Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	HOKK Token
Platform:	Ethereum, Heco Chain, Binance Smart Chain
Language:	Solidity
Date:	October 18th, 2021

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 HOKK Token team to perform the Security audit of the HOKK Token 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 October 18th, 2021.

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

HOKK Finance is a community-centric decentralized finance (DeFi) project.

Audit scope

Name	**Code Review and Security Analysis Report for HOKK Token Smart Contract**
Platform	**Ethereum / Solidity**
File 1	HOKK_ETH.sol
File 1 MD5 Hash	CAB9FCD99C2316F1B1E34D8E9185084C
File 2	HOKK_BEP.sol
File 2 MD5 Hash	C1660C6C24A033D7A9B5056FDFDDAF79
File 3	HOKK_HECO.sol
File 3 MD5 Hash	35A7A2CD58C4BF01DEFB82F247F51A68
Audit Date	October 18th, 2021

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 : HOKK_ETH.sol • Name: Hokkaido Inu • Symbol: HOKK • Decimals: 18 • Rewards Fee: 4% • Liquidity Fee: 4% • Total Supply : 1 billion	**YES, This is valid.**
File 2 : HOKK_BEP.sol • Name: Hokkaido Inu • Symbol: HOKK • Decimals: 18 • Rewards Fee: 4% • Liquidity Fee: 1% • Marketing Fee: 3% • Maximum Sell Transaction Amount: 100,000,000 • Swap Tokens At Amount: 2,00,000 • Gas For Processing: 5,00,000	**YES, This is valid.**
File 3 : HOKK_HECO.sol • Name: Hokkaido Inu • Symbol: HOKK • Decimals: 18 • Rewards Fee: 4% • Liquidity Fee: 1% • Marketing Fee: 3% • Maximum Sell Transaction Amount: 50,000,000 • Swap Tokens At Amount: 200,000 • Gas For Processing: 500,000	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it 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, 0 high, 0 medium and 14 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	Moderated
	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	Passed
	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	Moderated
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

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

The libraries in HOKK 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 HOKK Token.

The HOKK Token team has **not** provided scenario and 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 HOKK smart contract code in the form of a code form of a file. The hashes of that code are 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 Token.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

HOKK_ETH.sol

Functions

Sl.	Functions	Type	Observation	No Issue
1	constructor	read	Passed	No Issue
2	activate	write	access only Owner	No Issue
3	onlyBridge	modifier	Passed	No Issue
4	doConstructorStuff	external	Empty function defined	Refer audit findings section below
5	receive	external	Passed	No Issue
6	setAutomatedMarketMakerPair	write	access only Owner	No Issue
7	_setAutomatedMarketMakerPair	write	Passed	No Issue
8	excludeFromFees	write	access only Owner	No Issue
9	updateGasForTransfer	external	access only Owner	No Issue
10	allowTransferBeforewTradingIsEnabled	write	access only Owner	No Issue
11	updateGasForProcessing	write	access only Owner	No Issue
12	setBridgeAddresses	write	access only Owner	No Issue
13	swapAcrossChain	write	Passed	No Issue
14	portMessage	write	access only Bridge	No Issue
15	updateClaimWait	external	access only Owner	No Issue
16	getGasForTransfer	external	Passed	No Issue
17	enableDisableDevFee	write	Passed	No Issue
18	setMaxBuyEnabled	external	access only Owner	No Issue
19	getClaimWait	external	Passed	No Issue
20	getTotalDividendsDistributed	external	Passed	No Issue
21	isExcludedFromFees	write	Passed	No Issue
22	withdrawableDividendOf	read	Passed	No Issue
23	dividendTokenBalanceOf	read	Passed	No Issue
24	getAccountDividendsInfo	external	Passed	No Issue
25	getAccountDividendsInfoAtIndex	external	Passed	No Issue
26	processDividendTracker	external	Passed	No Issue
27	claim	external	Passed	No Issue
28	getLastProcessedIndex	external	Passed	No Issue
29	getNumberOfDividendTokenHolders	external	Passed	No Issue
30	_transfer	internal	False return functions	Refer audit findings section below

31	withdrawDividend	write	False return functions	Refer audit findings section below
32	swapAndSendToDev	write	Passed	No Issue
33	sendEthToDev	internal	Compile error	Refer audit findings section below
34	swapTokensForEth	write	Passed	No Issue
35	swapAndSendDividends	write	Passed	No Issue
36	name	read	Passed	No Issue
37	symbol	read	Passed	No Issue
38	decimals	read	Passed	No Issue
39	totalSupply	read	Passed	No Issue
40	balanceOf	read	Passed	No Issue
41	transfer	write	Passed	No Issue
42	allowance	read	Passed	No Issue
43	approve	write	Passed	No Issue
44	transferFrom	write	Passed	No Issue
45	increaseAllowance	write	Passed	No Issue
46	decreaseAllowance	write	Passed	No Issue
47	_transfer	internal	Transfer 0 amount	Refer audit findings section below
48	_mint	internal	Passed	No Issue
49	_burn	internal	Passed	No Issue
50	_approve	internal	Passed	No Issue
51	_beforeTokenTransfer	internal	Passed	No Issue
52	owner	read	Passed	No Issue
53	onlyOwner	modifier	Passed	No Issue
54	renounceOwnership	write	access only Owner	No Issue
55	transferOwnership	write	access only Owner	No Issue

HOKK_BEP.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	onlyBridge	modifier	Wrong validation	Refer audit findings section below
3	_transfer	internal	Passed	No Issue
4	withdrawDividend	write	Passed	No Issue
5	excludeFromDividends	external	access only Owner	No Issue
6	updateClaimWait	external	access only Owner	No Issue
7	getLastProcessedIndex	external	Passed	No Issue
8	getNumberOfTokenHolders	external	Passed	No Issue

9	getAccount	write	Passed	No Issue
10	getAccountAtIndex	write	Passed	No Issue
11	canAutoClaim	read	Passed	No Issue
12	setBalance	external	access only Owner	No Issue
13	process	write	Passed	No Issue
14	processAccount	write	access only Owner	No Issue
15	updateDividendTracker	write	access only Owner	No Issue
16	receive	external	Passed	No Issue
17	updateUniswapV2Router	write	access only Owner	No Issue
18	setMarketingWallet	write	access only Owner	No Issue
19	excludeFromFees	write	access only Owner	No Issue
20	excludeMultipleAccounts FromFees	write	access only Owner	No Issue
21	setAutomatedMarketMaker Pair	write	access only Owner	No Issue
22	_setAutomatedMarketMa kerPair	write	Passed	No Issue
23	setBridgeAddresses	write	access only Owner	No Issue
24	swapAcrossChain	write	Passed	No Issue
25	portMessage	write	access only Bridge	No Issue
26	updateLiquidityWallet	write	access only Owner	No Issue
27	updateGasForProcessin g	write	access only Owner	No Issue
28	updateClaimWait	external	access only Owner	No Issue
29	getClaimWait	external	Passed	No Issue
30	getTotalDividendsDistribu ted	external	Passed	No Issue
31	isExcludedFromFees	read	Passed	No Issue
32	withdrawableDividendOf	read	Passed	No Issue
33	dividendTokenBalanceOf	read	Passed	No Issue
34	getAccountDividendsInfo	external	Passed	No Issue
35	getAccountDividendsInfo AtIndex	external	Passed	No Issue
36	processDividendTracker	external	Passed	No Issue
37	claim	external	Passed	No Issue
38	getMarketingFeePercent	read	Passed	No Issue
39	getLastProcessedIndex	external	Passed	No Issue
40	getNumberOfDividendTo kenHolders	external	Passed	No Issue
41	_transfer	internal	Passed	No Issue
42	swapAndLiquify	write	Passed	No Issue
43	swapTokensForEth	write	Passed	No Issue
44	addLiquidity	write	Passed	No Issue
45	sendBNBToMarketing	write	Passed	No Issue
46	swapAndSendDividends	write	Passed	No Issue

HOKK_HECO.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	onlyBridge	modifier	Wrong validation	Refer audit findings section below
3	receive	external	Passed	No Issue
4	updateDividendTracker	write	access only Owner	No Issue
5	setMarketingWallet	write	access only Owner	No Issue
6	excludeFromFees	write	access only Owner	No Issue
7	excludeMultipleAccounts FromFees	write	Infinite loop possibility	Refer audit findings section below
8	setAutomatedMarketMak erPair	write	access only Owner	No Issue
9	_setAutomatedMarketMa kerPair	write	Passed	No Issue
10	updateLiquidityWallet	write	access only Owner	No Issue
11	updateGasForProcessin g	write	access only Owner	No Issue
12	updateClaimWait	external	access only Owner	No Issue
13	getClaimWait	external	Passed	No Issue
14	getTotalDividendsDistribu ted	external	Passed	No Issue
15	isExcludedFromFees	read	Passed	No Issue
16	withdrawableDividendOf	read	Passed	No Issue
17	dividendTokenBalanceOf	read	Passed	No Issue
18	getAccountDividendsInfo	external	Passed	No Issue
19	getAccountDividendsInfo AtIndex	external	Passed	No Issue
20	processDividendTracker	external	Passed	No Issue
21	claim	external	Passed	No Issue
22	getMarketingFeePercent	read	Passed	No Issue
23	getLastProcessedIndex	external	Passed	No Issue
24	getNumberOfDividendTo kenHolders	external	Passed	No Issue
25	setBridgeAddresses	write	access only Owner	No Issue
26	swapAcrossChain	write	Passed	No Issue
27	portMessage	write	access only Bridge	No Issue
28	_transfer	internal	Passed	No Issue
29	swapAndLiquify	write	Passed	No Issue
30	swapTokensForEth	write	Passed	No Issue
31	addLiquidity	write	Centralized risk in addLiquidity	Refer audit findings section below
32	sendHTToMarketing	write	Passed	No Issue

33	swapAndSendDividends	write	Passed	No Issue
34	_transfer	internal	False return functions	Refer audit findings section below
35	withdrawDividend	write	False return functions	Refer audit findings section below
36	excludeFromDividends	external	access only Owner	No Issue
37	updateClaimWait	external	access only Owner	No Issue
38	getLastProcessedIndex	external	Passed	No Issue
39	getNumberOfTokenHolders	external	Passed	No Issue
40	getAccount	write	Passed	No Issue
41	getAccountAtIndex	read	Passed	No Issue
42	canAutoClaim	read	Passed	No Issue
43	process	write	Passed	No Issue
44	setBalance	external	access only Owner	No Issue
45	processAccount	write	access only Owner	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.

Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

File : HOKK_ETH.sol

(1) Transferred 0 amount:

Transfers 0 amounts.

```
if (amount == 0) {
    super._transfer(from, to, 0);
    return;
}
```

Resolution: We suggest avoiding 0 amounts to get transferred.

(2) Compile error:

Exactly one argument expected for explicit type conversion.

```
function sendEthToDev(uint256 amount) private {
    address payable _DAOAddress = payable(); //CHANGE
    address payable _marketingAddress = payable(); //CHANGE

    _DAOAddress.transfer(amount.div(2));
    _marketingAddress.transfer(amount.div(2));
}
```

Resolution: Add address to avoid this error

(3) Commented code in if condition:

Only commented code is there inside the if condition.

Resolution: We suggest removing that condition to reduce gas fees.

(4) Critical operation lacks event log:

Missing event log for :

- portMessage
- sendEthToDev
- claim
- swapAcrossChain.

Resolution: Please write an event log for listed events.

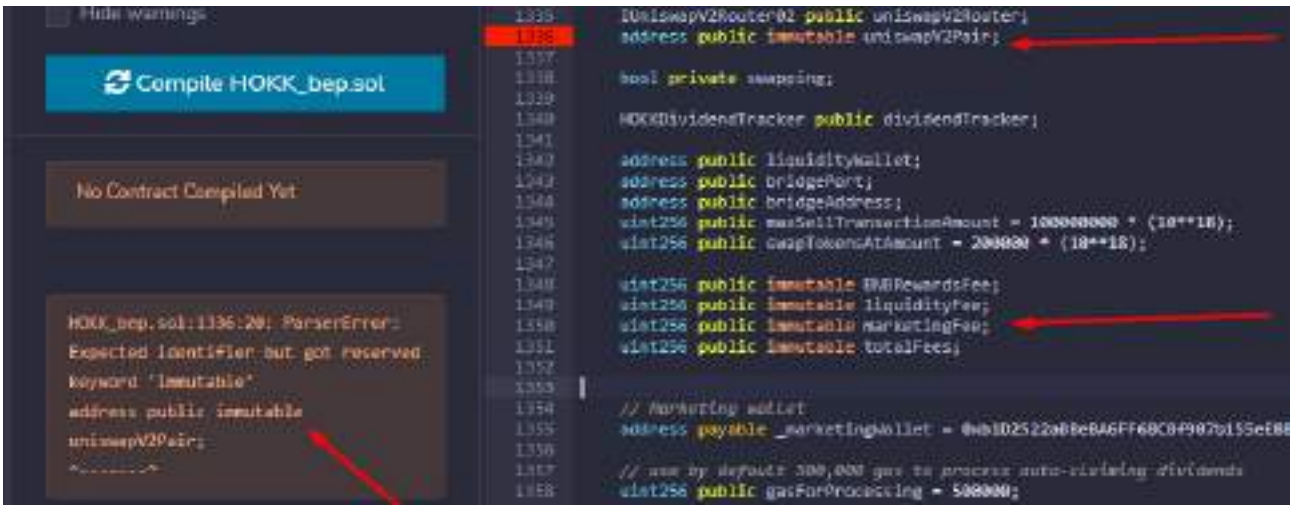
FILE : HOKK_BEP.sol

(1) Definition of base has to precede definition of derived contract:

Resolution: Define Context contract before Ownable and IERC20 Interface before IERC20Metadata.

(2) Expected identifier but got reserved keyword for solidity compiler version older than 0.6.9:

Expected identifier but got reserved keyword 'immutable'.



Resolution: Remove "Immutable" keyword

(3) Data location must be "storage" or "memory" for solidity compiler version older than 0.6.9:

Data location must be "storage" or "memory" for return parameter in function, but "calldata" was given.

Resolution: Instead of "calldata" add "memory" type.

(4) Centralized risk in addLiquidity:
In addLiquidityETH function, liquidity wallet gets Tokens from the Pool. If the private key of the liquidity wallet is compromised, then it will create a problem.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {

    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityWallet,
        block.timestamp
    );

}
```

Resolution: Ideally this can be a governance smart contract. On another hand, the liquidity can accept this risk and handle the private key very securely.

(5) Critical operation lacks event log:

Missing event log for :

- portMessage
- sendBNBToMarketing
- claim
- swapAcrossChain.

Resolution: Please write an event log for listed events.

(6) Function input parameters lack of check :

Variable validation is not performed in below functions :

- setMarketingWallet = wallet
- etBridgeAddresses = _port & _wallet
- withdrawableDividendOf = account

Resolution: Use validation : variable is not empty and should be greater than 0 and for address type check variable is not address(0).

(7) Wrong validation:

In onlyBridge modifier, the caller is validated for bridgePort.

