Neon Labs - Chainlink-Neon

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 20th, 2023 - February 22nd, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 4
CONTACTS 5
1 EXECUTIVE OVERVIEW 6
1.1 INTRODUCTION 7
1.2 AUDIT SUMMARY 7
1.3 TEST APPROACH & METHODOLOGY 8
RISK METHODOLOGY 8
1.4 SCOPE 10
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 11
3 FINDINGS & TECH DETAILS 12
3.1 (HAL-01) THE GETROUNDDATA FUNCTION REVERTS DUE TO UNDERFLOW - HIGH 14
Description 14
Code Location 15
Proof Of Concept 16
Recommendation 16
Remediation Plan 16
3.2 (HAL-02) MISSING CHECK FOR 0 TIMESTAMP - MEDIUM 17
Description 17
Code Location 17
Scenario 18
Recommendation 19
Remediation Plan 19
3.3 (HAL-03) FLOATING PRAGMA - LOW 20
Description 20

Code Location 20

Risk Level 20

Recommendation 20

Remediation Plan 21

3.4 (HAL-04) FOR LOOP CAN BE OPTIMIZED - INFORMATIONAL 22

Description 22

Code Location 22

Risk Level 23

Recommendation 23

Remediation Plan 23

3.5 (HAL-05) USE CALLDATA INSTEAD OF MEMORY FOR IMMUTABLE FUNCTION ARGUMENTS - INFORMATIONAL 24

Description 24

Code Location 24

Risk Level 24

Recommendation 24

Remediation Plan 24

3.6 (HAL-06) PACKING STORAGE VARIABLES INCREASES GAS EFFICIENCY - INFORMATIONAL 25

Description 25

Code Location 25

Risk Level 26

Recommendation 26

Remediation Plan 27

4 AUTOMATED TESTING 28

4.1 STATIC ANALYSIS REPORT 29

Description 29

Results 30
4.2 AUTOMATED SECURITY SCAN 31
Description 31
Results 32

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/21/2023	Josep Bove
0.2	Document Updates	02/22/2023	Francisco González
0.3	Document Updates	02/23/2023	Josep Bove
0.4	Draft Review	02/23/2023	Ataberk Yavuzer
0.5	Draft Review	02/23/2023	Piotr Cielas
0.6	Draft Review	02/23/2023	Gabi Urrutia
1.0	Remediation Plan	03/14/2023	Francisco González
1.1	Remediation Plan Review	03/15/2023	Ataberk Yavuzer
1.2	Remediation Plan Edits	03/15/2023	Francisco González
1.3	Remediation Plan Review	03/15/2023	Ataberk Yavuzer

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Josep Bove	Halborn	Josep.Bove@halborn.com
Francisco González	Halborn	Francisco.Villarejo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Chainlink-Neon introduces the Chainlink Solana data feed to Neon EVM enabling users to query the oracle.

Neon Labs engaged Halborn to conduct a security audit on their smart contracts beginning on February 20th, 2023 and ending on February 22nd, 2023 . The security assessment was scoped to the smart contracts provided in the NeonLabs - Chainlink GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided 3 days for the engagement and assigned 2 full-time security engineers to audit the security of the smart contracts in scope. The security engineers are blockchain and smart contract security experts with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the audits is to:

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

In summary, Halborn identified some security risks that were mostly addressed by the Neon Labs team. The main ones are the following:

- Handling reverts from `getRoundData()` function in case no historical data is present in the data feed.
- Checking that the returned `timestamp` value is not `0`.
- Locking pragma versions.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

- Repository: chainlink-neon
- Commit ID: 4dfd4beba6869a122f6bbb3c8b1bb2bcc2e91dbf
- Smart contracts in scope:

    1. (contracts/ChainlinkOracle.sol)
    2. (contracts/libraries/Utils.sol)
    3. (contracts/libraries/external/QueryAccount.sol)

