revert(string)(reason) (Node.sol#1453)
Variable 'NodeERC1155.getCorkPrice().Res0 (Node.sol#2132)' in NodeERC1155.getCorkPrice() (Node.sol#2130-2140) potentially us
ed before declaration: (Res0,Res1) = IJoePair(pairAddress).getReserves() (Node.sol#2136)
Variable 'NodeERC1155.getCorkPrice().Res1 (Node.sol#2132)' in NodeERC1155.getCorkPrice() (Node.sol#2130-2140) potentially us
ed before declaration: (Res0,Res1) = IJoePair(pairAddress).getReserves() (Node.sol#2136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
INFO:Detectors:
Reentrancy in NodeERC1155.bailOutMint(uint256,uint256,uint256,string) (Node.sol#1741-1796):
        External calls:
        - _mint(_msgSender(),nodeType,amount,) (Node.sol#1778)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        State variables written after the call(s):
        - _incrementTokenID() (Node.sol#1782)
                - _currentTokenID ++ (Node.sol#2167)
Reentrancy in NodeERC1155.mint(uint256,uint256,string) (Node.sol#1638-1707):
        External calls:
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount / 2) (Node.sol#1677-1681)
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount) (Node.sol#1683-1687)
        State variables written after the call(s):
        - _incrementTokenID() (Node.sol#1692)
                - _currentTokenID ++ (Node.sol#2167)
Reentrancy in NodeERC1155.mintTo(address[],uint256[],uint256[],uint256[],string[]) (Node.sol#1709-1739):
        External calls:
        - _mint(to[i],_nodeType[i],1,) (Node.sol#1711)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        State variables written after the call(s):
        - _incrementTokenID() (Node.sol#1714)
                - _currentTokenID ++ (Node.sol#2167)
        - collection[_nodeType[i]].currentSupply ++ (Node.sol#1737)
        - nodeState[_id].purchaser = to[i] (Node.sol#1715)
        - nodeState[_id].nodeType = _nodeType[i] (Node.sol#1716)
        - nodeState[_id].purchasedAt = block.timestamp (Node.sol#1719)
        - nodeState[_id].purchasedAt = mintDate[i] (Node.sol#1721)
        - nodeState[_id].snowballAt = block.timestamp (Node.sol#1725)
        - nodeState[_id].snowballAt = claimDate[i] (Node.sol#1727)
        - nodeState[_id].uri = _uri[i] (Node.sol#1730)
        - ownedNodes[to[i]].push(_id) (Node.sol#1736)
Reentrancy in NodeERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (Node.sol#1806-1834):
        External calls:
        - super.safeTransferFrom(from,to,id,amount,data) (Node.sol#1809)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        State variables written after the call(s):
        - nodeState[ownedNodes[from][i]].snowballAt = block.timestamp (Node.sol#1820)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in NodeERC1155.bailOutMint(uint256,uint256,uint256,string) (Node.sol#1741-1796):
        External calls:
        - _mint(_msgSender(),nodeType,amount,) (Node.sol#1778)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        Event emitted after the call(s):
        - URI(_uri,_id) (Node.sol#1790)
Reentrancy in NodeERC1155.mint(uint256,uint256,string) (Node.sol#1638-1707):
        External calls:
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount / 2) (Node.sol#1677-1681)
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount) (Node.sol#1683-1687)
        Event emitted after the call(s):
        - URI(_uri,_id) (Node.sol#1699)
Reentrancy in NodeERC1155.mint(uint256,uint256,string) (Node.sol#1638-1707):
        External calls:
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount / 2) (Node.sol#1677-1681)
        - corkToken.transferFrom(_msgSender(),address(this),collection[_nodeType].price * _amount) (Node.sol#1683-1687)
        - _mint(_msgSender(),_nodeType,_amount,) (Node.sol#1706)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        Event emitted after the call(s):
        - TransferSingle(operator,address(0),to,id,amount) (Node.sol#1236)
                - _mint(_msgSender(),_nodeType,_amount,) (Node.sol#1706)
Reentrancy in NodeERC1155.mintTo(address[],uint256[],uint256[],uint256[],string[]) (Node.sol#1709-1739):
        External calls:
        - _mint(to[i],_nodeType[i],1,) (Node.sol#1711)
                - IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (Node.sol#1425-1433)
        Event emitted after the call(s):
        - URI(_uri[i],_id) (Node.sol#1733)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
NodeERC1155.bailOutMint(uint256,uint256,uint256,string) (Node.sol#1741-1796) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(claimableCork >= wastedCork,claimable cork is less than price) (Node.sol#1754-1757)
        - wastedCork < claimableCork (Node.sol#1773)
NodeERC1155.claim() (Node.sol#1840-1862) uses timestamp for comparisons
        Dangerous comparisons:
```

