Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: HotDogs
Website: thehotdogsproject.com
Platform: Binance Smart Chain
Language: Solidity
Date: August 23rd, 2022

Table of contents

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 HotDogs to perform the Security audit of the HotDogs 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 August 23rd, 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

HotDogs is a multi-contract BNB miner in which users can deposit BNB in return for daily rewards. The HotDogs contract inherits the LinkTokenInterface, VRFCoordinatorV2Interface, VRFConsumerBaseV2standard smart contracts from the chain link library. These chainlink 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 HotDogs Protocol Smart Contracts**
Platform	**BSC / Solidity**
File 1	HotDogsLotto.sol
File 1 MD5 Hash	E426450138D767CC588CF317280E2CFE
Updated File 1 MD5 Hash	1B1E4D7ADA787C1B5410B11AE4DC43FB
File 2	HotDogsMiner.sol
File 2 MD5 Hash	7EAFD38CD35E8EDDB653A9C8433B28E9
Updated File 2 MD5 Hash	50C26FBE1189AB1CEA63E4FA4F54F0F6
File 3	HotDogsTreasury.sol
File 3 MD5 Hash	A002C4906B4A922755DA884E05671600
Updated File 3 MD5 Hash	33700646D159AFC52F532A864552CE9A

File 4	RandomNumberGenerator.sol
File 4 MD5 Hash	28C96AB9C0D5200461CFD96DC468FBD1
Updated File 4 MD5 Hash	CCA2F1587202C40EBB85972FC84B5F00
Audit Date	August 23rd,2022
Revise Audit Date	August 31st,2022

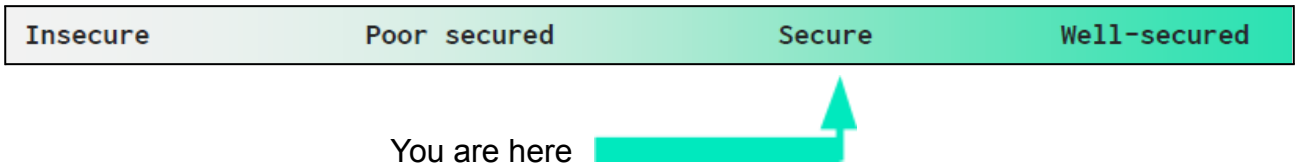
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 HotDogsLotto.sol • Maximum Price Ticket: 0.1 BNB • Minimum Price Ticket: 0.005 BNB • Maximum Length Lottery: 30 Days • Minimum Length Lottery: 1 Day • Maximum Number Tickets: 100 • Minimum Discount Divisor: 10% • Maximum Treasury Fee: 20% • Tvl Fee: 10%	**YES, This is valid.**
File 2 HotDogsMiner.sol • Accumulation Exemption: 0.01 BNB • Lotto Fee: 1% • Dev Fee: 4% • AP minimum exemption: 0.01 BNB	**YES, This is valid.**
File 3 HotDogsTreasury.sol • HotDogsTreasury has functions like: setLotteryAddress, fundLottery, etc.	**YES, This is valid.**
File 4 RandomNumberGenerator.sol • Request Confirmations: 5 • Callback Gas Limit: 0.1 Million • Returned Values: 1	**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, 0 high, 1 medium and 2 low and some very low level issues.
All the issues have been resolved in the revised code.

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	Passed
	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	Passed
	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	Passed
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	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 4 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in HotDogs 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 HotDogs.

The HotDogs team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

All code parts are well commented on smart contracts.

Documentation

We were given a HotDogs 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 well commented. And the logic is straightforward. So it is 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://thehotdogsproject.com/ which provided rich information about the project architecture.

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

HotDogsLotto.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	nonReentrant	modifier	Passed	No Issue
7	notContract	modifier	Passed	No Issue
8	onlyOperator	modifier	Passed	No Issue
9	onlyOwnerOrOperator	modifier	Passed	No Issue
10	buyTickets	external	Passed	No Issue
11	claimTickets	external	Passed	No Issue
12	closeLottery	external	access only Operator	No Issue
13	drawFinalNumberAndMakeClaimable	external	access only Operator	No Issue
14	changeRandomGenerator	external	access only Owner	No Issue
15	injectFunds	external	access only Owner Or Operator	No Issue
16	startLottery	external	access only Operator	No Issue
17	setMinAndMaxTicketPrice	external	access only Owner	No Issue
18	setMinerAndTreasuryAddresses	external	access only Owner	No Issue
19	setOperatorAddress	external	access only Owner	No Issue
20	calculateTicketsBulkPrice	external	Passed	No Issue
21	viewCurrentLotteryId	external	Passed	No Issue
22	viewLottery	external	Passed	No Issue
23	viewNumbersAndStatusesForTicketIds	external	Passed	No Issue
24	viewRewardsForTicketId	external	Passed	No Issue
25	viewUserInfoForLotteryId	external	Passed	No Issue
26	_calculateRewardsForTicketId	internal	Passed	No Issue
27	_calculateTicketsBulkPrice	internal	Passed	No Issue
28	_isContract	internal	Passed	No Issue
29	reinjectNonClaimedRewardsToLottery	external	access only Owner	No Issue
30	giveawayTickets	external	access only Owner Or Operator	No Issue
31	receive	external	Passed	No Issue
32	checkBalance	read	Passed	No Issue

33	checkNonClaimedRewards	external	access only Owner Or Operator	No Issue

HotDogsMiner.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	nonReentrant	modifier	Passed	No Issue
7	cookHotDogs	external	Passed	No Issue
8	eatHotDogs	external	Passed	No Issue
9	cookHotDogsInternal	write	Passed	No Issue
10	buyHotDogs	external	Passed	No Issue
11	switchVacationMode	external	Passed	No Issue
12	getBalance	external	Passed	No Issue
13	getMarketHotDogs	external	access only Owner	No Issue
14	getMyHotDogsChefs	external	Passed	No Issue
15	getMyHotDogs	external	Passed	No Issue
16	getHotDogsSincelastCooked	read	Passed	No Issue
17	getUserInfo	external	access only Owner	No Issue
18	getMyHotDogsInternal	read	Passed	No Issue
19	getVacationMode	external	Passed	No Issue
20	hasEatPenalty	read	Passed	No Issue
21	getPenalties	external	Passed	No Issue
22	getEatsMonth	external	Passed	No Issue
23	getDatePenalties	external	Passed	No Issue
24	hotDogsRewards	external	Passed	No Issue
25	seedMarket	external	access only Owner	No Issue
26	balanceMarket	external	access only Owner	No Issue
27	setAccumulationPenaltyExemption	external	access only Owner	No Issue
28	updateLotteryStatus	external	access only Owner	No Issue
29	retrieveEatPenalty	read	Passed	No Issue
30	applyDailyCap	read	Passed	No Issue
31	applyMaxRewardsLimit	read	Passed	No Issue
32	applyAccumulationPenalty	read	Passed	No Issue
33	retrieveAccumulationPenalty	read	Passed	No Issue
34	getEnableCookHotdogs	external	Passed	No Issue
35	receive	external	Passed	No Issue
36	percent	write	Passed	No Issue
37	calculateTrade	write	Passed	No Issue
38	calculateHotDogsSell	read	Passed	No Issue