```
modifier onlyBridge {
    require(msg.sender == bridgePort);
    _;
}
```

Resolution: Instead of bridgePort, it should be validated as bridgeAddress.

FILE : HOKK_HECO.sol

(1) Wrong Marketing wallet address:
As per the document, the marketing wallet is different. So the deployer has to change the variable value before deploying the contract.

```
    // Marketing wallet
    address payable _marketingWallet = 0xa4fDaEA89c192e220754EDaecB80fB8e48024a57;
```

Resolution: This variable can be updated by the contract owner.

(2) Infinite loop possibility:

In the excludeMultipleAccountsFromFees function, the for loop does not have an accountIds length limit , which costs more gas.

```
function excludeMultipleAccountsFromFees(address[] memory accounts, bool excluded) public onlyOwner {
    for(uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

Resolution: Accounts limit should be limited in for loops.

(3) Critical operation lacks event log:

Missing event log for :

- portMessage
- sendHTToMarketing
- claim
- swapAcrossChain.

Resolution: Please write an event log for listed events.

(4) Centralized risk in addLiquidity:

In addLiquidityETH function, liquidity wallet gets Tokens from the Pool. If the private key of the liquidity wallet is compromised, then it will create a problem.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {

    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(Router), tokenAmount);

    // add the liquidity
    Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityWallet,
        block.timestamp
    );

}
```

Resolution: Ideally this can be a governance smart contract. On another hand, the liquidity can accept this risk and handle the private key very securely.

(7) Wrong validation:

In onlyBridge modifier, the caller is validated for bridgePort.

```
modifier onlyBridge {
    require(msg.sender == bridgePort);
    _;
}
```

Resolution: Instead of bridgePort, it should be validated as bridgeAddress.

Very Low / Informational / Best practices:

File : HOKK_ETH.sol

(1) Unused variable:

The variable has not been used anywhere.

```
address payable private _devWallet;
```

Resolution: We suggest removing unused variable.

(2) Make variable constant:

This variable will be unchanged. So, please make it constant. It will save some gas.

```
// liquidate tokens for ETH when the contract reaches 100k tokens by default
uint256 public liquidateTokensAtAmount = 100000 * (10**18); //CHANGE
```

Resolution: Declare this variable as constant. Just put a constant keyword.

(3) Empty function defined:

Empty function defined in contract.

```
function doConstructorStuff() external onlyOwner{

}
```

Resolution: We suggest to remove empty function.

(4) False return functions:

These functions will always return 0 or False.

```
function _transfer(address, address, uint256) internal pure override {
    require(false, "HOKK_Dividend_Tracker: No transfers allowed");
}

function withdrawDividend() public pure override {
    require(false, "HOKK_Dividend_Tracker: withdrawDividend disabled. Use the 'claim' function on the main HOKK contract.");
}
```

Resolution: We suggest removing this kind of function.

(5) Irrelevant variable name:

Here marketing fee is defined as Liquidity fee.

```
    uint256 public constant ETH_REWARDS_FEE = 4
    uint256 public constant LIQUIDITY_FEE = 4;
    uint256 public constant TOTAL_FEES = ETH_RE
```

Resolution: The variable name should be marketing_fee.

FILE : HOKK_BEP.sol

(1) Multiple pragma added:

There are multiple pragma added in the code.

Resolution: We suggest keeping only pragma.

(2) Use the latest solidity version:

Using the latest solidity will prevent any compiler-level bugs.

```
pragma solidity ^0.6.2;
```

Resolution: Please use 0.8.9 which is the latest version.

(3) SPDX license identifier is missing:

SPDX license identifier not provided in source file.

Resolution: We suggest adding SPDX license identifier.

(4) Missing error message:

Require without error message

```
function distributeDividends() public override payable {
  require(totalSupply() > 0);
```

```
function excludeFromDividends(address account) external onlyOwner {
    require(!excludedFromDividends[account]);
    excludedFromDividends[account] = true;
```

.**Resolution:** Write appropriate error message.

FILE : HOKK_HECO.sol

(1) Multiple pragma added:

There are multiple pragma added in the code.

Resolution: We suggest keeping only pragma.

(2) Use the latest solidity version:

Using the latest solidity will prevent any compiler-level bugs.

```
pragma solidity ^0.6.2;
```

Resolution: Please use 0.8.9 which is the latest version.

(3) SPDX license identifier is missing:

SPDX license identifier not provided in source file.

Resolution: We suggest adding SPDX license identifier.

(4) Make variable constant:

This variable will be unchanged. So, please make it constant. It will save some gas.

```
uint256 public maxSellTransactionAmount = 50000000 * (10**18);
uint256 public swapTokensAtAmount = 200000 * (10**18);
```

Resolution: Declare this variable as constant. Just put a constant keyword.

(5) False return functions:

These functions will always return 0 or False.

```
function _transfer(address, address, uint256) internal pure override {
    require(false, "HOKK_Dividend_Tracker: No transfers allowed");
}