```
NodeERC1155.calculateMainAmount(uint256,uint256,uint256,uint256,uint256) (Node.sol#1969-1995) uses timestamp for comparisons
        Dangerous comparisons:
        - _lastedMainDays > _roiTime (Node.sol#1978)
        - (_lastedMainDays - _noClaimDays) < _roiTime (Node.sol#1982)
NodeERC1155.calculateSnowballAmount(uint256,uint256,uint256,uint256) (Node.sol#1997-2021) uses timestamp for comparisons
        Dangerous comparisons:
        - _lastedSnowballDays < _roiTime (Node.sol#2004)
        - i <= _lastedSnowballDays (Node.sol#2006)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Node.sol#507-527) uses assembly
        - INLINE ASM (Node.sol#519-522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
NodeERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (Node.sol#1806-1834) compares to a boolean constant:
        -require(bool,string)(isTradeActive == true,Node: Transfer is disabled) (Node.sol#1807)
NodeERC1155.Blacklist() (Node.sol#1525-1528) compares to a boolean constant:
        -require(bool,string)(blacklist[_msgSender()] == false,you're blacklisted) (Node.sol#1526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (Node.sol#418-420) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (Node.sol#428-434) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (Node.sol#447-453) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Node.sol#461-472) is never used and should be remove
d
AddressUpgradeable.functionStaticCall(address,bytes) (Node.sol#480-482) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (Node.sol#490-499) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (Node.sol#393-398) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Node.sol#507-527) is never used and should be removed
ContextUpgradeable.__Context_init() (Node.sol#806-807) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (Node.sol#809-810) is never used and should be removed
ContextUpgradeable._msgData() (Node.sol#815-817) is never used and should be removed
ERC1155Upgradeable._burn(address,uint256,uint256) (Node.sol#1284-1306) is never used and should be removed
ERC1155Upgradeable._burnBatch(address,uint256[],uint256[]) (Node.sol#1315-1341) is never used and should be removed
ERC1155Upgradeable._mintBatch(address,uint256[],uint256[],bytes) (Node.sol#1252-1274) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (Node.sol#828-829) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (Node.sol#831-832) is never used and should be removed
```

```
        - collection.push(CollectionStruct(Blue,400000000000000000,30000,1500000,350000,1700,50000,15000000,0,30)) (Node.so
l#1572-1585)
NodeERC1155.nodeInit() (Node.sol#1571-1628) uses literals with too many digits:
        - collection.push(CollectionStruct(Red,1000000000000000000,15000,2000000,900000,3333,100000,10000000,0,30)) (Node.s
ol#1586-1599)
NodeERC1155.nodeInit() (Node.sol#1571-1628) uses literals with too many digits:
        - collection.push(CollectionStruct(Black,10000000000000000000,5000,2200000,1000000,3333,100000,5000000,0,30)) (Node
.sol#1600-1613)
NodeERC1155.nodeInit() (Node.sol#1571-1628) uses literals with too many digits:
        - collection.push(CollectionStruct(DoubleBlack,100000000000000000000,1000,2200000,1000000,4167,125000,5000000,0,10)
) (Node.sol#1614-1627)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (Node.sol#968) is never used in NodeERC1155 (Node.sol#1475-2200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (Node.sol#886-888)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (Node.sol#894-897)
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (Node.sol#1014-1016)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (Node.sol#1037-1053)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (Node.sol#1058-1060)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (Node.sol#1089-1101)
setBlacklisted(address,bool) should be declared external:
        - NodeERC1155.setBlacklisted(address,bool) (Node.sol#1530-1532)
initialize() should be declared external:
        - NodeERC1155.initialize() (Node.sol#1535-1543)
setManager(address) should be declared external:
        - NodeERC1155.setManager(address) (Node.sol#1545-1547)
withdraw() should be declared external:
        - NodeERC1155.withdraw() (Node.sol#1550-1559)
setClaimFeePercentage(uint256) should be declared external:
```