39	calculateHotDogsBuy	read	Passed	No Issue
40	calculateFeeValue	write	Passed	No Issue
41	calculateHotDogsBuySimple	read	Passed	No Issue
42	min	write	Passed	No Issue
43	max	write	Passed	No Issue

HotDogsTreasury.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	nonReentrant	modifier	Passed	No Issue
7	setLotteryAddress	external	access only Owner	No Issue
8	fundLottery	external	access only Owner	No Issue
9	checkBalance	read	Passed	No Issue
10	receive	external	Passed	No Issue

RandomNumberGenerator.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	getRandomNumber	external	Passed	No Issue
7	fulfillRandomWords	internal	Passed	No Issue
8	createNewSubscription	write	access only Owner	No Issue
9	topUpSubscription	external	access only Owner	No Issue
10	addConsumer	external	access only Owner	No Issue
11	removeConsumer	external	access only Owner	No Issue
12	cancelSubscription	external	access only Owner	No Issue
13	withdraw	external	access only Owner	No Issue
14	setKeyHash	external	access only Owner	No Issue
15	setLotteryAddress	external	access only Owner	No Issue
16	viewLatestLotteryId	external	Passed	No Issue
17	viewRandomResult	external	Passed	No Issue
18	functionCallWithValue	internal	Removed	
19	sendValue	internal	Removed	

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 loss
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

No High severity vulnerabilities were found.

Medium

(1) Out of Gas issue:

HotDogsTreasury.sol

```
76
77      function fundLottery() external payable onlyOwner {
78          require(initialized == true, "Must set the Lottery address first");
79          uint256 _amount = address(this).balance;
80
81          (bool success, ) = payable(lotteryAddress).call{value: _amount}("");
82              require(success, "Transfer failed.");
83
84          emit FundLottery(_amount);
85      }
86
```

RandomNumberGenerator.sol.sol

```
function functionCallWithValue(
    address target,
    bytes memory data,
    uint256 value,
    string memory errorMessage
) internal returns (bytes memory) {
    require(address(this).balance >= value, "Address: insufficient balance for call");
    require(isContract(target), "Address: call to non-contract");

    (bool success, bytes memory returndata) = target.call{value: value}(data);
    return _verifyCallResult(success, returndata, errorMessage);
}
```

```
function sendValue(address payable recipient, uint256 amount) internal {
    require(address(this).balance >= amount, "Address: insufficient balance");

    (bool success, ) = recipient.call{value: amount}("");
    require(success, "Address: unable to send value, recipient may have reverted");
}
```

HotDogsMiner.sol

function: eatHotDogs()

```
        payable(wDevs[0]).transfer(_devFee/2);
        payable(wDevs[1]).transfer(_devFee/2);
        (bool success, ) = payable(wLotto[lotteryStatus]).call{value: _lottoFee}("");
        require(success, "Transfer failed.");
        payable(msg.sender).transfer(_toWithdraw);
        emit EatHotDogs(msg.sender, _toWithdraw);
    }

}
```

function: buyHotDogs()

```
    //Cooks previous accumulated rewards
    cookHotDogsInternal(hotDogsBought);

    payable(wDevs[0]).transfer(_devFee/2);
    payable(wDevs[1]).transfer(_devFee/2);
    (bool success, ) = payable(wLotto[lotteryStatus]).call{value: _lottoFee}("");
    require(success, "Transfer failed.");

    emit HotDogsBought(msg.sender, hotDogsBought);
}
```

".call" is used to transfer coins without a gas limit.

Resolution: Remove it and use ". transfer" instead or apply the gas amount.

Status: Fixed by adding gas limit with call.

Low

(1) Function input parameters lack of check:

RandomNumberGenerator.sol

```
function withdraw(uint256 amount, address to) external onlyOwner {
    LINKTOKEN.transfer(to, amount);
}
```

Amount and address should be checked for "0" and address(0) value.

HotDogsMiner.sol

```
constructor(address _w1, address _w2, address _hdLottery, address _treasury) {
    wDevs.push(_w1);
    wDevs.push(_w2);
    wLotto.push(_hdLottery);
    wLotto.push(_treasury);
}
```

Check the requirements of all the addresses assigned in the constructor.

HotDogsLotto.sol

```
constructor(address _randomGeneratorAddress) {
    randomGenerator = IRandomNumberGenerator(_randomGeneratorAddress);

    _bracketCalculator[0] = 1;
    _bracketCalculator[1] = 11;
    _bracketCalculator[2] = 111;
    _bracketCalculator[3] = 1111;
    _bracketCalculator[4] = 11111;
    _bracketCalculator[5] = 111111;
}
```

Check for address(0) of _randomGeneratorAddress in the constructor.

Resolution: We suggest using validation like amount should be greater than zero and address should not be dead address.

Status: Fixed.

(2) Gas Consumption: **HotDogsMiner.sol**

```
function buyHotDogs(address ref) nonReentrant external payable {
    require(initialized);

    if (users[msg.sender].cycleStartDate == 0) {
        users[msg.sender].cycleStartDate = block.timestamp;
    }

    if (users[msg.sender].withdrawalAmount > users[msg.sender].depositedAmount) {
        users[msg.sender].additionalDeposits += msg.value;
    } else {
        users[msg.sender].depositedAmount += msg.value;
    }

    uint256 hotDogsBought = calculateHotDogsBuy(msg.value, address(this).balance - msg.value);

    uint256 _devFee = calculateFeeValue(msg.value, DEV_FEE);
    uint256 _lottoFee = calculateFeeValue(msg.value, LOTTO_FEE);
    hotDogsBought -= calculateFeeValue(hotDogsBought, DEV_FEE + LOTTO_FEE);
```

Use a local variable instead of calling msg.sender and msg.value multiple times inside a function.

Resolution: Instead of calling state variables multiple times, assign this state variable value to a local variable to that function and use it.

Status: Fixed.

Very Low / Informational / Best practices

(1) Initialized by default value: **HotDogsMiner.sol**

```
bool private initialized = false;
```

Resolution: We suggest ignoring to initialize the variables by default value.