function withdrawDividend() public pure override {
    require(false, "HOKK_Dividend_Tracker: withdrawDividend disabled. Use the 'claim' function on the main HOKK contract.");
}
```

Resolution: We suggest removing this kind of function.

Centralization

These smart contracts have 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:

- activate: The Owner can activate the account address.
- doConstructorStuff: The Owner can do constructor stuff.
- setAutomatedMarketMakerPair: The Owner can set automated marketing marker pairs.
- excludeFromFees: The owner can exclude from fees.
- updateGasForTransfer: The Owner can update gas for transfer.
- allowTransferBeforeTradingIsEnabled: The Owner can allow transfer before trading is enabled or not.
- updateGasForProcessing: The Owner can update gas for processing.
- setBridgeAddresses: The Owner can set bridge addresses.
- portMessage: The Owner can port messages.
- updateClaimWait: The Owner can update the claim wait process.
- setMaxBuyEnabled: The Owner can set maximum buy enabled.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are not critical ones. So, **it's good to go to production**.

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 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.

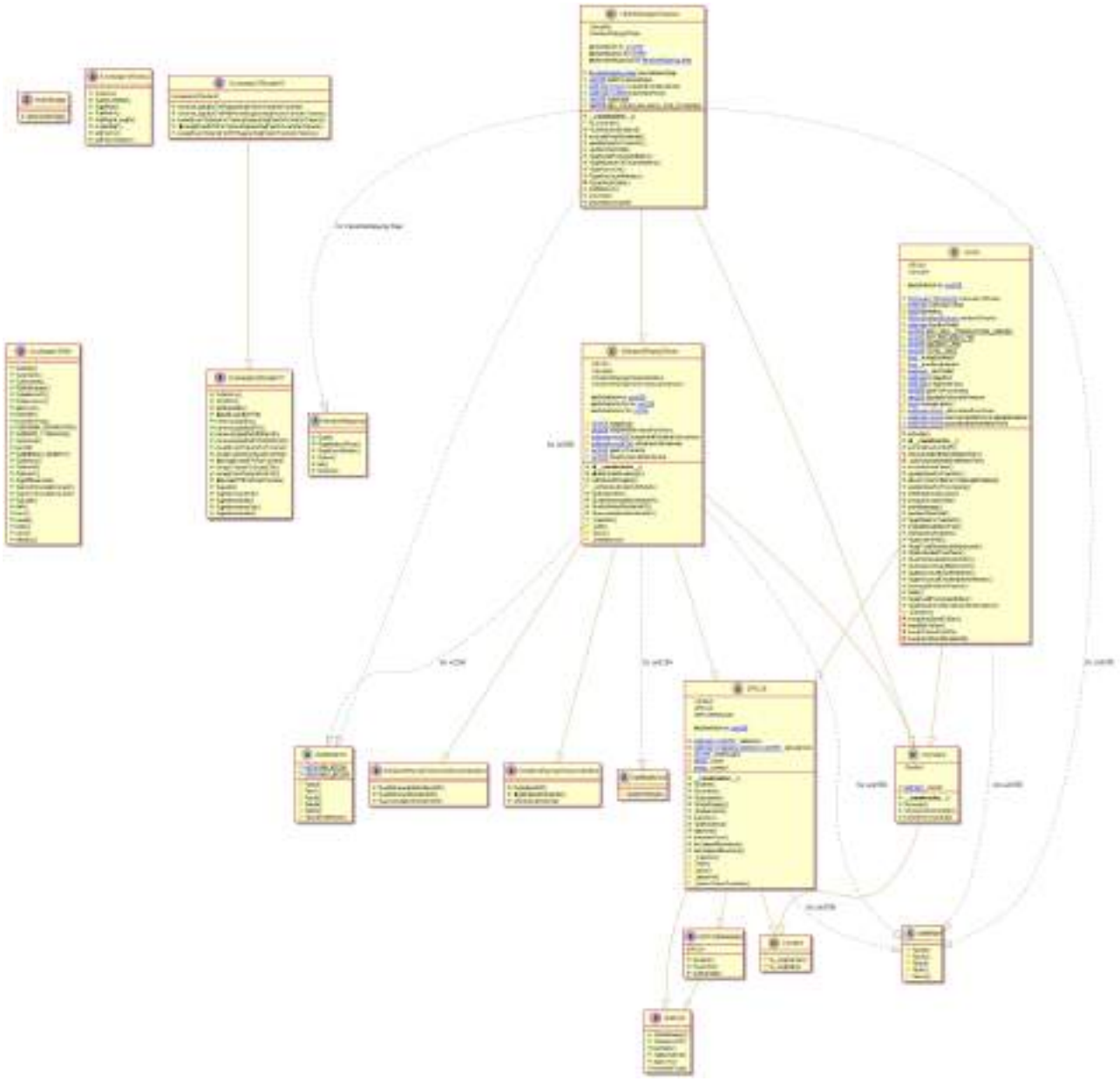
Technical Disclaimer

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.

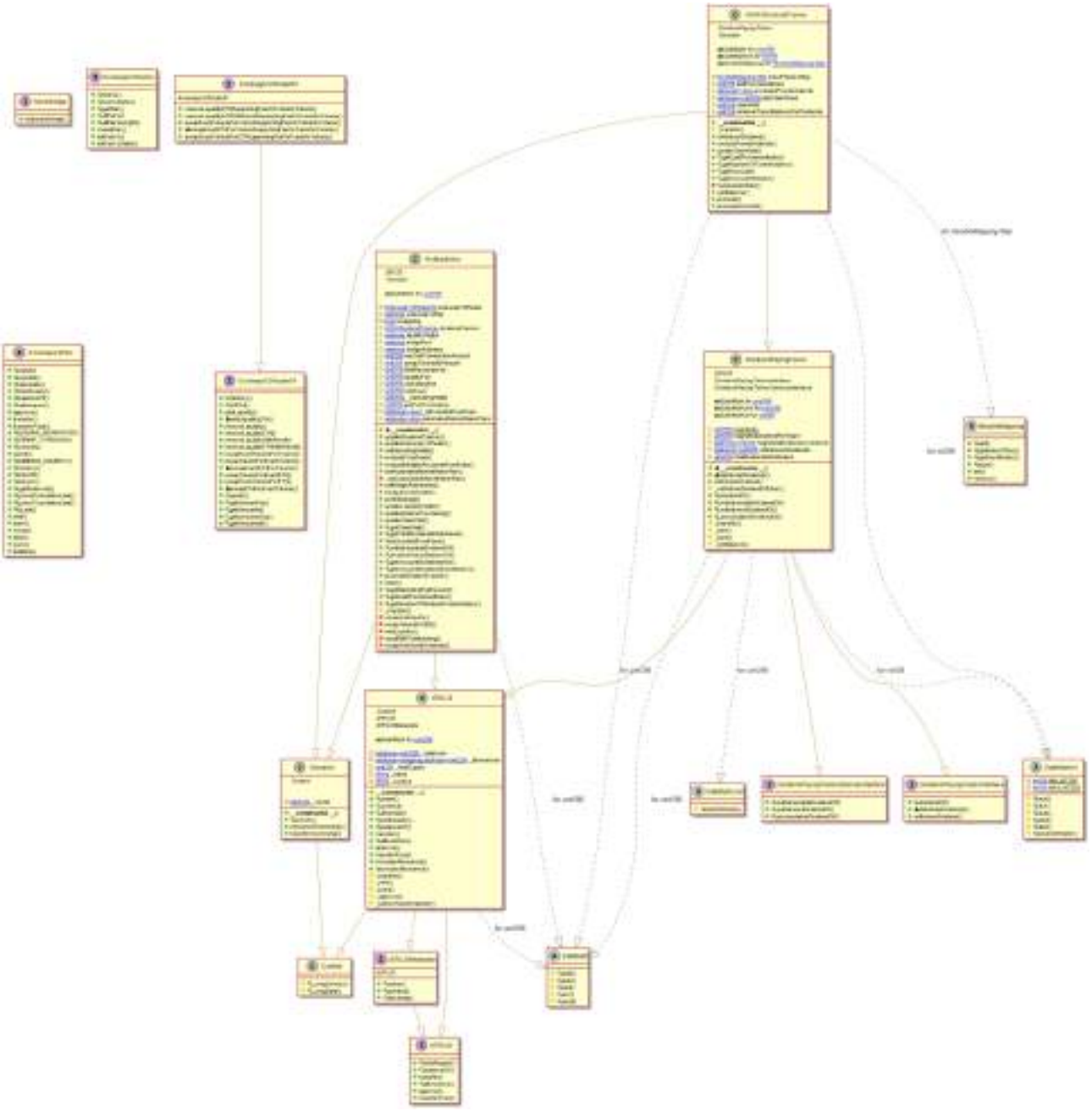
Appendix

Code Flow Diagram - HOKK Token

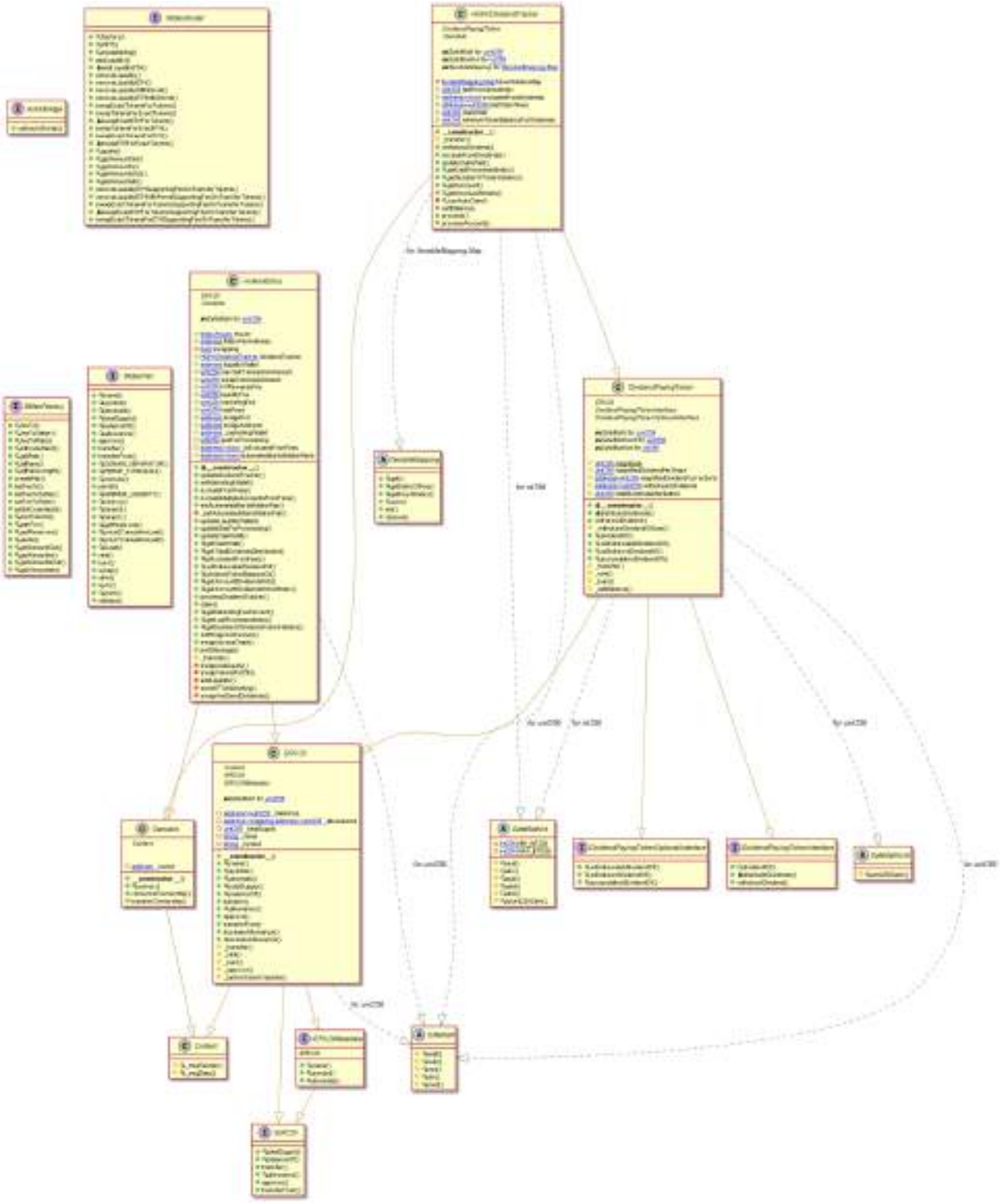
HOKK-ETH Diagram



HOKK-BEP Diagram



HOKK-HECO Diagram

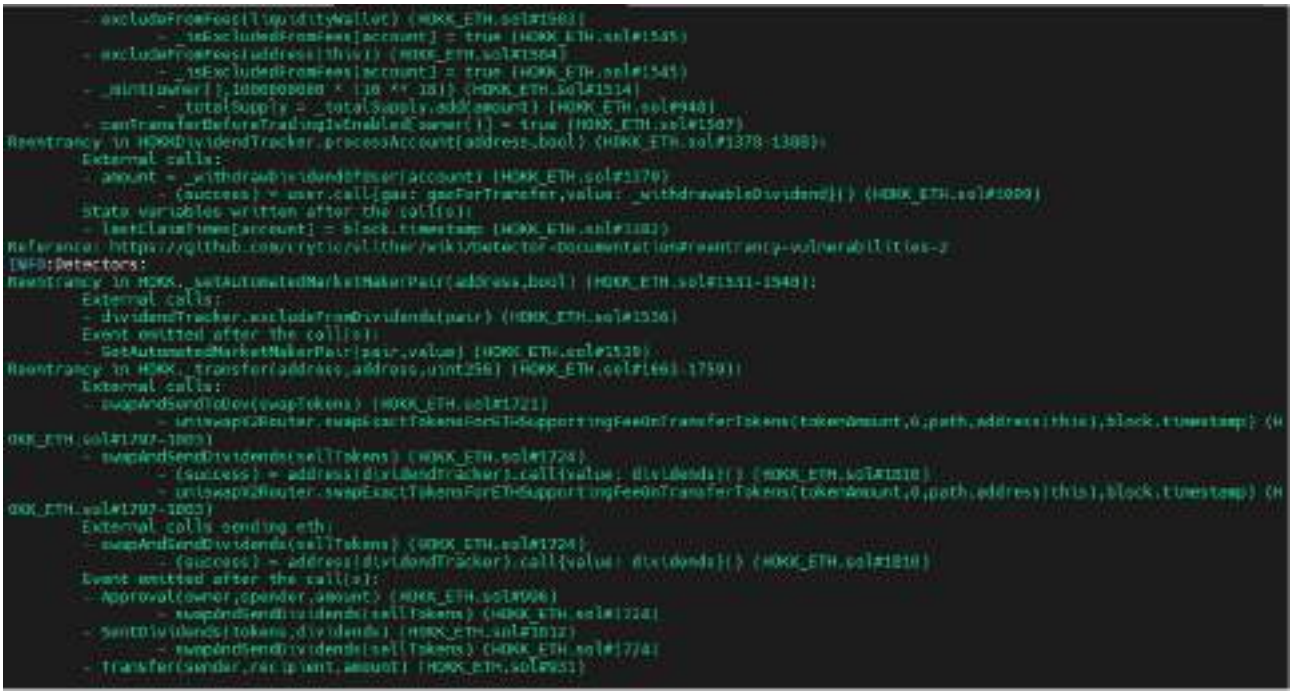


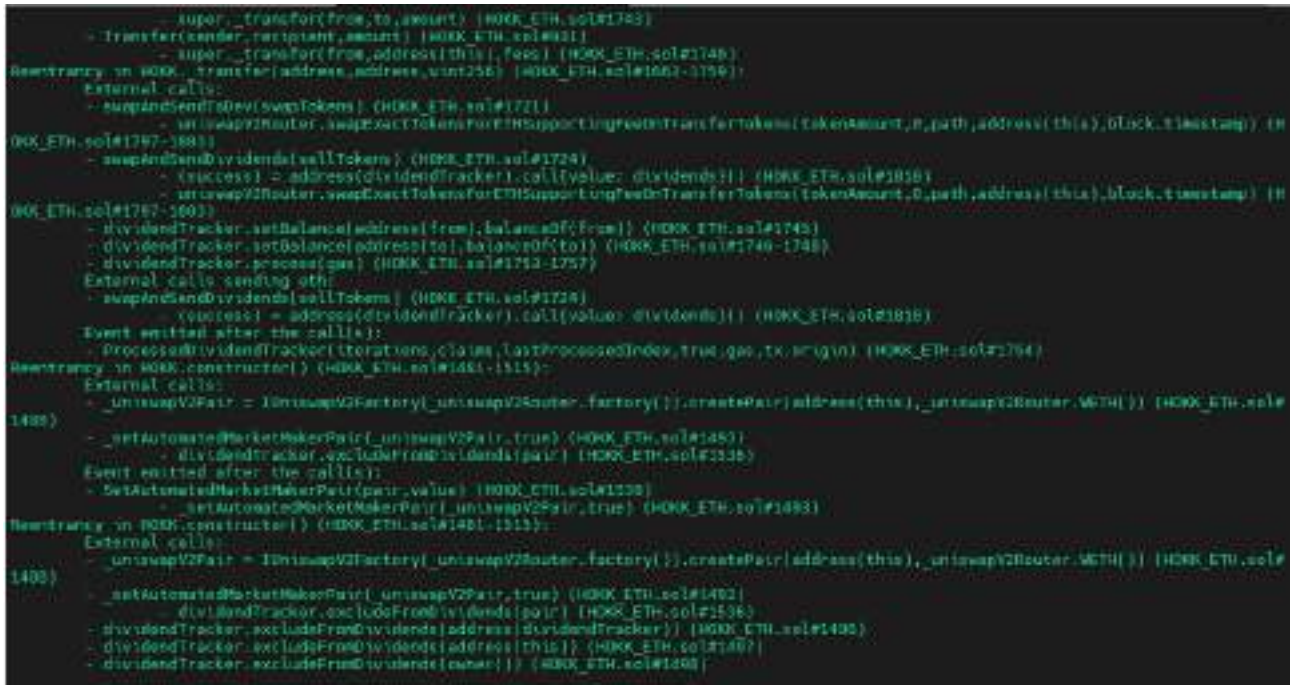
Slither Results Log

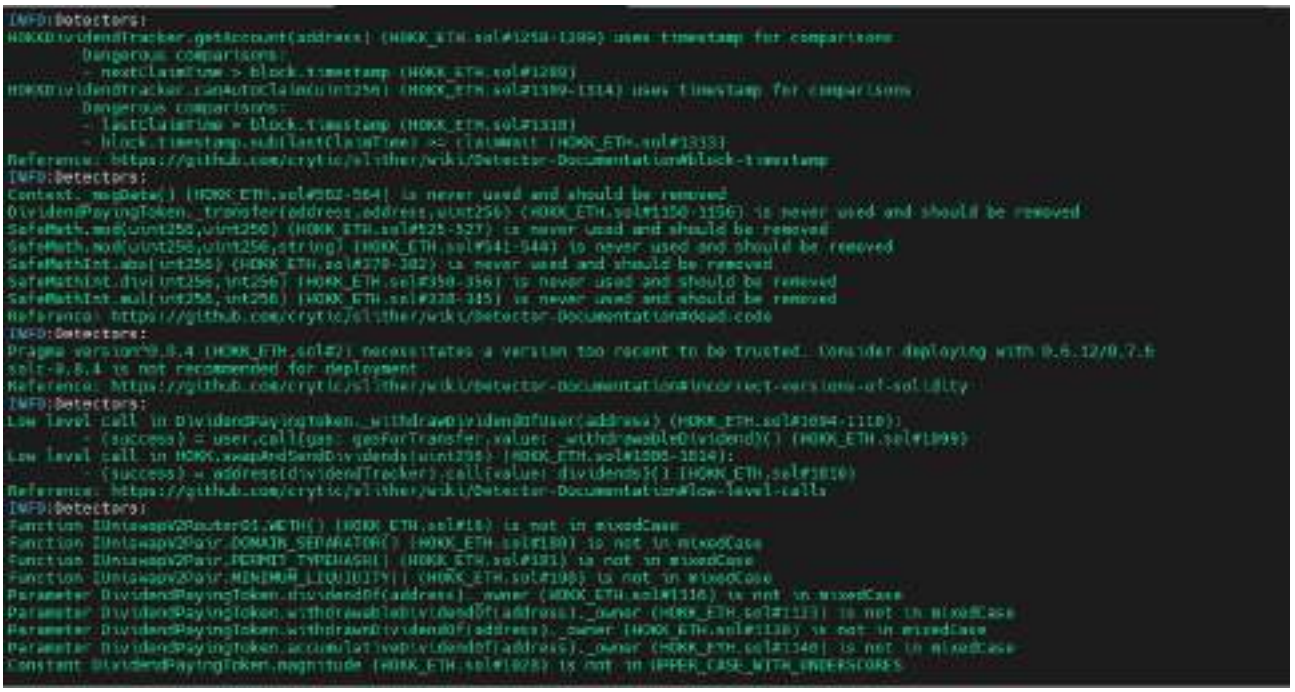
Slither log >> HOKK-ETH.sol

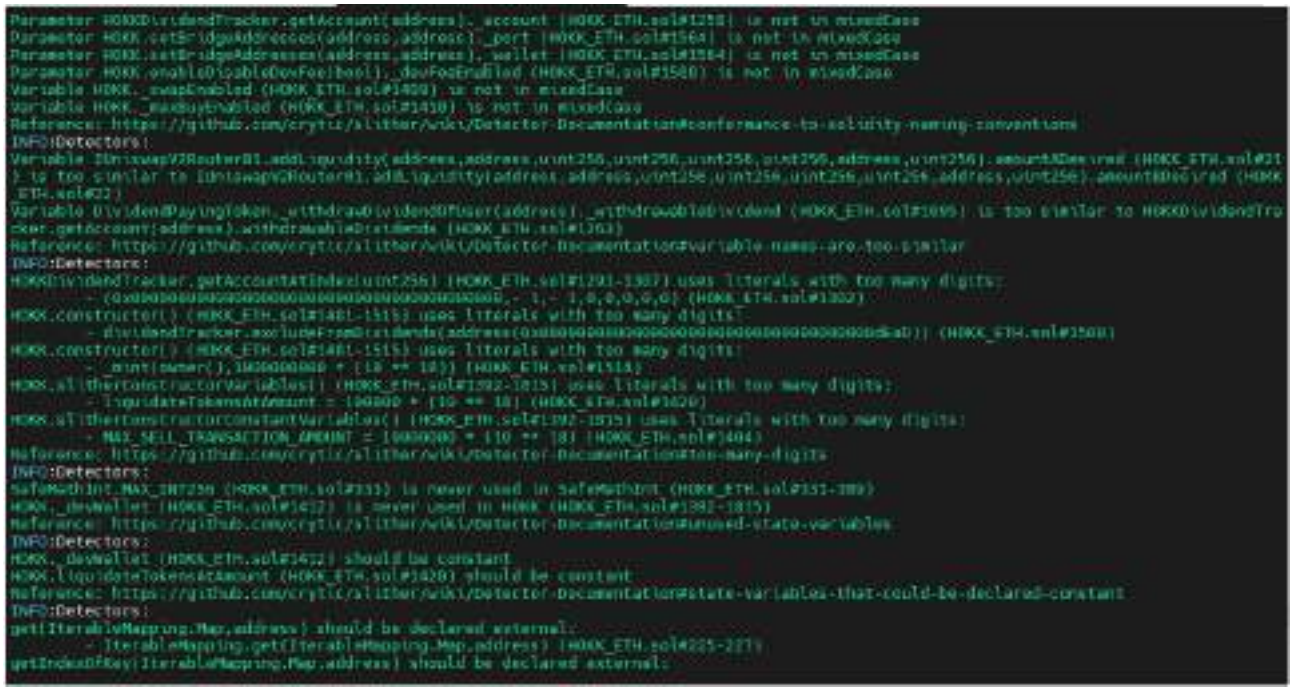
```
INFO:Detectors:
Reentrancy in HOKK._transfer(address,address,uint256) (HOKK_ETH.sol#1661-1759):
        External calls:
        - swapAndSendTobev(swapTokens) (HOKK_ETH.sol#1721)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (H
OKK_ETH.sol#1797-1803)
        - swapAndSendDividends(sellTokens) (HOKK_ETH.sol#1724)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_ETH.sol#1810)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (H
OKK_ETH.sol#1797-1803)
        External calls sending eth:
        - swapAndSendDividends(sellTokens) (HOKK_ETH.sol#1724)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_ETH.sol#1810)
        State variables written after the call(s):