```
withdraw() should be declared external:
        - NodeERC1155.withdraw() (Node.sol#1550-1559)
setClaimFeePercentage(uint256) should be declared external:
        - NodeERC1155.setClaimFeePercentage(uint256) (Node.sol#1562-1564)
withdrawCork(uint256) should be declared external:
        - NodeERC1155.withdrawCork(uint256) (Node.sol#1567-1569)
mint(uint256,uint256,string) should be declared external:
        - NodeERC1155.mint(uint256,uint256,string) (Node.sol#1638-1707)
mintTo(address[],uint256[],uint256[],string[]) should be declared external:
        - NodeERC1155.mintTo(address[],uint256[],uint256[],uint256[],string[]) (Node.sol#1709-1739)
bailOutMint(uint256,uint256,uint256,string) should be declared external:
        - NodeERC1155.bailOutMint(uint256,uint256,uint256,string) (Node.sol#1741-1796)
setTradeActivate(bool) should be declared external:
        - NodeERC1155.setTradeActivate(bool) (Node.sol#1802-1804)
updateCollection(uint256,string,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) should be declared
external:
        - NodeERC1155.updateCollection(uint256,string,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint25
6) (Node.sol#2079-2093)
addToCollection(string,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) should be declared external:
        - NodeERC1155.addToCollection(string,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) (Node.
sol#2095-2108)
swapTokensForAVAX(uint256) should be declared external:
        - NodeERC1155.swapTokensForAVAX(uint256) (Node.sol#2110-2113)
setTraderJoeDivideSide(bool) should be declared external:
        - NodeERC1155.setTraderJoeDivideSide(bool) (Node.sol#2125-2127)
getNodeState(uint256) should be declared external:
        - NodeERC1155.getNodeState(uint256) (Node.sol#2158-2160)
setPresaleActive(bool) should be declared external:
        - NodeERC1155.setPresaleActive(bool) (Node.sol#2178-2181)
checkPresaleActive() should be declared external:
        - NodeERC1155.checkPresaleActive() (Node.sol#2183-2185)
updateRewardInterval(uint256) should be declared external:
        - NodeERC1155.updateRewardInterval(uint256) (Node.sol#2196-2198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Node.sol analyzed (17 contracts with 75 detectors), 166 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Pair.sol

```
INFO:Detectors:
JoePair.initialize(address,address)._token0 (Pair.sol#345) lacks a zero-check on :
                - token0 = _token0 (Pair.sol#347)
JoePair.initialize(address,address)._token1 (Pair.sol#345) lacks a zero-check on :
                - token1 = _token1 (Pair.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in JoePair.burn(address) (Pair.sol#443-472):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Pair.sol#464)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - _safeTransfer(_token1,to,amount1) (Pair.sol#465)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#469)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (Pair.sol#366-
368)
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#469)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (Pair.sol#369-
371)
Reentrancy in JoePair.swap(uint256,uint256,address,bytes) (Pair.sol#475-533):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (Pair.sol#498)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - _safeTransfer(_token1,to,amount1Out) (Pair.sol#499)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - IJoeCallee(to).joeCall(msg.sender,amount0Out,amount1Out,data) (Pair.sol#501-506)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#531)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (Pair.sol#366-
368)
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#531)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (Pair.sol#369-
371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in JoePair.burn(address) (Pair.sol#443-472):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Pair.sol#464)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - _safeTransfer(_token1,to,amount1) (Pair.sol#465)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (Pair.sol#471)
        - Sync(reserve0,reserve1) (Pair.sol#376)
                - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#469)
Reentrancy in JoePair.swap(uint256,uint256,address,bytes) (Pair.sol#475-533):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (Pair.sol#498)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - _safeTransfer(_token1,to,amount1Out) (Pair.sol#499)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
        - IJoeCallee(to).joeCall(msg.sender,amount0Out,amount1Out,data) (Pair.sol#501-506)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (Pair.sol#532)
        - Sync(reserve0,reserve1) (Pair.sol#376)
                - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#531)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
JoeERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Pair.sol#116-148) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Joe: EXPIRED) (Pair.sol#125)
JoePair._update(uint256,uint256,uint112,uint112) (Pair.sol#352-377) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (Pair.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
JoeERC20.constructor() (Pair.sol#43-59) uses assembly
        - INLINE ASM (Pair.sol#45-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
JoeERC20.constructor() (Pair.sol#43-59) uses assembly
        - INLINE ASM (Pair.sol#45-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Low level call in JoePair._safeTransfer(address,address,uint256) (Pair.sol#309-321):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#314-316)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable JoeERC20.DOMAIN_SEPARATOR (Pair.sol#30) is not in mixedCase
Parameter JoePair.initialize(address,address)._token0 (Pair.sol#345) is not in mixedCase
Parameter JoePair.initialize(address,address)._token1 (Pair.sol#345) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable JoePair.swap(uint256,uint256,address,bytes).balance0Adjusted (Pair.sol#522) is too similar to JoePair.swap(uint256,
uint256,address,bytes).balance1Adjusted (Pair.sol#523)
Variable JoePair.price0CumulativeLast (Pair.sol#283) is too similar to JoePair.price1CumulativeLast (Pair.sol#284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Slither:Pair.sol analyzed (9 contracts with 75 detectors), 20 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

CorkToken.sol

Gas & Economy

Gas costs:

Gas requirement of function CorkToken.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 549:4:

Gas costs:

Gas requirement of function CorkToken.transferTax is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 962:4:

Miscellaneous

Constant/View/Pure functions:

ISwapCork.swapCorkForAVAX(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 6:4:

Constant/View/Pure functions:

CorkToken.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 947:4:

Email: audit@EtherAuthority.io

Similar variable names:

CorkToken.burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 908:23:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 982:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 998:8:

Node.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 461:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in NodeERC1155.safeTransferFrom(address,address,uint256,uint256,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1806:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1695:41:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1557:27:

Gas & Economy

Gas costs:

Gas requirement of function NodeERC1155.withdraw is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1550:4:

Gas costs:

Gas requirement of function NodeERC1155.withdrawCork is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1567:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1812:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 12:4:

Miscellaneous

Constant/View/Pure functions:

NodeERC1155.calculateClaimableAmount(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1935:4:

Similar variable names:

NodeERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) : Variables have very similar names "to" and "id". Note: Modifiers are currently not considered by this static analysis.

Pos: 1813:59:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1917:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 2188:24:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 2193:15:

Pair.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JoePair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 380:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 362:39:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 314:44:

Gas & Economy

Gas costs:

Gas requirement of function JoePair.swap is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 475:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 197:4:

Miscellaneous

Similar variable names:

JoePair.getReserves() : Variables have very similar names "reserve0" and "_reserve1". Note: Modifiers are currently not considered by this static analysis.
Pos: 304:20:

Similar variable names:

JoePair.swap(uint256,uint256,address,bytes) : Variables have very similar names "token0" and "_token1". Note: Modifiers are currently not considered by this static analysis.
Pos: 499:46:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 486:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 458:18:

Solhint Linter

CorkToken.sol