Status: Fixed

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:

- getRandomNumber: RandomNumberGenerator hotDogsLottery owner can get a random valid key hash list.
- createNewSubscription: RandomNumberGenerator owner can create new subscriptions.
- topUpSubscription: RandomNumberGenerator owner can top up subscription amount.
- addConsumer: RandomNumberGenerator owner can add a new consumer address.
- removeConsumer: RandomNumberGenerator owner can remove consumer address.
- cancelSubscription: RandomNumberGenerator owner can cancel consumer address.
- withdraw: RandomNumberGenerator owner can withdraw amount from address.
- setKeyHash: RandomNumberGenerator owner can set key hash like: vrfCoordinator address, _linkToken address, _keyHash.
- setLotteryAddress: RandomNumberGenerator owner can set the lottery address.
- setLotteryAddress: HotDogsTreasury owner can set the lottery address.
- fundLottery: HotDogsTreasury owner can fund the lottery.
- getMarketHotDogs: HotDogsMiner owner can get market hot dogs.
- getUserInfo: HotDogsMiner owner can get user information address.
- seedMarket: HotDogsMiner owner can send funds to the treasury.
- balanceMarket: HotDogsMiner owner can balance market funds.
- setAccumulationPenaltyExemption: HotDogsMiner owner can set accumulation penalty exemption value.
- updateLotteryStatus: HotDogsMiner owner can update lottery status.
- closeLottery: HotDogsLotto owner can close lottery status.
- drawFinalNumberAndMakeClaimable: HotDogsLotto owner can draw the final number and make the claimable value.

- changeRandomGenerator: HotDogsLotto owner can change the random generator address.
- injectFunds: HotDogsLotto owner or operator can inject funds.
- startLottery: HotDogsLotto owner can start a lottery.
- setMinAndMaxTicketPrice: HotDogsLotto owner can set the minimum and maximum ticket price.
- setMinerAndTreasuryAddresses: HotDogsLotto owner can set minimum and maximum ticket addresses.
- setOperatorAddress: HotDogsLotto owner can set operator address.
- reinjectNonClaimedRewardsToLottery: HotDogsLotto owner can reinject non claimed rewards to lottery.
- giveawayTickets: HotDogsLotto owner can giveaway ticket addresses.
- checkNonClaimedRewards: HotDogsLotto owner can check non claimed rewards.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the 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 have observed 1 medium Severity issue and 2 low issues and some informational issues in smart contracts. But those are not critical ones and all issues have been resolved in the revised code. **So, the smart contracts are ready for the mainnet deployment**.

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.

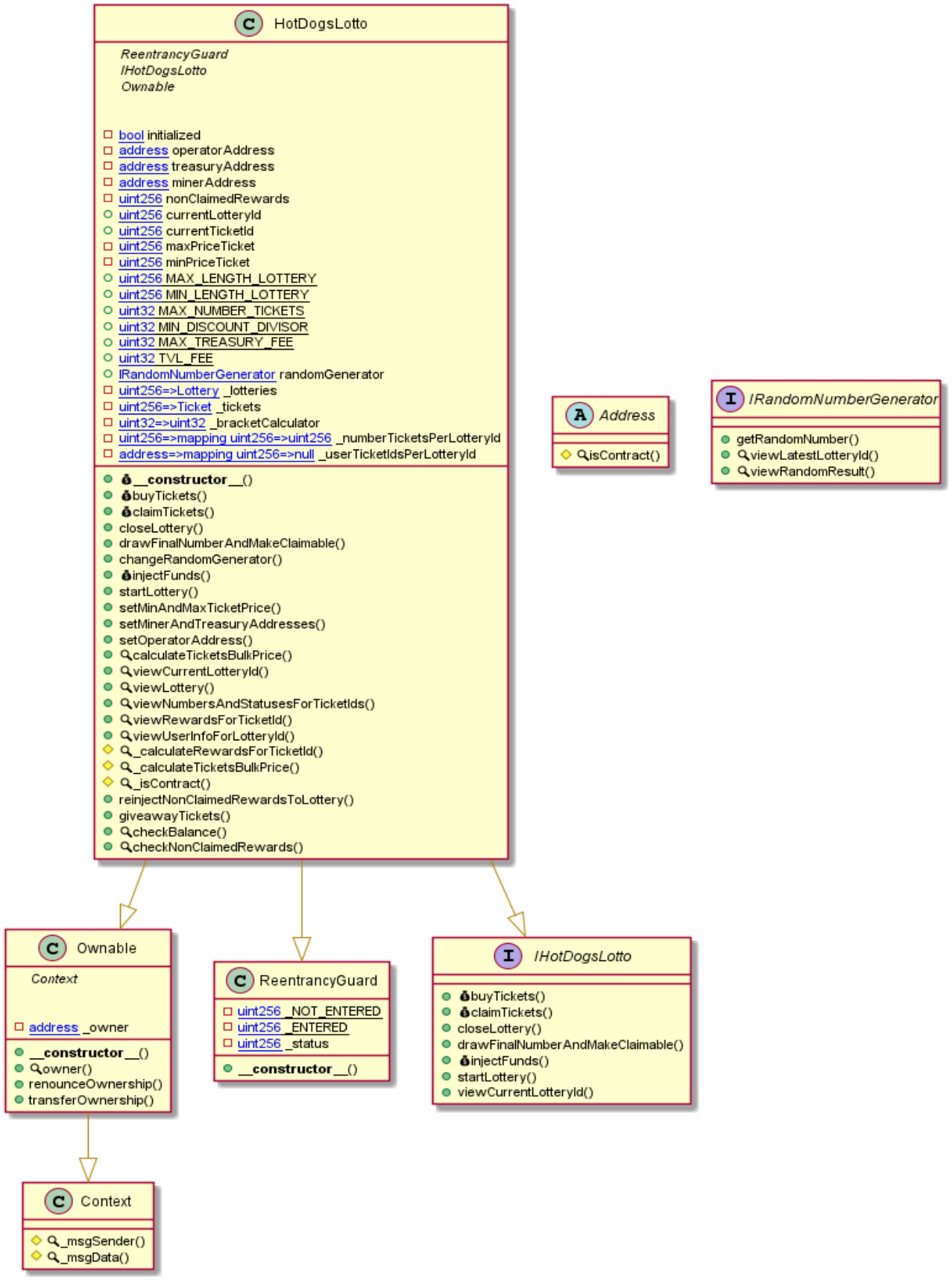
Technical Disclaimer

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.