        - super._transfer(from,address(this),fees) (HOKK_ETH.sol#1748)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (HOKK_ETH.sol#929)
                - _balances[recipient] = _balances[recipient].add(amount) (HOKK_ETH.sol#930)
        - super._transfer(from,to,amount) (HOKK_ETH.sol#1742)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (HOKK_ETH.sol#929)
                - _balances[recipient] = _balances[recipient].add(amount) (HOKK_ETH.sol#930)
        - liquidating = false (HOKK_ETH.sol#1726)
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (HOKK_ETH.sol#1804-1818):
        External calls:
        - (success) = user.call{gas: gasForTransfer,value: _withdrawableDividend}() (HOKK_ETH.sol#1809)
        State variables written after the call(s):
        - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend) (HOKK_ETH.sol#1812)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
HOKK._transfer(address,address,uint256).claims (HOKK_ETH.sol#1753) is a local variable never initialized
HOKK._transfer(address,address,uint256).iterations (HOKK_ETH.sol#1755) is a local variable never initialized
HOKK._transfer(address,address,uint256).lastProcessedIndex (HOKK_ETH.sol#1753) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
HOKK.claim() (HOKK_ETH.sol#1850-1852) ignores return value by dividendTracker.processAccount(address(msg.sender),false) (HOKK_ETH.sol#185
```

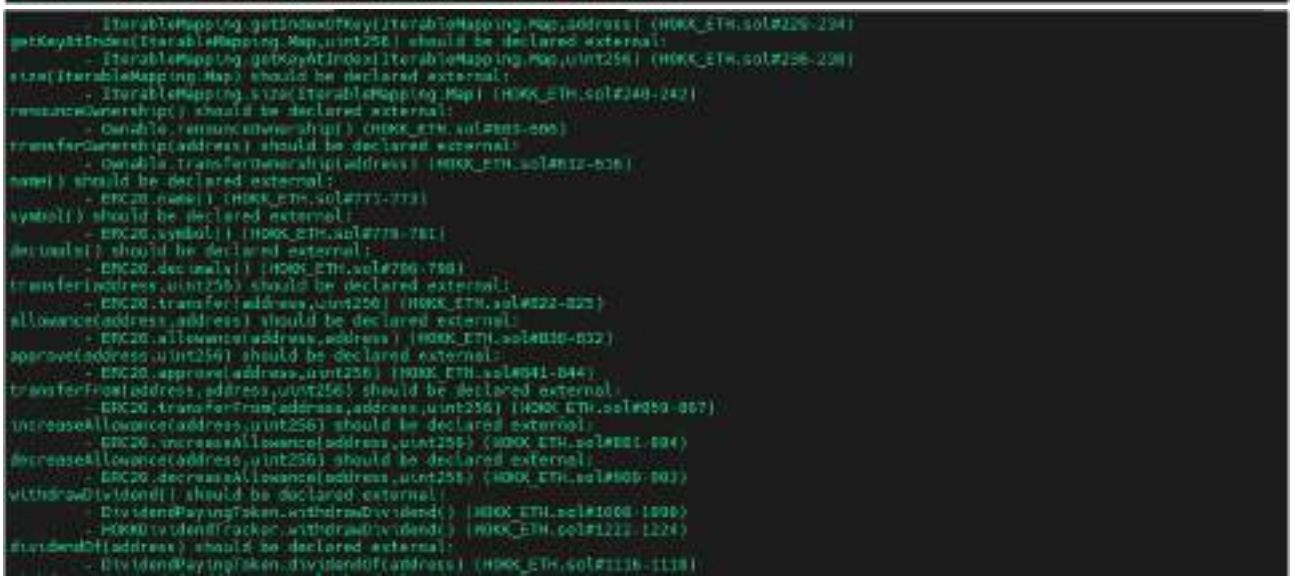
```
        - swapAndSendTobev(swapTokens) (HOKK_ETH.sol#1721)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (H
OKK_ETH.sol#1797-1803)
        - swapAndSendDividends(sellTokens) (HOKK_ETH.sol#1724)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_ETH.sol#1810)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (H
OKK_ETH.sol#1797-1803)
        External calls sending eth:
        - swapAndSendDividends(sellTokens) (HOKK_ETH.sol#1724)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_ETH.sol#1810)
        State variables written after the call(s):
        - swapAndSendDividends(sellTokens) (HOKK_ETH.sol#1724)
                - _allowances[owner][spender] = amount (HOKK_ETH.sol#905)
Reentrancy in HOKK.constructor() (HOKK_ETH.sol#1481-1515):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (HOKK_ETH.sol#
1489)
        State variables written after the call(s):
        - uniswapV2Pair = _uniswapV2Pair (HOKK_ETH.sol#1491)
        - uniswapV2Router = _uniswapV2Router (HOKK_ETH.sol#1490)