```
CorkToken.sol:128:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:141:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:153:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:170:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:182:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:278:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:301:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:327:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:692:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:725:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:774:18: Error: Parse error: missing ';' at '{'
CorkToken.sol:825:22: Error: Parse error: missing ';' at '{'
```

Node.sol

```
Node.sol:137:18: Error: Parse error: missing ';' at '{'
Node.sol:150:18: Error: Parse error: missing ';' at '{'
Node.sol:162:18: Error: Parse error: missing ';' at '{'
Node.sol:179:18: Error: Parse error: missing ';' at '{'
Node.sol:191:18: Error: Parse error: missing ';' at '{'
Node.sol:287:18: Error: Parse error: missing ';' at '{'
Node.sol:310:18: Error: Parse error: missing ';' at '{'
Node.sol:336:18: Error: Parse error: missing ';' at '{'
Node.sol:1132:18: Error: Parse error: missing ';' at '{'
Node.sol:1174:22: Error: Parse error: missing ';' at '{'
Node.sol:1299:18: Error: Parse error: missing ';' at '{'
Node.sol:1333:22: Error: Parse error: missing ';' at '{'
```

Pair.sol

```
Pair.sol:4:1: Error: Compiler version =0.6.12 does not satisfy the r
semver requirement
Pair.sol:23:28: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:24:28: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:25:27: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:30:20: Error: Variable name must be in mixedCase
Pair.sol:45:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Pair.sol:125:29: Error: Avoid to make time-based decisions in your
business logic
Pair.sol:172:5: Error: Explicitly mark visibility of state
```

```
Pair.sol:314:45: Error: Avoid using low level calls.
Pair.sol:362:40: Error: Avoid to make time-based decisions in your
business logic
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io