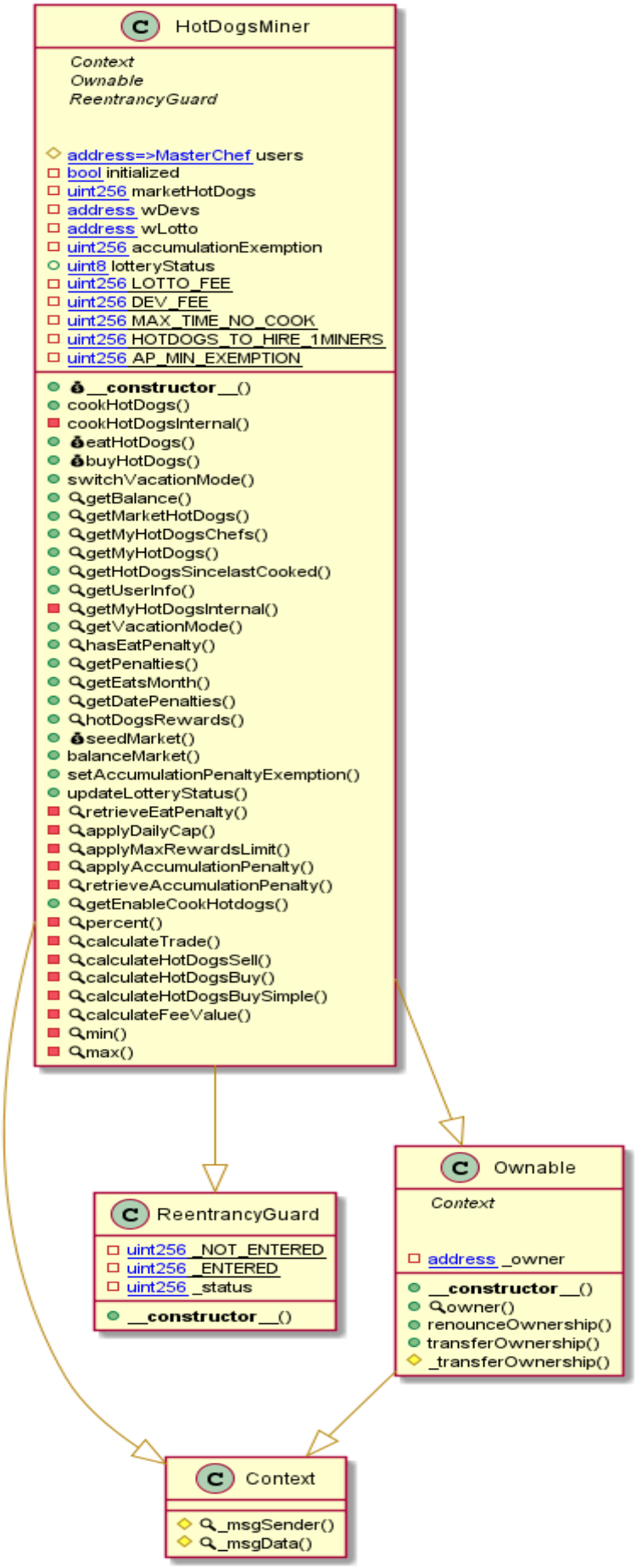
Appendix

Code Flow Diagram - HotDogs Protocol

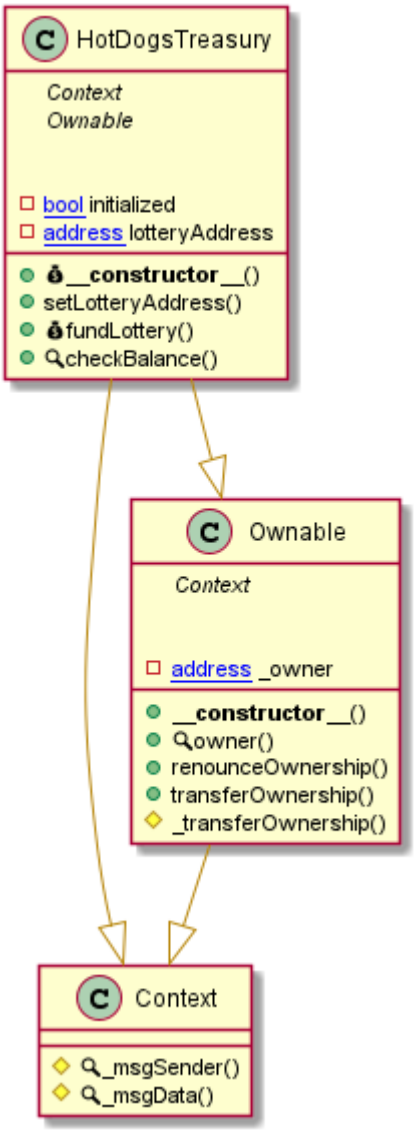
HotDogsLotto Diagram



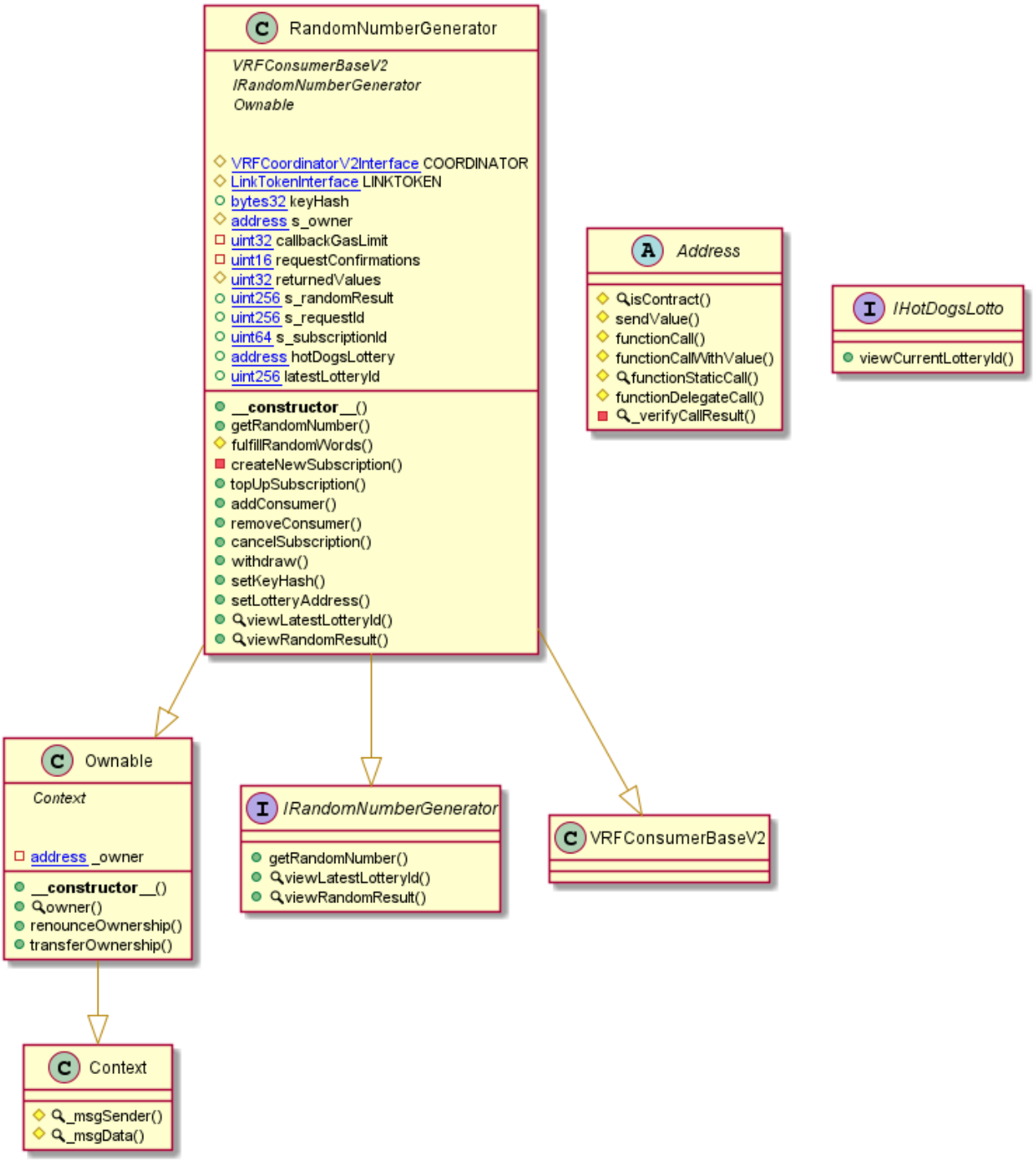
HotDogsMiner Diagram



HotDogsTreasury Diagram



RandomNumberGenerator Diagram



Slither Results Log

Slither log >> HotDogsLotto.sol

```
INFO:Detectors:
Reentrancy in HotDogsLotto.changeRandomGenerator(address) (HotDogsLotto.sol#390-400):
        External calls:
        - IRandomNumberGenerator(_randomGeneratorAddress).getRandomNumber() (HotDogsLotto.sol#393)
        State variables written after the call(s):
        - randomGenerator = IRandomNumberGenerator(_randomGeneratorAddress) (HotDogsLotto.sol#397)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in HotDogsLotto.changeRandomGenerator(address) (HotDogsLotto.sol#390-400):
        External calls:
        - IRandomNumberGenerator(_randomGeneratorAddress).getRandomNumber() (HotDogsLotto.sol#393)
        Event emitted after the call(s):
        - NewRandomGenerator(_randomGeneratorAddress) (HotDogsLotto.sol#399)
Reentrancy in HotDogsLotto.closeLottery(uint256) (HotDogsLotto.sol#322-332):
        External calls:
        - randomGenerator.getRandomNumber() (HotDogsLotto.sol#327)
        Event emitted after the call(s):
        - LotteryClose(_lotteryId,currentTicketId) (HotDogsLotto.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[_lotteryId].status == Status.Open,Lottery is not open) (HotDogsLotto.sol#229)
        - require(bool,string)(block.timestamp < _lotteries[_lotteryId].endTime,Lottery is over) (HotDogsLotto.sol#230)
HotDogsLotto.claimTickets(uint256,uint256[],uint32[]) (HotDogsLotto.sol#279-320) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[_lotteryId].status == Status.Claimable,Lottery not claimable) (HotDogsLotto.sol#287)
        - require(bool,string)(_lotteries[_lotteryId].firstTicketIdNextLottery > thisTicketId,TicketId too high) (HotDogsLotto.
sol#296)
        - require(bool,string)(_lotteries[_lotteryId].firstTicketId <= thisTicketId,TicketId too low) (HotDogsLotto.sol#297)
        - require(bool,string)(msg.sender == _tickets[thisTicketId].owner,Not the owner) (HotDogsLotto.sol#298)
        - require(bool,string)(rewardForTicketId != 0,No prize for bracket) (HotDogsLotto.sol#304)
        - require(bool,string)(_calculateRewardsForTicketId(_lotteryId,thisTicketId,_brackets[i] + 1) == 0,Bracket must be high
er) (HotDogsLotto.sol#307-310)
HotDogsLotto.closeLottery(uint256) (HotDogsLotto.sol#322-332) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[_lotteryId].status == Status.Open,Lottery not open) (HotDogsLotto.sol#323)
```

```
        Dangerous comparisons:
        - require(bool,string)(_lotteries[currentLotteryId].status == Status.Claimable,Lottery not in claimable) (HotDogsLotto.
sol#391)
HotDogsLotto.injectFunds() (HotDogsLotto.sol#402-406) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[currentLotteryId].status == Status.Open,Lottery must be open) (HotDogsLotto.sol#403)
HotDogsLotto.startLottery(uint256,uint256,uint256,uint256[6],uint256) (HotDogsLotto.sol#408-471) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((currentLotteryId == 0) || (_lotteries[currentLotteryId].status == Status.Claimable),Not time to
 start lottery) (HotDogsLotto.sol#416-419)
        - require(bool,string)(((_endTime - block.timestamp) > MIN_LENGTH_LOTTERY) && ((_endTime - block.timestamp) < MAX_LENGT
H_LOTTERY),Lottery length outside of range) (HotDogsLotto.sol#421-424)
HotDogsLotto.viewNumbersAndStatusesForTicketIds(uint256[]) (HotDogsLotto.sol#522-541) uses timestamp for comparisons