Reentrancy in HOKK.constructor() (HOKK_ETH.sol#1481-1515):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (HOKK_ETH.sol#
1489)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (HOKK_ETH.sol#1493)
                - dividendTracker.excludeFromDividends(pair) (HOKK_ETH.sol#1535)
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (HOKK_ETH.sol#1495)
        - dividendTracker.excludeFromDividends(address(this)) (HOKK_ETH.sol#1497)
        - dividendTracker.excludeFromDividends(owner()) (HOKK_ETH.sol#1498)
        - dividendTracker.excludeFromDividends(address(_uniswapV2Router)) (HOKK_ETH.sol#1499)
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (HOKK_ETH.sol#1500)
        State variables written after the call(s):
        - _mint(owner(),5000000000 * (10 ** 18)) (HOKK_ETH.sol#1514)
                - _balances[account] = _balances[account].add(amount) (HOKK_ETH.sol#904)
```

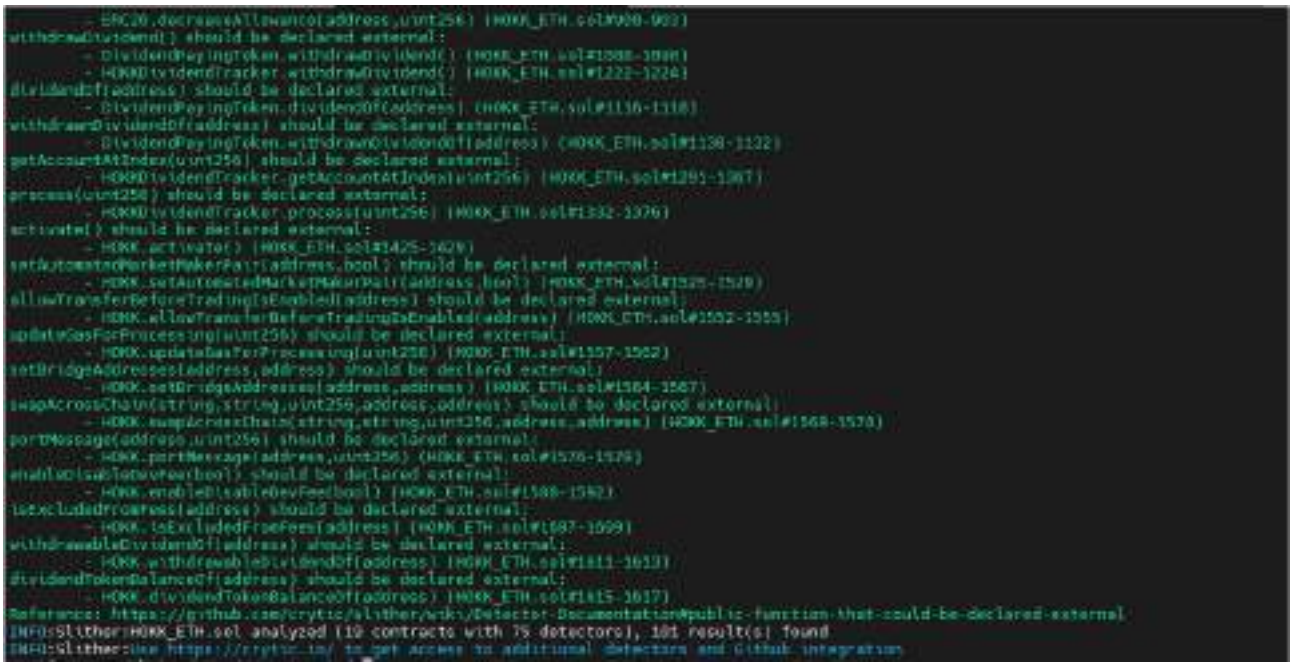




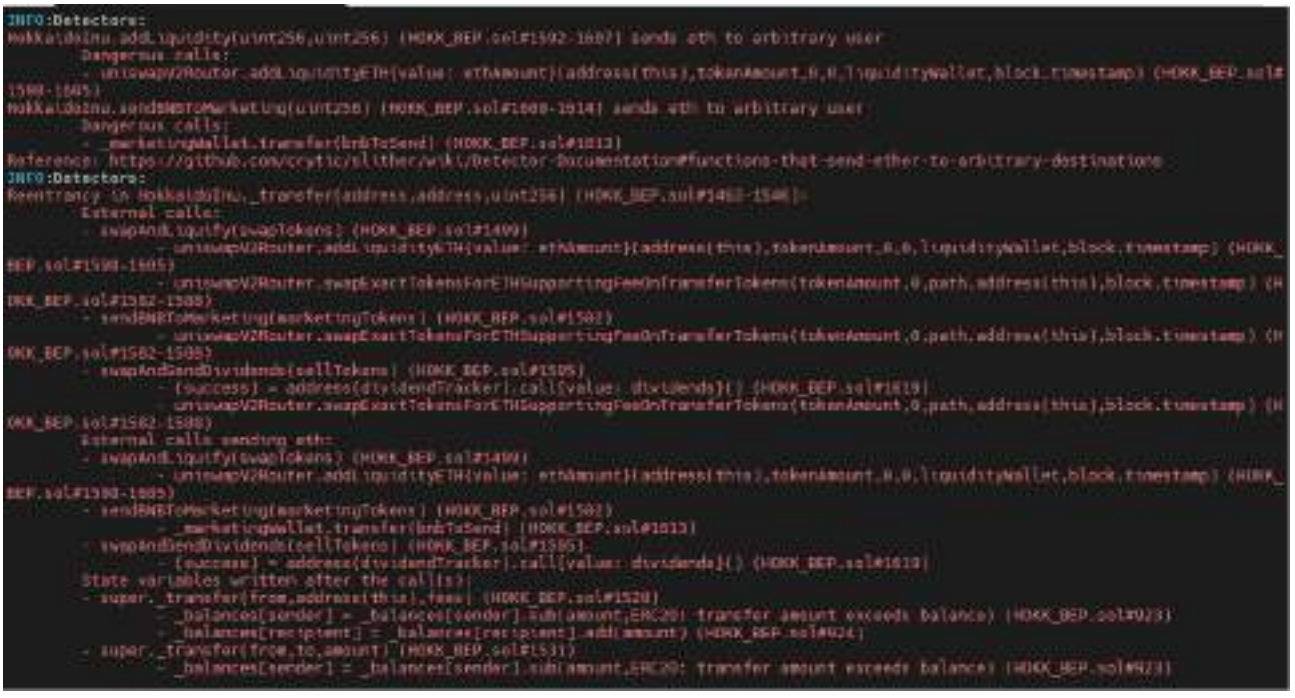


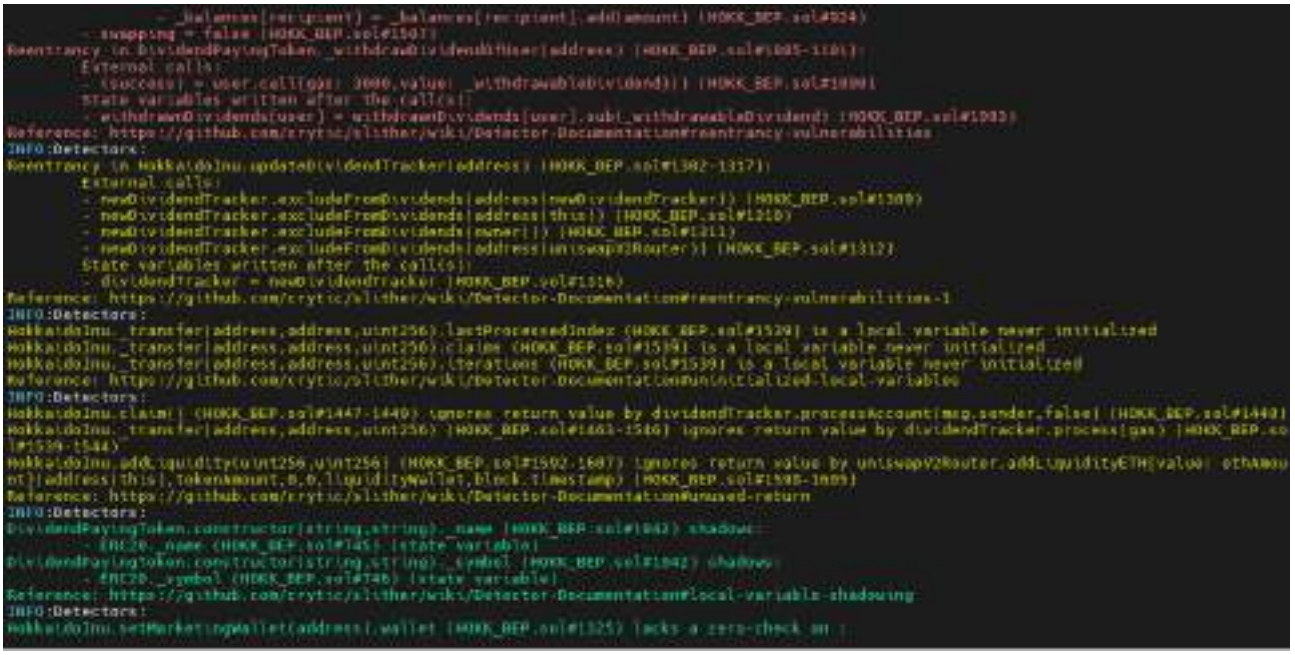


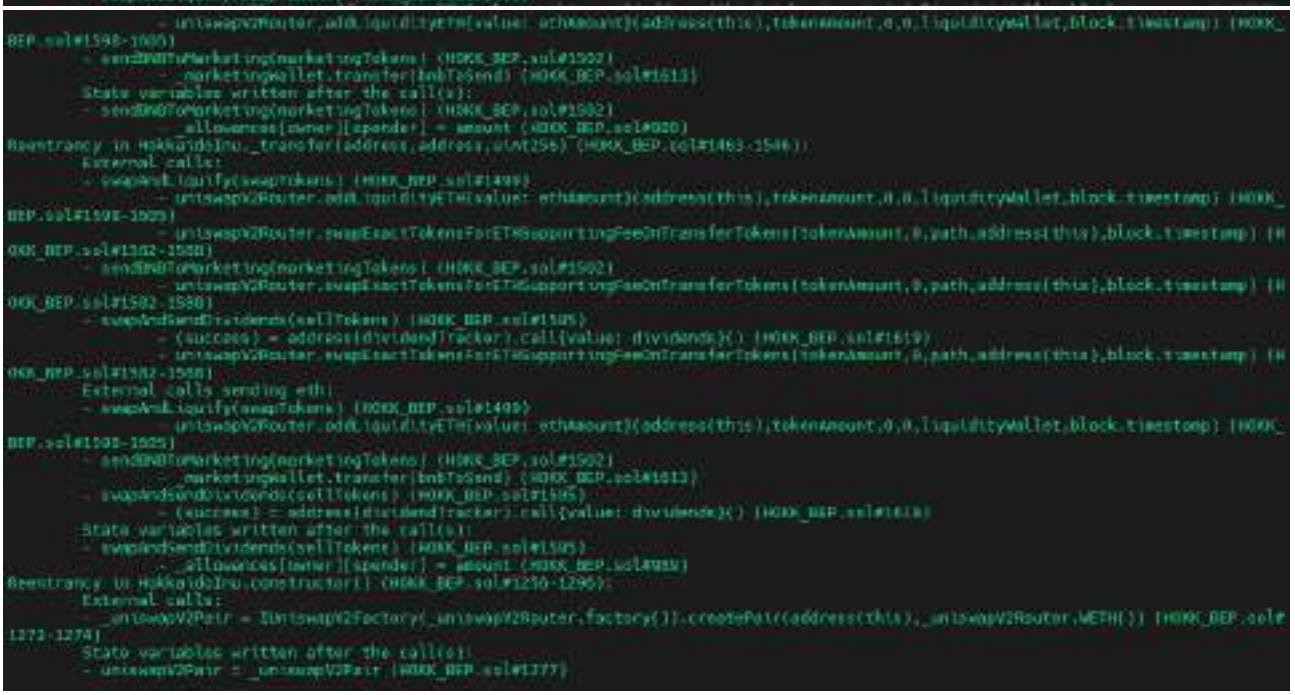


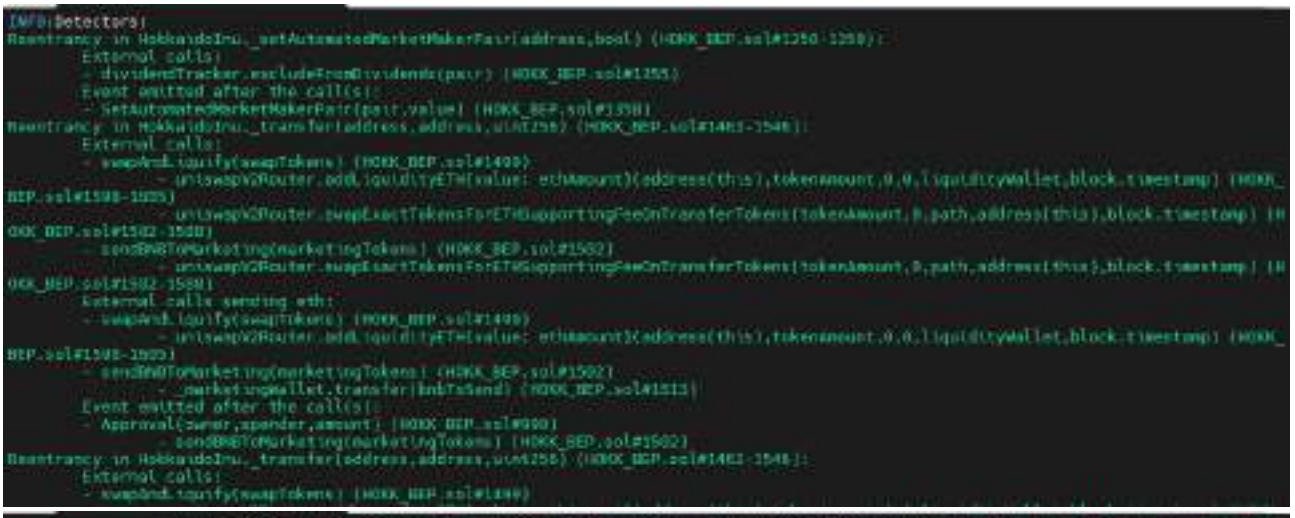


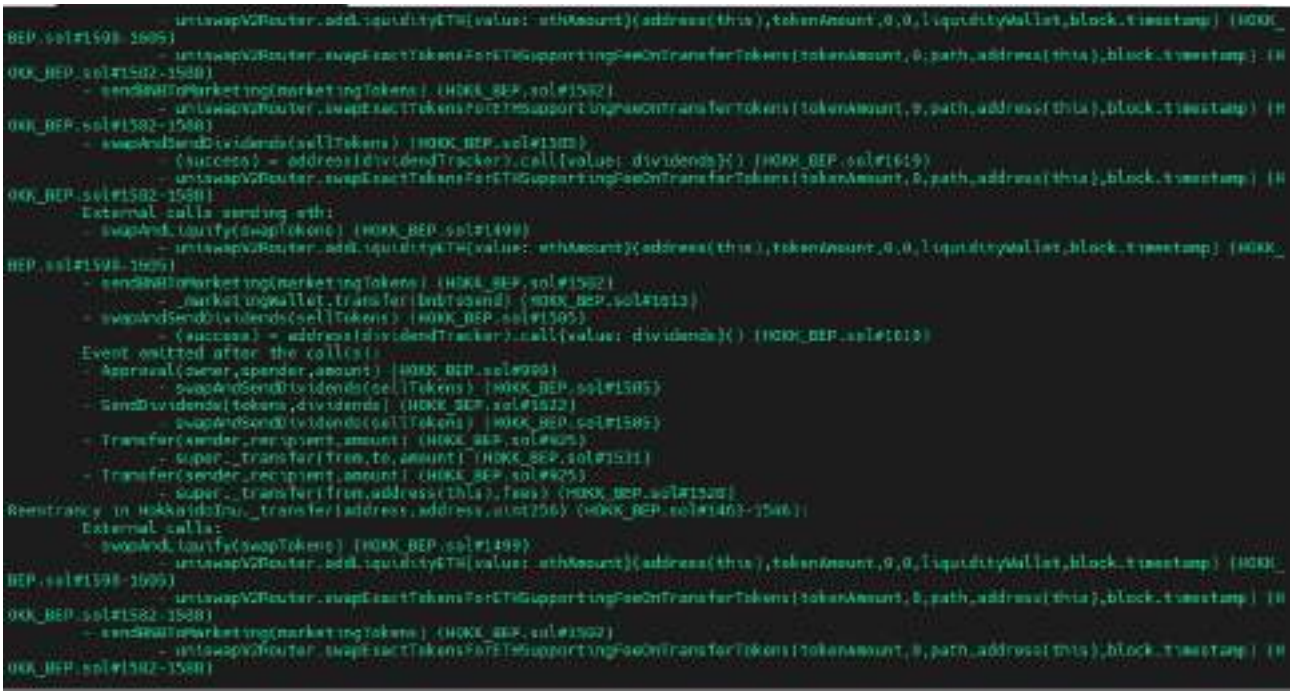
Slither log >> HOKK-BEP.sol

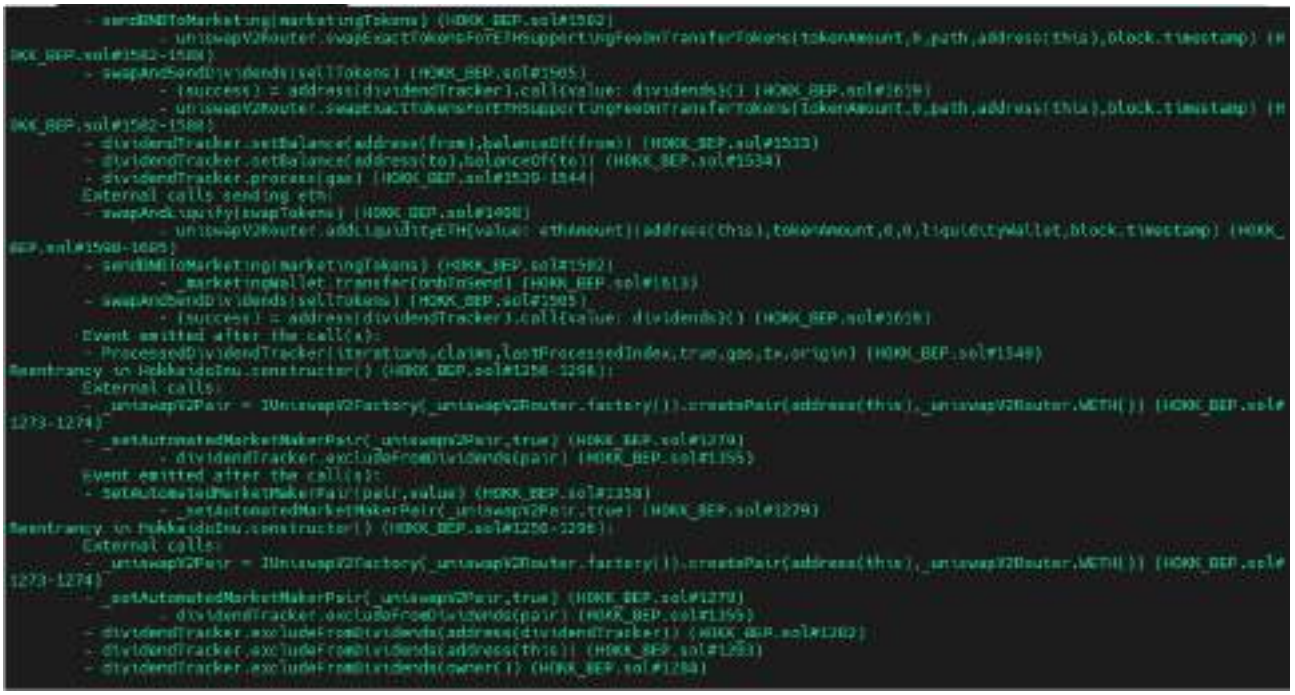


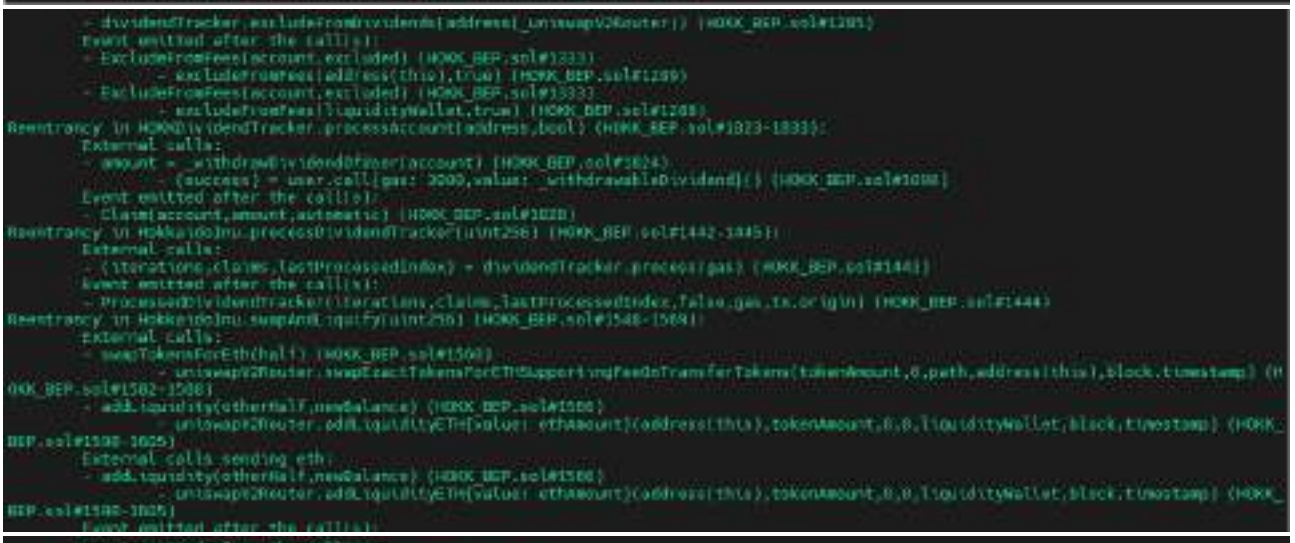


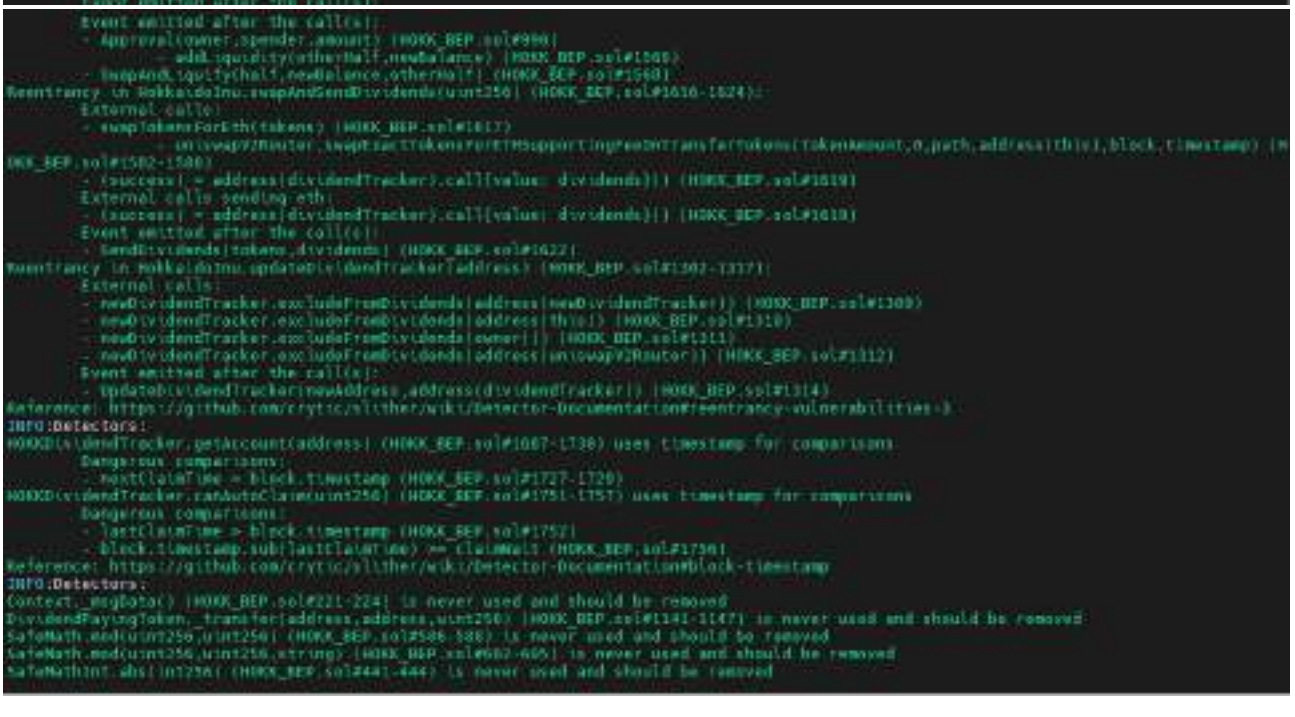


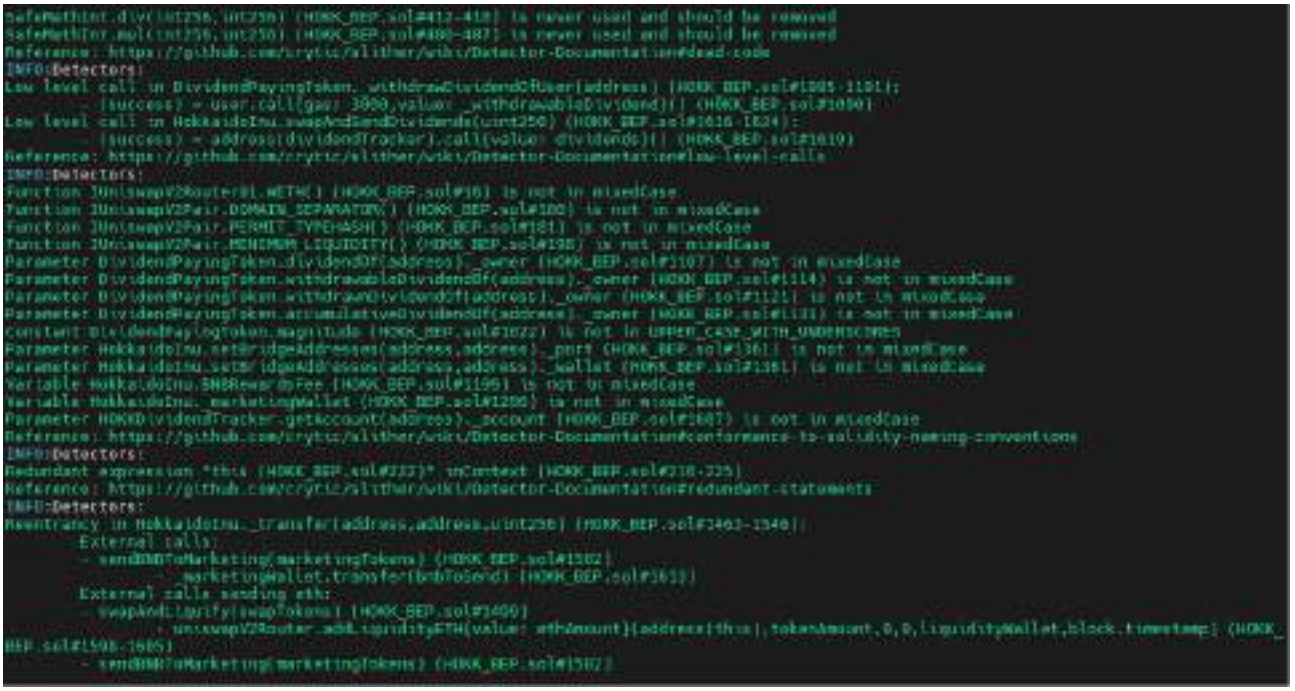










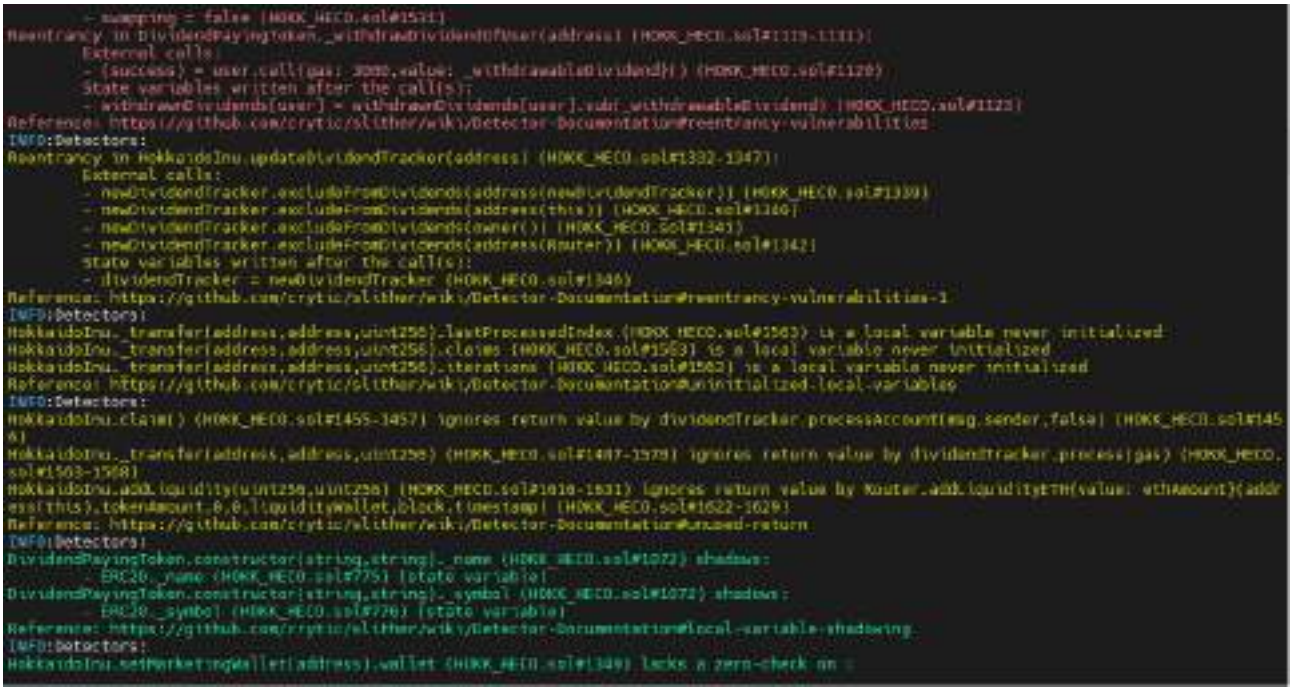


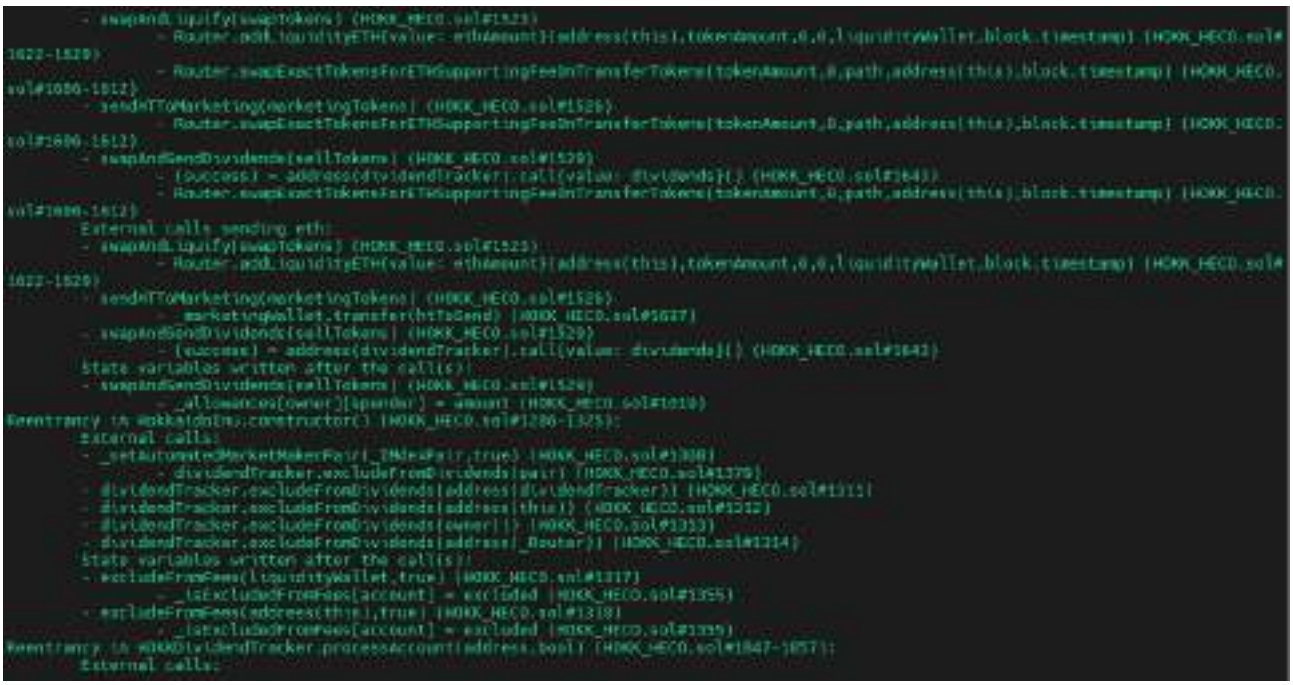
```
        - ERC20.decimals() (HOKK_BEP.sol#700-702)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (HOKK_BEP.sol#858-860)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (HOKK_BEP.sol#824-826)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (HOKK_BEP.sol#835-838)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (HOKK_BEP.sol#853-861)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (HOKK_BEP.sol#875-878)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (HOKK_BEP.sol#894-897)
withdrawDividend() should be declared external:
        - DividendPayingToken.withdrawDividend() (HOKK_BEP.sol#1079-1081)
        - HOKKDividendTracker.withdrawDividend() (HOKK_BEP.sol#1656-1658)
dividendOf(address) should be declared external:
        - DividendPayingToken.dividendOf(address) (HOKK_BEP.sol#1107-1109)
withdrawnDividendOf(address) should be declared external:
        - DividendPayingToken.withdrawnDividendOf(address) (HOKK_BEP.sol#1121-1123)
updateDividendTracker(address) should be declared external:
        - HokkaidoInu.updateDividendTracker(address) (HOKK_BEP.sol#1302-1317)