        Dangerous comparisons:
        - _tickets[_ticketIds[i]].owner == address(0) (HotDogsLotto.sol#533)
HotDogsLotto.viewRewardsForTicketId(uint256,uint256,uint32) (HotDogsLotto.sol#543-560) uses timestamp for comparisons
        Dangerous comparisons:
        - _lotteries[_lotteryId].status != Status.Claimable (HotDogsLotto.sol#548)
        - (_lotteries[_lotteryId].firstTicketIdNextLottery < _ticketId) && (_lotteries[_lotteryId].firstTicketId >= _ticketId)
(HotDogsLotto.sol#553-554)
HotDogsLotto.viewUserInfoForLotteryId(address,uint256,uint256,uint256) (HotDogsLotto.sol#562-600) uses timestamp for comparison
s
        Dangerous comparisons:
        - _tickets[lotteryTicketIds[i]].owner == address(0) (HotDogsLotto.sol#592)
HotDogsLotto._calculateRewardsForTicketId(uint256,uint256,uint32) (HotDogsLotto.sol#602-621) uses timestamp for comparisons
        Dangerous comparisons:
        - transformedWinningNumber == transformedUserNumber (HotDogsLotto.sol#616)
HotDogsLotto.reinjectNonClaimedRewardsToLottery() (HotDogsLotto.sol#639-649) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[currentLotteryId].status == Status.Open,Lottery must be open) (HotDogsLotto.sol#640)
        - require(bool,string)(block.timestamp > _lotteries[currentLotteryId - 1].endTime + 864000,At least 10 days must pass s
ince last lottery) (HotDogsLotto.sol#641)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lotteries[currentLotteryId].status == Status.Open,Lottery is not open) (HotDogsLotto.sol#653)
        - require(bool,string)((_totalAmount + _lotteries[currentLotteryId].giveawayTickets) * 25 <= currentTickets,Only 4% of
total ticket can be gifted) (HotDogsLotto.sol#661)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (HotDogsLotto.sol#70-77) uses assembly
        - INLINE ASM (HotDogsLotto.sol#73-75)
HotDogsLotto._isContract(address) (HotDogsLotto.sol#631-637) uses assembly
        - INLINE ASM (HotDogsLotto.sol#633-635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
HotDogsLotto.startLottery(uint256,uint256,uint256,uint256[6],uint256) (HotDogsLotto.sol#408-471) compares to a boolean constant
:
        -require(bool,string)(initialized == true,Treasury address is not set) (HotDogsLotto.sol#415)
HotDogsLotto.setMinerAndTreasuryAddresses(address,address) (HotDogsLotto.sol#483-493) compares to a boolean constant:
        -require(bool,string)(initialized == false,Treasury address already set) (HotDogsLotto.sol#484)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) has costly operations inside a loop:
        - ++ currentTicketId (HotDogsLotto.sol#265)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) has costly operations inside a loop:
        - ++ currentTicketId (HotDogsLotto.sol#691)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
INFO:Detectors:
Redundant expression "this (HotDogsLotto.sol#11)" inContext (HotDogsLotto.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in HotDogsLotto.claimTickets(uint256,uint256[],uint32[]) (HotDogsLotto.sol#279-320):
        External calls:
        - address(msg.sender).transfer(rewardToTransfer) (HotDogsLotto.sol#317)
        Event emitted after the call(s):
        - TicketsClaim(msg.sender,rewardToTransfer,_lotteryId,_ticketIds.length) (HotDogsLotto.sol#319)
Reentrancy in HotDogsLotto.drawFinalNumberAndMakeClaimable(uint256) (HotDogsLotto.sol#334-388):
        External calls:
        - address(treasuryAddress).transfer(amountToWithdrawToTreasury) (HotDogsLotto.sol#384)
        - address(minerAddress).transfer(amountToMiner) (HotDogsLotto.sol#385)
        Event emitted after the call(s):
        - LotteryNumberDrawn(currentLotteryId,finalNumber,numberAddressesInPreviousBracket) (HotDogsLotto.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses literals with too many digits:
        - require(bool,string)((thisTicketNumber >= 1000000) && (thisTicketNumber <= 1999999),Outside range) (HotDogsLotto.sol#
246)
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses literals with too many digits:
        - _numberTicketsPerLotteryId[_lotteryId][11111 + (thisTicketNumber % 100000)] ++ (HotDogsLotto.sol#255)
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses literals with too many digits:
        - _numberTicketsPerLotteryId[_lotteryId][111111 + (thisTicketNumber % 1000000)] ++ (HotDogsLotto.sol#258)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _random = (uint256(keccak256(bytes)(abi.encodePacked(block.timestamp,msg.sender,_count))) % 1000000) + 1000000 (HotDo
gsLotto.sol#667)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - require(bool,string)((thisTicketNumber >= 1000000) && (thisTicketNumber <= 1999999),Outside range) (HotDogsLotto.sol#
671)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _numberTicketsPerLotteryId[currentLotteryId][11111 + (thisTicketNumber % 100000)] ++ (HotDogsLotto.sol#680)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _numberTicketsPerLotteryId[currentLotteryId][111111 + (thisTicketNumber % 1000000)] ++ (HotDogsLotto.sol#683)
HotDogsLotto.slitherConstructorVariables() (HotDogsLotto.sol#115-713) uses literals with too many digits:
        - maxPriceTicket = 10000000000000000 (HotDogsLotto.sol#127)
```

```
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses literals with too many digits:
        - _numberTicketsPerLotteryId[_lotteryId][11111 + (thisTicketNumber % 100000)] ++ (HotDogsLotto.sol#255)
HotDogsLotto.buyTickets(uint256,uint32[]) (HotDogsLotto.sol#221-276) uses literals with too many digits:
        - _numberTicketsPerLotteryId[_lotteryId][111111 + (thisTicketNumber % 1000000)] ++ (HotDogsLotto.sol#258)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _random = (uint256(keccak256(bytes)(abi.encodePacked(block.timestamp,msg.sender,_count))) % 1000000) + 1000000 (HotDo
gsLotto.sol#667)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - require(bool,string)((thisTicketNumber >= 1000000) && (thisTicketNumber <= 1999999),Outside range) (HotDogsLotto.sol#
671)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _numberTicketsPerLotteryId[currentLotteryId][11111 + (thisTicketNumber % 100000)] ++ (HotDogsLotto.sol#680)
HotDogsLotto.giveawayTickets(address[],uint32[]) (HotDogsLotto.sol#651-698) uses literals with too many digits:
        - _numberTicketsPerLotteryId[currentLotteryId][111111 + (thisTicketNumber % 1000000)] ++ (HotDogsLotto.sol#683)
HotDogsLotto.slitherConstructorVariables() (HotDogsLotto.sol#115-713) uses literals with too many digits:
        - maxPriceTicket = 10000000000000000 (HotDogsLotto.sol#127)
HotDogsLotto.slitherConstructorVariables() (HotDogsLotto.sol#115-713) uses literals with too many digits:
        - minPriceTicket = 5000000000000000 (HotDogsLotto.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HotDogsLotto.sol#36-39)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HotDogsLotto.sol#41-45)
checkBalance() should be declared external:
        - HotDogsLotto.checkBalance() (HotDogsLotto.sol#705-707)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HotDogsLotto.sol analyzed (7 contracts with 75 detectors), 92 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> HotDogsMiner.sol