updateUniswapV2Router(address) should be declared external:
        - HokkaidoInu.updateUniswapV2Router(address) (HOKK_BEP.sol#1319-1323)
setMarketingWallet(address) should be declared external:
        - HokkaidoInu.setMarketingWallet(address) (HOKK_BEP.sol#1325-1327)
excludeMultipleAccountsFromFees(address[],bool) should be declared external:
        - HokkaidoInu.excludeMultipleAccountsFromFees(address[],bool) (HOKK_BEP.sol#1330-1342)
setAutomatedMarketMakerPair(address,bool) should be declared external:
        - HokkaidoInu.setAutomatedMarketMakerPair(address,bool) (HOKK_BEP.sol#1344-1348)
setBridgeAddresses(address,address) should be declared external:
        - HokkaidoInu.setBridgeAddresses(address,address) (HOKK_BEP.sol#1381-1384)
swapAcrossChain(string,string,uint256,address,address) should be declared external:
        - HokkaidoInu.swapAcrossChain(string,string,uint256,address,address) (HOKK_BEP.sol#1386-1391)
portMessage(address,uint256) should be declared external:
        - HokkaidoInu.portMessage(address,uint256) (HOKK_BEP.sol#1372-1375)
updateLiquidityWallet(address) should be declared external:
        - HokkaidoInu.updateLiquidityWallet(address) (HOKK_BEP.sol#1378-1380)
```

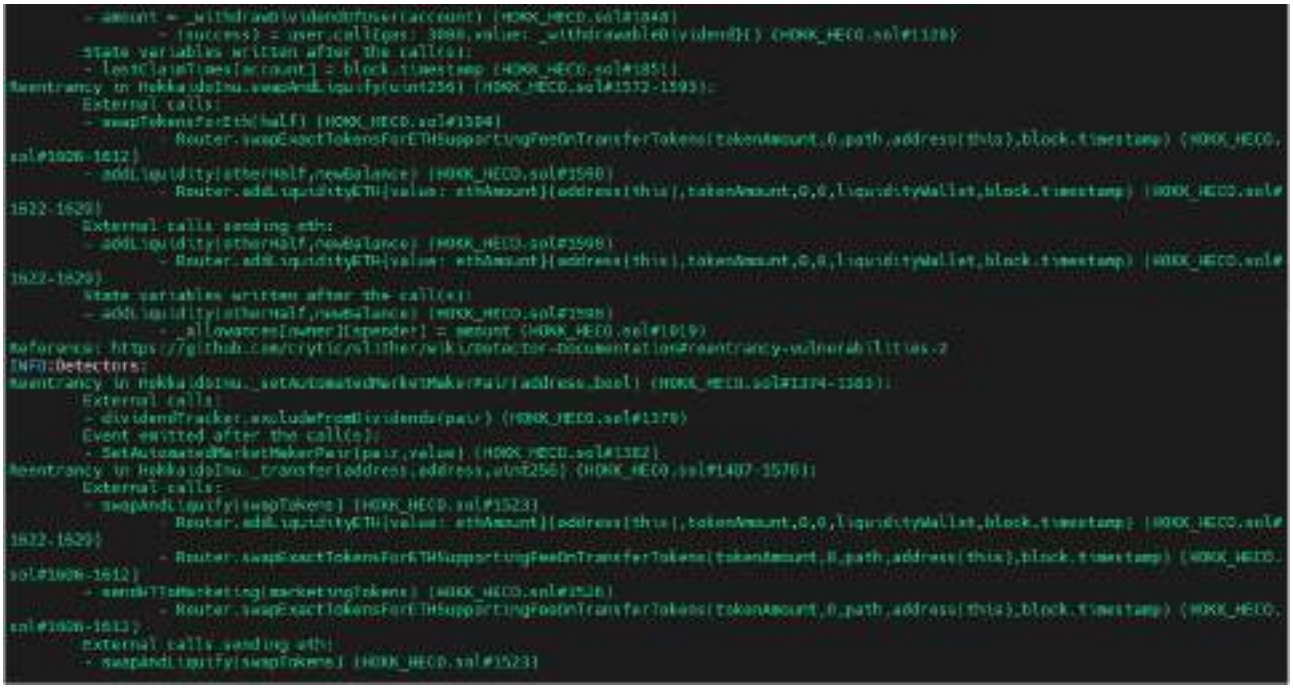
```
updateLiquidityWallet(address) should be declared external:
        - HokkaidoInu.updateLiquidityWallet(address) (HOKK_BEP.sol#1378-1380)
updateGasForProcessing(uint256) should be declared external:
        - HokkaidoInu.updateGasForProcessing(uint256) (HOKK_BEP.sol#1385-1398)
isExcludedFromFees(address) should be declared external:
        - HokkaidoInu.isExcludedFromFees(address) (HOKK_BEP.sol#1404-1406)
withdrawableDividendOf(address) should be declared external:
        - HokkaidoInu.withdrawableDividendOf(address) (HOKK_BEP.sol#1408-1410)
dividendTokenBalanceOf(address) should be declared external:
        - HokkaidoInu.dividendTokenBalanceOf(address) (HOKK_BEP.sol#1412-1414)
getMarketingFeePercent() should be declared external:
        - HokkaidoInu.getMarketingFeePercent() (HOKK_BEP.sol#1451-1453)
getAccountAtIndex(uint256) should be declared external:
        - HOKKDividendTracker.getAccountAtIndex(uint256) (HOKK_BEP.sol#1732-1740)
process(uint256) should be declared external:
        - HOKKDividendTracker.process(uint256) (HOKK_BEP.sol#1775-1821)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HOKK_BEP.sol analyzed (10 contracts with 75 detectors), 107 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