```
INFO:Detectors:
Context._msgData() (HotDogsMiner.sol#10-12) is never used and should be removed
HotDogsMiner.applyDailyCap(address,uint256,uint256) (HotDogsMiner.sol#228-229) is never used and should be removed
HotDogsMiner.applyMaxRewardsLimit(address,uint256,uint256) (HotDogsMiner.sol#231-233) is never used and should be removed
HotDogsMiner.calculateFeeValue(uint256,uint256) (HotDogsMiner.sol#266-268) is never used and should be removed
HotDogsMiner.calculateHotDogsBuy(uint256,uint256) (HotDogsMiner.sol#258-260) is never used and should be removed
HotDogsMiner.calculateHotDogsBuySimple(uint256) (HotDogsMiner.sol#262-264) is never used and should be removed
HotDogsMiner.max(uint256,uint256) (HotDogsMiner.sol#274-276) is never used and should be removed
HotDogsMiner.min(uint256,uint256) (HotDogsMiner.sol#270-272) is never used and should be removed
HotDogsMiner.percent(uint256,uint256,uint256) (HotDogsMiner.sol#243-248) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (HotDogsMiner.sol#3) 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:
Parameter HotDogsMiner.setAccumulationPenaltyExemption(uint256)._accumulationExemption (HotDogsMiner.sol#201) is not in mixedCa
Parameter HotDogsMiner.updateLotteryStatus(uint8)._value (HotDogsMiner.sol#210) is not in mixedCase
Parameter HotDogsMiner.calculateFeeValue(uint256,uint256)._amount (HotDogsMiner.sol#266) is not in mixedCase
Parameter HotDogsMiner.calculateFeeValue(uint256,uint256)._fee (HotDogsMiner.sol#266) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
HotDogsMiner.seedMarket() (HotDogsMiner.sol#191-196) uses literals with too many digits:
        - marketHotDogs = 165500000000 (HotDogsMiner.sol#194)
HotDogsMiner.slitherConstructorVariables() (HotDogsMiner.sol#75-278) uses literals with too many digits:
        - accumulationExemption = 10000000000000000 (HotDogsMiner.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
HotDogsMiner.wLotto (HotDogsMiner.sol#84) is never used in HotDogsMiner (HotDogsMiner.sol#75-278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (HotDogsMiner.sol#49-51)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HotDogsMiner.sol#59-62)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HotDogsMiner.sol#64-66)
```

```
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HotDogsMiner.sol#64-66)
getHotDogsSincelastCooked(address) should be declared external:
        - HotDogsMiner.getHotDogsSincelastCooked(address) (HotDogsMiner.sol#141-142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HotDogsMiner.sol analyzed (4 contracts with 75 detectors), 26 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> HotDogsTreasury.sol

```
INFO:Detectors:
Reentrancy in HotDogsTreasury.fundLottery() (HotDogsTreasury.sol#77-85):
        External calls:
        - (success) = address(lotteryAddress).call{value: _amount}() (HotDogsTreasury.sol#81)
        Event emitted after the call(s):
        - FundLottery(_amount) (HotDogsTreasury.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
HotDogsTreasury.setLotteryAddress(address) (HotDogsTreasury.sol#64-74) compares to a boolean constant:
        -require(bool,string)(initialized == false,Address cannot be changed) (HotDogsTreasury.sol#67)
HotDogsTreasury.fundLottery() (HotDogsTreasury.sol#77-85) compares to a boolean constant:
        -require(bool,string)(initialized == true,Must set the Lottery address first) (HotDogsTreasury.sol#78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (HotDogsTreasury.sol#8-10) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (HotDogsTreasury.sol#1) 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:
Low level call in HotDogsTreasury.fundLottery() (HotDogsTreasury.sol#77-85):
        - (success) = address(lotteryAddress).call{value: _amount}() (HotDogsTreasury.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter HotDogsTreasury.setLotteryAddress(address)._lotteryAddress (HotDogsTreasury.sol#65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (HotDogsTreasury.sol#24-26)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HotDogsTreasury.sol#34-37)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HotDogsTreasury.sol#39-41)
checkBalance() should be declared external:
        - HotDogsTreasury.checkBalance() (HotDogsTreasury.sol#88-90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HotDogsTreasury.sol analyzed (3 contracts with 75 detectors), 12 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> RandomNumberGenerator.sol

```
INFO:Detectors:
RandomNumberGenerator.setLotteryAddress(address)._hotDogsLottery (RandomNumberGenerator.sol#334) lacks a zero-check on :
                - hotDogsLottery = _hotDogsLottery (RandomNumberGenerator.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (RandomNumberGenerator.sol#48-55) uses assembly
        - INLINE ASM (RandomNumberGenerator.sol#51-53)
Address._verifyCallResult(bool,bytes,string) (RandomNumberGenerator.sol#127-145) uses assembly
        - INLINE ASM (RandomNumberGenerator.sol#137-140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (RandomNumberGenerator.sol#127-145) is never used and should be removed
Address.functionCall(address,bytes) (RandomNumberGenerator.sol#64-66) is never used and should be removed
Address.functionCall(address,bytes,string) (RandomNumberGenerator.sol#68-74) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (RandomNumberGenerator.sol#76-82) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (RandomNumberGenerator.sol#84-95) is never used and should be remov
ed
Address.functionDelegateCall(address,bytes) (RandomNumberGenerator.sol#112-114) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (RandomNumberGenerator.sol#116-125) is never used and should be removed
Address.functionStaticCall(address,bytes) (RandomNumberGenerator.sol#97-99) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (RandomNumberGenerator.sol#101-110) is never used and should be removed
Address.isContract(address) (RandomNumberGenerator.sol#48-55) is never used and should be removed
Address.sendValue(address,uint256) (RandomNumberGenerator.sol#57-62) is never used and should be removed
Context._msgData() (RandomNumberGenerator.sol#9-12) is never used and should be removed
VRFConsumerBaseV2.fulfillRandomWords(uint256,uint256[]) (RandomNumberGenerator.sol#243) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (RandomNumberGenerator.sol#3) 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:
RandomNumberGenerator.fulfillRandomWords(uint256,uint256[]) (RandomNumberGenerator.sol#293-299) uses literals with too many dig
its:
        - s_randomResult = (randomWords[0] % 1000000) + 1000000 (RandomNumberGenerator.sol#297)
RandomNumberGenerator.slitherConstructorVariables() (RandomNumberGenerator.sol#252-347) uses literals with too many digits:
        - callbackGasLimit = 100000 (RandomNumberGenerator.sol#258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
RandomNumberGenerator.callbackGasLimit (RandomNumberGenerator.sol#258) should be constant
RandomNumberGenerator.requestConfirmations (RandomNumberGenerator.sol#260) should be constant
RandomNumberGenerator.returnedValues (RandomNumberGenerator.sol#262) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (RandomNumberGenerator.sol#35-38)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (RandomNumberGenerator.sol#40-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:RandomNumberGenerator.sol analyzed (9 contracts with 75 detectors), 43 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

HotDogsLotto.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: 191:30:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HotDogsLotto.claimTickets(uint256,uint256[],uint32[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 292:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HotDogsLotto.changeRandomGenerator(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 403:4:

Gas & Economy

Gas costs:

Gas requirement of function HotDogsLotto.buyTickets 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: 234:4:

Gas costs:

Gas requirement of function HotDogsLotto.claimTickets 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: 292:4:

Email: audit@EtherAuthority.io

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: 677:2:

Miscellaneous

Constant/View/Pure functions:

HotDogsLotto.viewNumbersAndStatusesForTicketIds(uint256[]) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 535:4:

Similar variable names:

HotDogsLotto.startLottery(uint256,uint256,uint256,uint256[6],uint256) : Variables have very similar names "maxPriceTicket" and "minPriceTicket". Note: Modifiers are currently not considered by this static analysis.
Pos: 440:65:

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: 490:8:

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: 714:2:

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: 641:15:

HotDogsMiner.sol

Security

Check-effects-interaction:

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

more

Pos: 168:4:

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: 631:7:

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: 241:22:

Gas & Economy

Gas costs:

Gas requirement of function HotDogsMiner.balanceMarket 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: 470:1:

Gas costs:

Gas requirement of function HotDogsMiner.getEnableCookHotdogs 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: 629:2:

Miscellaneous

Constant/View/Pure functions:

HotDogsMiner.applyMaxRewardsLimit(address,uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 558:2:

Similar variable names:

HotDogsMiner.getPenalties(address) : Variables have very similar names "eatPenalty" and "nextPenalty". Note: Modifiers are currently not considered by this static analysis.
Pos: 422:3:

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: 542:3:

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: 648:3:

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: 671:16:

HotDogsTreasury.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HotDogsTreasury.fundLottery(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 77:4:

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: 81:27:

Gas & Economy

Gas costs:

Gas requirement of function HotDogsTreasury.setLotteryAddress 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: 64:1:

Gas costs:

Gas requirement of function HotDogsTreasury.fundLottery 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: 77:4:

Miscellaneous

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: 30:6:

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: 78:8:

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: 82:5:

RandomNumberGenerator.sol

Security

Check-effects-interaction:

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

more

Pos: 247:4:

Gas & Economy

Gas costs:

Gas requirement of function RandomNumberGenerator.setKeyHash 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: 257:4:

ERC

ERC20:

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

more

Pos: 11:2:

Miscellaneous

Constant/View/Pure functions:

IHotDogsLotto.viewCurrentLotteryId() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 171:4:

Similar variable names:

RandomNumberGenerator.(address,address,bytes32) : Variables have very similar names "s_owner" and "_owner". Note: Modifiers are currently not considered by this static analysis.

Pos: 205:8:

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: 211:8:

Solhint Linter

HotDogsLotto.sol

```
HotDogsLotto.sol:277:21: Error: Parse error: missing ';' at '{'
HotDogsLotto.sol:703:13: Error: Parse error: missing ';' at '{'
```

HotDogsMiner.sol

```
HotDogsMiner.sol:1:1: Error: Compiler version 0.8.14 does not satisfy
the r semver requirement
HotDogsMiner.sol:13:1: Error: Compiler version 0.8.14 does not
satisfy the r semver requirement
HotDogsMiner.sol:22:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
HotDogsMiner.sol:43:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
HotDogsMiner.sol:93:2: Error: Explicitly mark visibility of state
HotDogsMiner.sol:119:2: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
HotDogsMiner.sol:135:13: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:138:6: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:140:39: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:146:33: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:168:40: Error: Visibility modifier must be first in
list of modifiers
HotDogsMiner.sol:175:6: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:177:39: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:183:33: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:194:33: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:201:40: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:203:40: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:241:23: Error: Avoid using low level calls.
HotDogsMiner.sol:251:51: Error: Visibility modifier must be first in
list of modifiers
HotDogsMiner.sol:255:39: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:280:9: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:281:35: Error: Avoid to make time-based decisions in
your business logic
```

```
HotDogsMiner.sol:290:22: Error: Avoid using low level calls.
HotDogsMiner.sol:299:12: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:303:41: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:306:9: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:307:35: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:332:42: Error: Visibility modifier must be first in
list of modifiers
HotDogsMiner.sol:337:25: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:353:27: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:356:48: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:406:43: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:406:119: Error: Avoid to make time-based decisions
in your business logic
HotDogsMiner.sol:432:6: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:446:7: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:515:25: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:519:32: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:611:27: Error: Avoid to make time-based decisions in
your business logic
HotDogsMiner.sol:631:7: Error: Avoid to make time-based decisions in
your business logic
```

HotDogsTreasury.sol

```
HotDogsTreasury.sol:1:1: Error: Compiler version 0.8.14 does not
satisfy the r semver requirement
HotDogsTreasury.sol:18:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
HotDogsTreasury.sol:60:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
HotDogsTreasury.sol:60:19: Error: Code contains empty blocks
HotDogsTreasury.sol:81:28: Error: Avoid using low level calls.
```

RandomNumberGenerator.sol

```
RandomNumberGenerator.sol:174:1: Error: Import statements must be on
top
RandomNumberGenerator.sol:175:1: Error: Import statements must be on
top
RandomNumberGenerator.sol:176:1: Error: Import statements must be on
```

```
top
RandomNumberGenerator.sol:2:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:16:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:23:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
RandomNumberGenerator.sol:51:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:57:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
RandomNumberGenerator.sol:66:28: Error: Avoid using low level calls.
RandomNumberGenerator.sol:99:51: Error: Avoid using low level calls.
RandomNumberGenerator.sol:129:51: Error: Avoid using low level calls.
RandomNumberGenerator.sol:143:17: Error: Avoid using inline assembly.
It is acceptable only in rare cases
RandomNumberGenerator.sol:157:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:168:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:179:1: Error: Compiler version 0.8.14 does
not satisfy the r semver requirement
RandomNumberGenerator.sol:182:5: Error: Explicitly mark visibility of
state
RandomNumberGenerator.sol:182:31: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:183:5: Error: Explicitly mark visibility of
state
RandomNumberGenerator.sol:183:24: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:185:5: Error: Explicitly mark visibility of
state
RandomNumberGenerator.sol:185:13: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:191:5: Error: Explicitly mark visibility of
state
RandomNumberGenerator.sol:193:20: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:194:20: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:195:19: Error: Variable name must be in
mixedCase
RandomNumberGenerator.sol:201:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

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