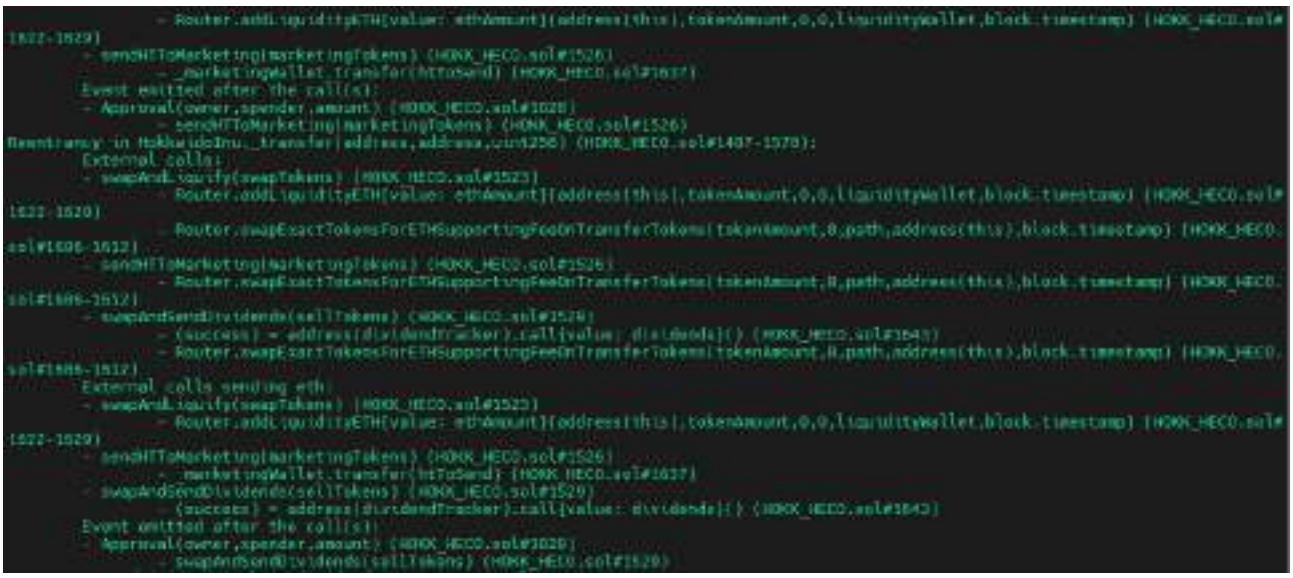
Slither log >> HOKK-HECO.sol

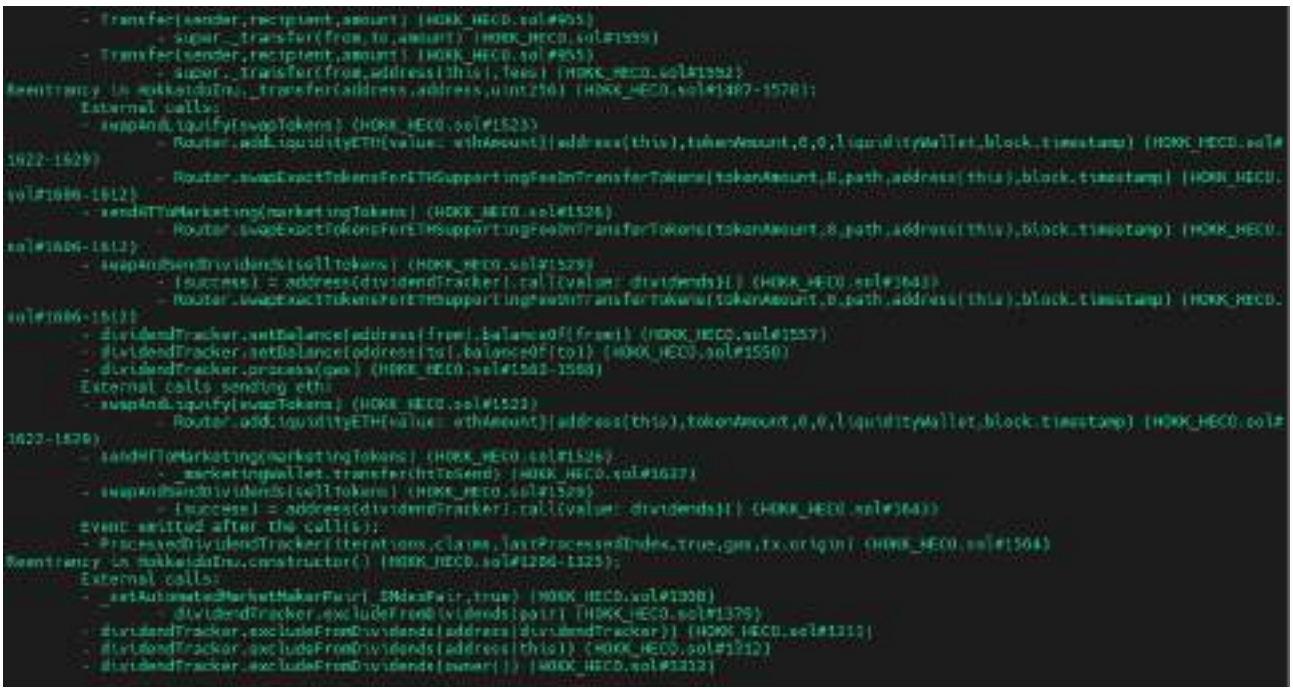
```
INFO:Detectors:
HokkaidoInu.sendBNBToMarketing(uint256) (HOKK_HECO.sol#1633-1638) sends eth to arbitrary user
        Dangerous calls:
        - _marketingWallet.transfer(bnbToSend) (HOKK_HECO.sol#1637)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in HokkaidoInu._transfer(address,address,uint256) (HOKK_HECO.sol#1607-1676):
        External calls:
        - swapAndLiquify(swapTokens) (HOKK_HECO.sol#1523)
                - Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityWallet,block.timestamp) (HOKK_HECO.sol#1622-1629)
                - Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (HOKK_HECO.sol#1606-1612)
        - sendBNBToMarketing(marketingTokens) (HOKK_HECO.sol#1526)
                - Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (HOKK_HECO.sol#1606-1612)
        - swapAndSendDividends(sellTokens) (HOKK_HECO.sol#1529)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_HECO.sol#1643)
                - Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (HOKK_HECO.sol#1606-1612)
        External calls sending eth:
        - swapAndLiquify(swapTokens) (HOKK_HECO.sol#1523)
                - Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityWallet,block.timestamp) (HOKK_HECO.sol#1622-1629)
        - sendBNBToMarketing(marketingTokens) (HOKK_HECO.sol#1526)
                - _marketingWallet.transfer(bnbToSend) (HOKK_HECO.sol#1637)
        - swapAndSendDividends(sellTokens) (HOKK_HECO.sol#1529)
                - (success) = address(dividendTracker).call{value: dividends}() (HOKK_HECO.sol#1643)
        State variables written after the call(s):
        - super._transfer(from,address(this),fees) (HOKK_HECO.sol#1552)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (HOKK_HECO.sol#953)
                - _balances[recipient] = _balances[recipient].add(amount) (HOKK_HECO.sol#954)
        - super._transfer(from,to,amount) (HOKK_HECO.sol#1555)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (HOKK_HECO.sol#953)
                - _balances[recipient] = _balances[recipient].add(amount) (HOKK_HECO.sol#954)
```

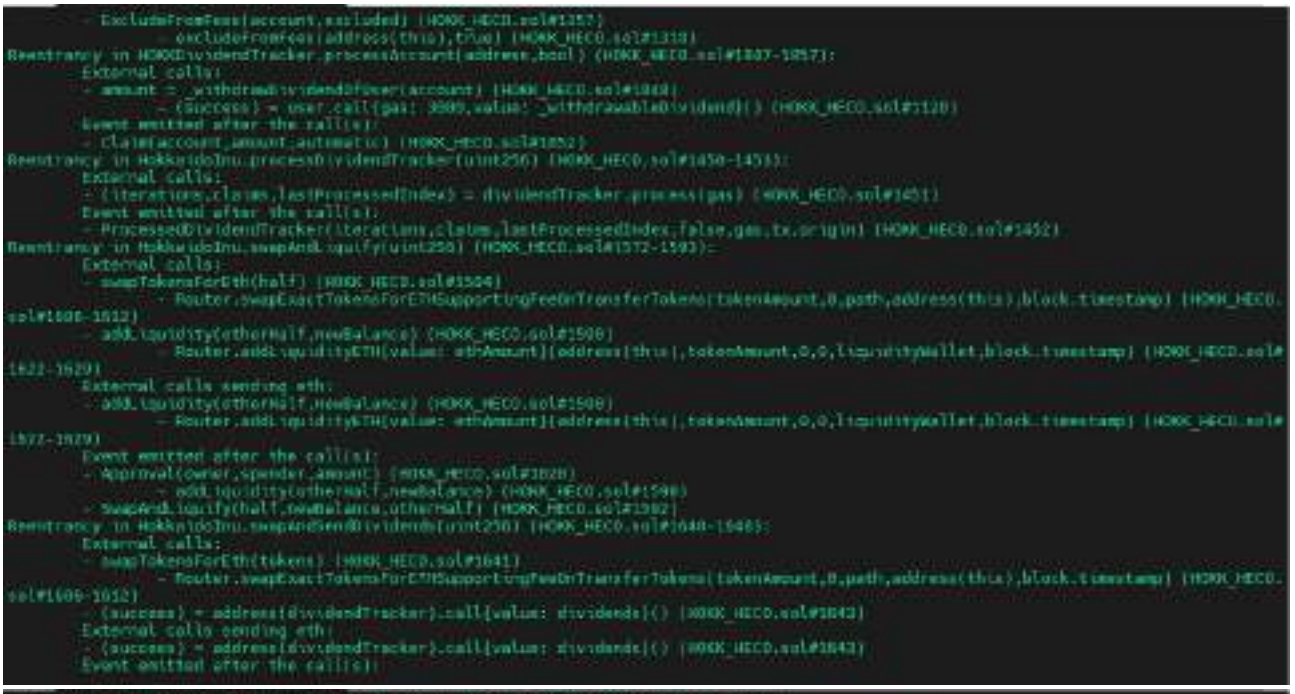


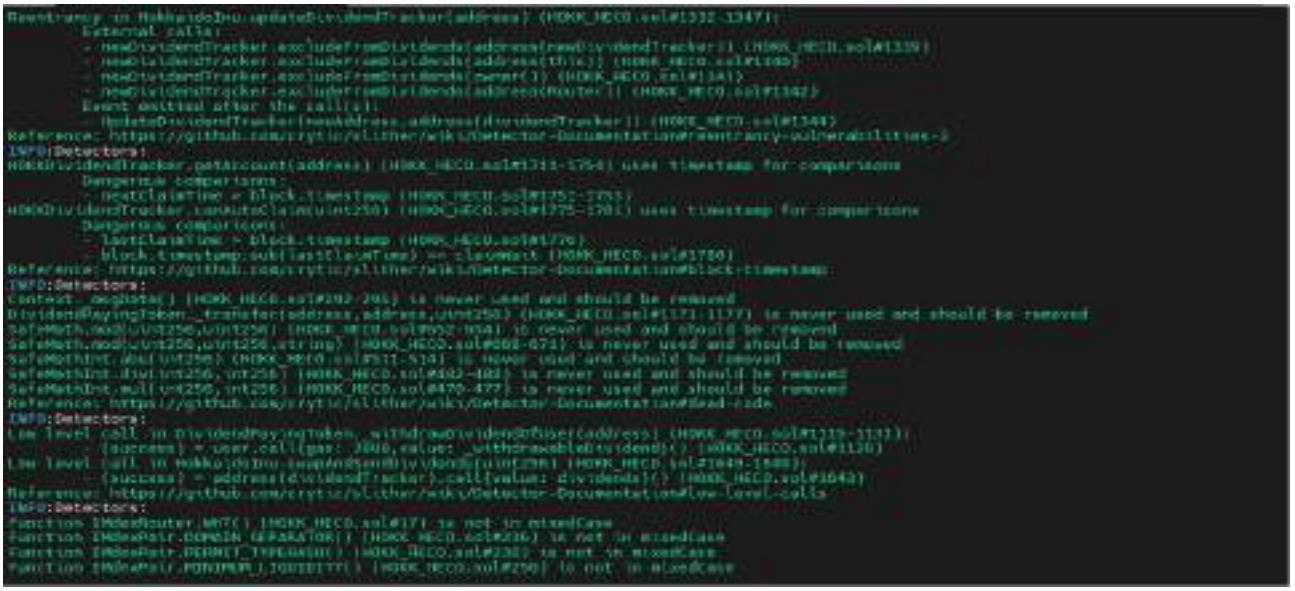


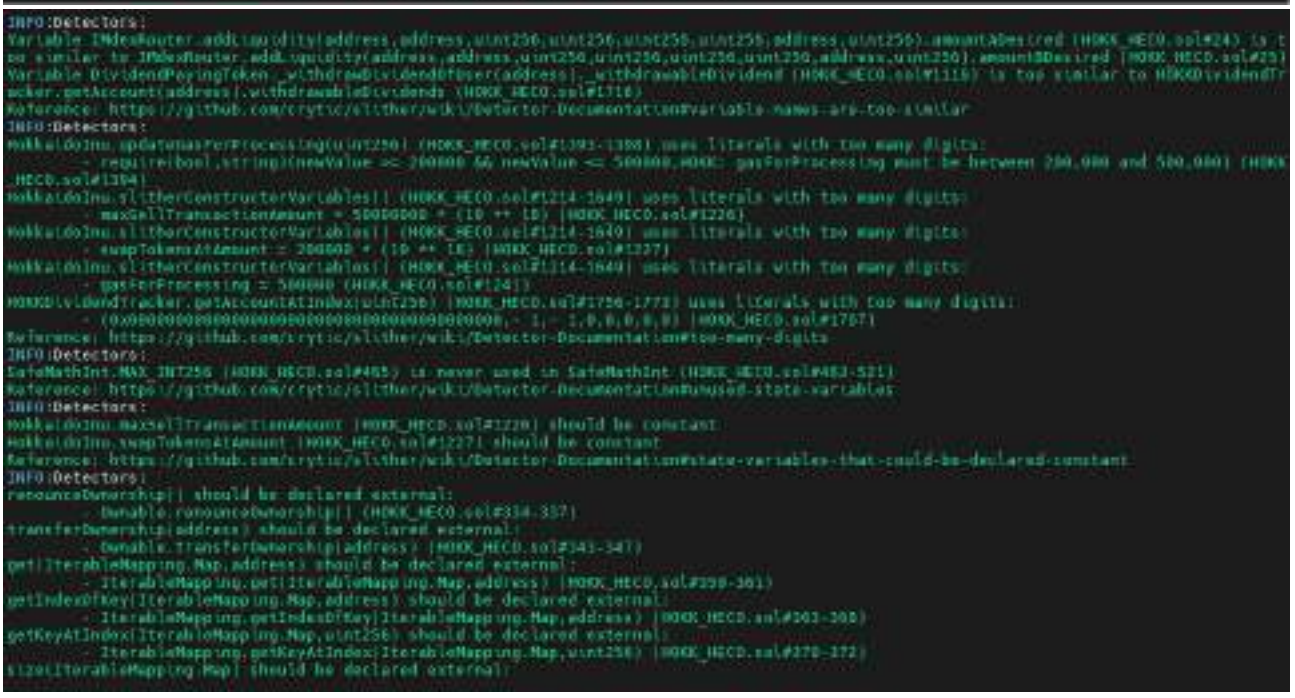


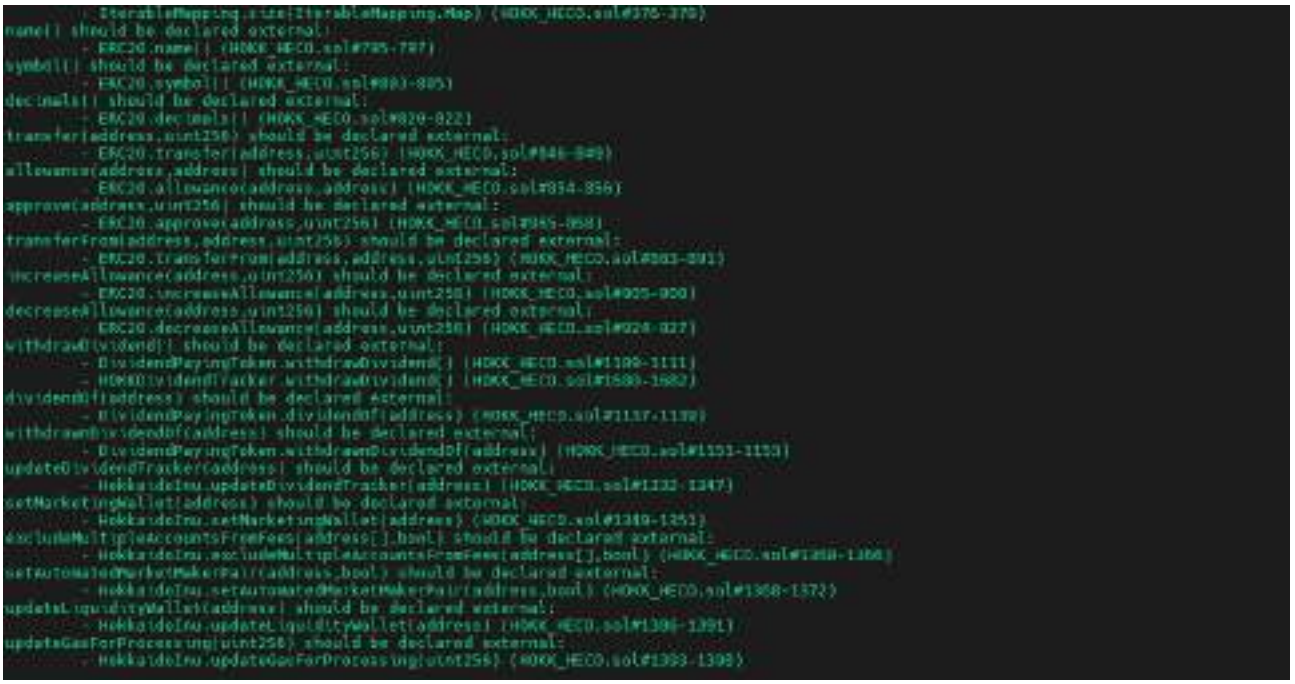


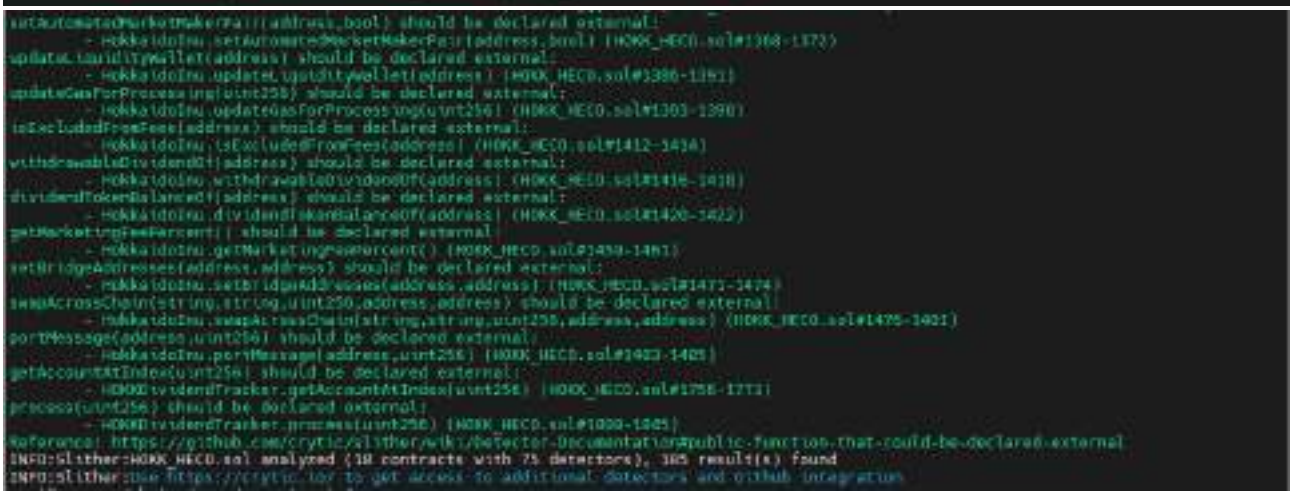












Solidity static analysis

HOKK-ETH.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1647:90:

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1754:97:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1094:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HOKK.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1481:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HOKK._setAutomatedMarketMakerPair(address,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1531:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HOKK.processDividendTracker(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1645:4:

Gas & Economy

Gas costs:

Gas requirement of function DividendPayingToken.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: 771:4:

Gas costs:

Gas requirement of function ERC20.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: 771:4:

Gas costs:

Gas requirement of function HOKK.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: 771:4:

Gas costs:

Gas requirement of function ERC20.transfer 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: 822:4:

Gas costs:

Gas requirement of function HOKK.transfer 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: 822:4:

Gas costs:

Gas requirement of function DividendPayingToken.allowance 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: 830:4:

Gas costs:

Gas requirement of function ERC20.transferFrom 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: 859:4:

Gas costs:

Gas requirement of function HOKK.transferFrom 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: 859:4:

Gas costs:

Gas requirement of function HOKKDividendTracker.transferFrom 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: 859:4:

No return:

IERC20Metadata.decimals(): Defines a return type but never explicitly returns a value.
Pos: 715:4:

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: 342:8:

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: 343:8:

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: 352:8:

HOKK-BEP.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1444:84:

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1540:88:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1085:2:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu (): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1256:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu.updateDividendTracker(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1302:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu._setAutomatedMarketMakerPair(address,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1350:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HokkaidoInu.processDividendTracker(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1442:1:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HokkaidoInu._transfer(address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1463:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HokkaidoInu.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1571:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HokkaidoInu.swapAndSendDividends(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1616:4:

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: 1587:12:

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: 1604:12:

Gas & Economy

Gas costs:

Gas requirement of function HOKKDividendTracker.transferOwnership 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: 272:4:

Gas costs:

Gas requirement of function HokkaidoInu.transferOwnership 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: 272:4:

Gas costs:

Gas requirement of function Ownable.transferOwnership 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: 272:4:

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: 1252:8:

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: 1303:8:

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: 1307:8:

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: 1320:8:

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1452:84:

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1564:88:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DividendPayingToken._withdrawDividendOfUser(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1115:2:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu._setAutomatedMarketMakerPair(address,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1374:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu.processDividendTracker(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1450:1:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hokkaidoinu._transfer(address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1487:4:

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: 1611:12:

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: 1628:12:

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: 1751:57:

Gas & Economy

Gas costs:

Gas requirement of function HOKKDividendTracker.transferOwnership 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: 343:4:

Gas costs:

Gas requirement of function HokkaidoInu.transferOwnership 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: 343:4:

Gas costs:

Gas requirement of function Ownable.transferOwnership 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: 343:4:

Gas costs:

Gas requirement of function Hokkaidolnu.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: 795:4:

Gas costs:

Gas requirement of function DividendPayingToken.symbol 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: 803:4:

Gas costs:

Gas requirement of function ERC20.symbol 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: 803:4:

Constant/View/Pure functions:

HOKKDividendTracker._transfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1676:4:

Constant/View/Pure functions:

HOKKDividendTracker.withdrawDividend() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1680:4:

Constant/View/Pure functions:

HOKKDividendTracker.getAccount(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1711:4:

Constant/View/Pure functions:

HOKKDividendTracker.getAccountAtIndex(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1756:4:

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: 599:16:

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: 634:20:

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: 1099:8:

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: 1162:11:

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: 1681:8:

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: 1685:5:

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: 1695:8:

Solhint Linter

HOKK-ETH.sol

```
HOKK_ETH.sol:2:1: Error: Compiler version ^0.8.4 does not satisfy the r
semver requirement
HOKK_ETH.sol:16:5: Error: Function name must be in mixedCase
HOKK_ETH.sol:180:5: Error: Function name must be in mixedCase
HOKK_ETH.sol:181:5: Error: Function name must be in mixedCase
HOKK_ETH.sol:198:5: Error: Function name must be in mixedCase
HOKK_ETH.sol:575:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
HOKK_ETH.sol:763:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
HOKK_ETH.sol:1017:24: Error: Code contains empty blocks
HOKK_ETH.sol:1028:31: Error: Constant name must be in capitalized
SNAKE_CASE
HOKK_ETH.sol:1051:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
HOKK_ETH.sol:1099:31: Error: Avoid using low level calls.
HOKK_ETH.sol:1214:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
HOKK_ETH.sol:1288:58: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1288:94: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1310:29: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1313:16: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1382:39: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1409:5: Error: Explicitly mark visibility of state
HOKK_ETH.sol:1410:5: Error: Explicitly mark visibility of state
HOKK_ETH.sol:1481:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
HOKK_ETH.sol:1517:53: Error: Code contains empty blocks
HOKK_ETH.sol:1520:32: Error: Code contains empty blocks
HOKK_ETH.sol:1647:91: Error: Avoid to use tx.origin
HOKK_ETH.sol:1686:85: Error: Code contains empty blocks
HOKK_ETH.sol:1745:72: Error: Code contains empty blocks
HOKK_ETH.sol:1745:81: Error: Code contains empty blocks
HOKK_ETH.sol:1746:68: Error: Code contains empty blocks
HOKK_ETH.sol:1746:77: Error: Code contains empty blocks
HOKK_ETH.sol:1754:98: Error: Avoid to use tx.origin
HOKK_ETH.sol:1755:21: Error: Code contains empty blocks
HOKK_ETH.sol:1780:51: Error: Code contains empty blocks
HOKK_ETH.sol:1802:13: Error: Avoid to make time-based decisions in your
business logic
HOKK_ETH.sol:1810:27: Error: Avoid using low level calls.
```

HOKK-BEP.sol

```
HOKK_BEP.sol:2:1: Error: Compiler version ^0.6.2 does not satisfy the r
semver requirement
HOKK_BEP.sol:16:5: Error: Function name must be in mixedCase
HOKK_BEP.sol:180:5: Error: Function name must be in mixedCase
HOKK_BEP.sol:181:5: Error: Function name must be in mixedCase
HOKK_BEP.sol:198:5: Error: Function name must be in mixedCase
HOKK_BEP.sol:1011:24: Error: Code contains empty blocks
HOKK_BEP.sol:1022:29: Error: Constant name must be in capitalized
SNAKE_CASE
HOKK_BEP.sol:1042:88: Error: Code contains empty blocks
HOKK_BEP.sol:1090:25: Error: Avoid using low level calls.
HOKK_BEP.sol:1199:30: Error: Variable name must be in mixedCase
HOKK_BEP.sol:1206:5: Error: Explicitly mark visibility of state
HOKK_BEP.sol:1257:9: Error: Variable name must be in mixedCase
HOKK_BEP.sol:1298:32: Error: Code contains empty blocks
HOKK_BEP.sol:1444:85: Error: Avoid to use tx.origin
HOKK_BEP.sol:1533:72: Error: Code contains empty blocks
HOKK_BEP.sol:1533:81: Error: Code contains empty blocks
HOKK_BEP.sol:1534:68: Error: Code contains empty blocks
HOKK_BEP.sol:1534:77: Error: Code contains empty blocks
HOKK_BEP.sol:1540:89: Error: Avoid to use tx.origin
HOKK_BEP.sol:1542:13: Error: Code contains empty blocks
HOKK_BEP.sol:1587:13: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1604:13: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1619:27: Error: Avoid using low level calls.
HOKK_BEP.sol:1727:58: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1728:71: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1752:25: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1756:13: Error: Avoid to make time-based decisions in your
business logic
HOKK_BEP.sol:1827:33: Error: Avoid to make time-based decisions in your
business logic
```

HOKK-HECO.sol

```
HOKK_HECO.sol:2:1: Error: Compiler version ^0.6.2 does not satisfy the
r semver requirement
HOKK_HECO.sol:17:5: Error: Function name must be in mixedCase
HOKK_HECO.sol:236:5: Error: Function name must be in mixedCase
HOKK_HECO.sol:238:5: Error: Function name must be in mixedCase
HOKK_HECO.sol:256:5: Error: Function name must be in mixedCase
HOKK_HECO.sol:1041:24: Error: Code contains empty blocks
HOKK_HECO.sol:1052:29: Error: Constant name must be in capitalized
SNAKE_CASE
HOKK_HECO.sol:1072:88: Error: Code contains empty blocks
```

```
HOKK_HECO.sol:1120:25: Error: Avoid to use low level calls.
HOKK_HECO.sol:1217:24: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1218:30: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1229:30: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1238:5: Error: Explicitly mark visibility of state
HOKK_HECO.sol:1287:9: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1301:6: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1303:9: Error: Variable name must be in mixedCase
HOKK_HECO.sol:1328:32: Error: Code contains empty blocks
HOKK_HECO.sol:1452:85: Error: Avoid to use tx.origin
HOKK_HECO.sol:1557:72: Error: Code contains empty blocks
HOKK_HECO.sol:1557:81: Error: Code contains empty blocks
HOKK_HECO.sol:1558:68: Error: Code contains empty blocks
HOKK_HECO.sol:1558:77: Error: Code contains empty blocks
HOKK_HECO.sol:1564:89: Error: Avoid to use tx.origin
HOKK_HECO.sol:1566:13: Error: Code contains empty blocks
HOKK_HECO.sol:1611:13: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1628:13: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1643:27: Error: Avoid to use low level calls.
HOKK_HECO.sol:1751:58: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1752:71: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1776:25: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1780:13: Error: Avoid to make time-based decisions in
your business logic
HOKK_HECO.sol:1851:33: 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