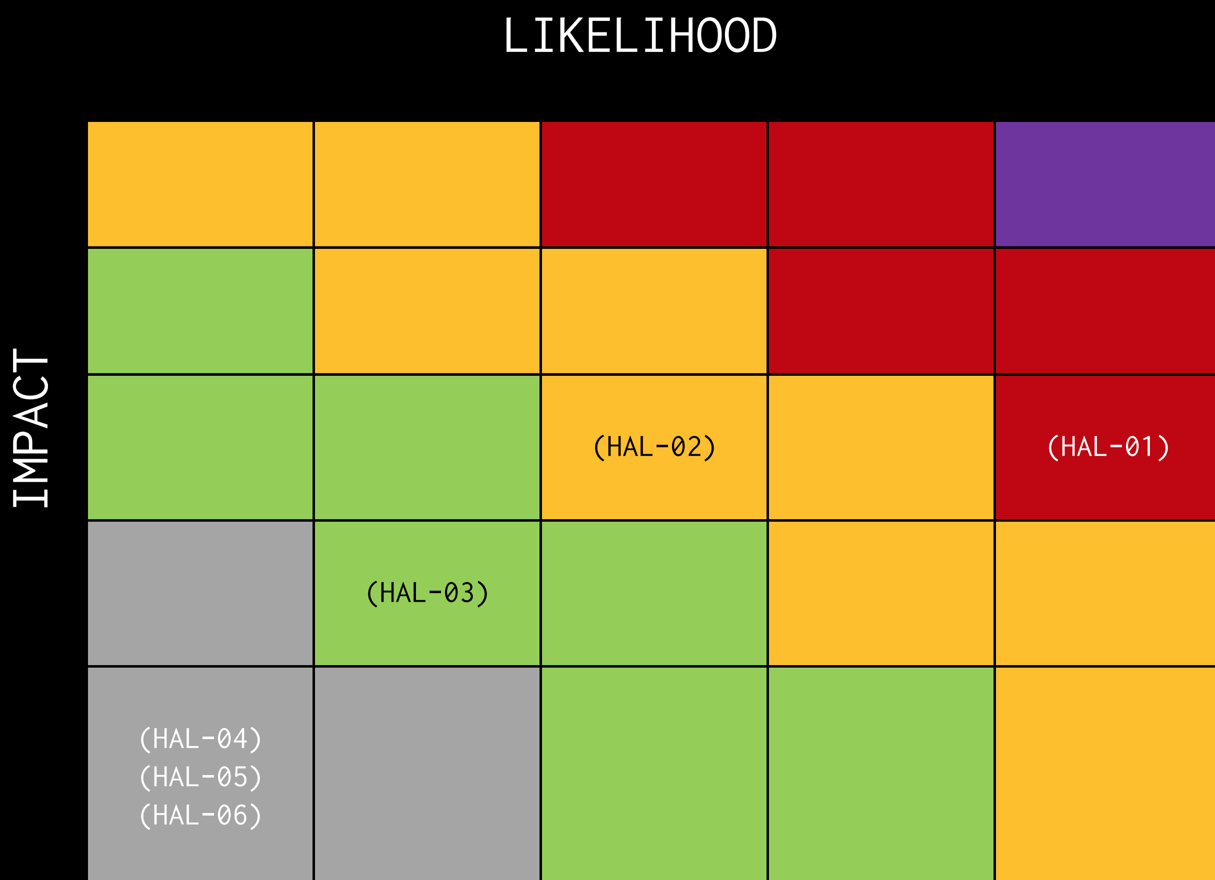
- Remediations Commit ID: e7b8f479c7105bae5245e11851acbe2a99b7c9e9

Out-of-scope:

- third-party libraries and dependencies
- economic attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	1	3



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) THE GETROUNDDATA FUNCTION REVERTS DUE TO UNDERFLOW	High	SOLVED - 03/14/2023
(HAL-02) MISSING CHECK FOR 0 TIMESTAMP	Medium	SOLVED - 03/14/2023
(HAL-03) FLOATING PRAGMA	Low	SOLVED - 03/14/2023
(HAL-04) FOR LOOP CAN BE OPTIMIZED	Informational	ACKNOWLEDGED
(HAL-05) USE CALLDATA INSTEAD OF MEMORY FOR IMMUTABLE FUNCTION ARGUMENTS	Informational	ACKNOWLEDGED
(HAL-06) PACKING STORAGE VARIABLES INCREASES GAS EFFICIENCY	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) THE GETROUNDDATA FUNCTION REVERTS DUE TO UNDERFLOW - HIGH

Description:

To retrieve Oracle prices, ChainlinkOracle contract implements two different functions, getRoundData(), and latestRoundData(). While the latter retrieves the latest price data available from the data feed, the getRoundData() function allows users to retrieve historical price data at a specific roundId.

Although latestRoundData() successfully retrieves valid and fresh data from the selected data feed, it was detected that calls to getRoundData() always revert due to an integer underflow for any of the valid data fees found in Chainlink's documentation.

When getRoundData() function is called, a subcall to Utils.sol's locateRound() function is performed. This function takes multiple input parameters, including historicalLength, the most relevant one for this finding. historicalLength is calculated in the Utils.sol library with the getHistoricalLength() function, which takes feedAddress and liveLength as input parameters.

For every data feed address tested on both Solana Devnet and Mainnet, liveLength is set to 1, which returns a historicalLength value of 0.

This causes the underflow in locateRound() function, since (historicalLength - 1) is used to calculate historicalStartRoundId, causing every getRoundData() call for every observed data feed to revert.

Code Location:

Listing 1: contracts/libraries/Utils.sol (Line 130)

```
112     function locateRound(
113         uint32 roundId,
114         uint32 liveCursor,
115         uint32 liveLength,
116         uint32 latestRoundId,
117         uint32 historicalCursor,
118         uint32 historicalLength,
119         uint8 granularity
120     )
121         public
122         pure
123         returns (
124             uint32 position,
125             uint32 correctedRoundId
126         )
127     {
128         uint32 liveStartRoundId = saturatingSub(latestRoundId, liveLength - 1);
129         uint32 historicalEndRoundId = latestRoundId - (latestRoundId % granularity);
130         uint32 historicalStartRoundId = saturatingSub(historicalEndRoundId, granularity * (historicalLength - 1));
```

Proof Of Concept:

For this PoC, ETH/USD data feed from Chainlink docs will be used. A new ChainlinkOracle was deployed on Neon Devnet at address 0x471469AE55a3A22b0FA81a7e296e72AFc3A4E25a and is used as a price oracle.

Once is set up, calls to latestRoundData() and getRoundData() are performed:

```
>>> underflowPoC()
Using ETH/USD data feed for this PoC.

Successfull call to latestRoundData() --> contract_OracleWETH.latestRoundData(): (3540523, 162019000000, 1677099831, 1677099831, 3540523)

Retrieving WETH/USD data feed address --> contract_OracleWETH.feedAddress(): 34198935755373527536852904497581405559534967731673328705110581177736424914282
Retrieving Retrieving WETH/USD data feed Historical Length --> library_Utils.getHistoricalLength(contract_OracleWETH.feedAddress(), 1): 0

Trying to retrieve historical data from valid round 3,540,485 --> contract_OracleWETH.getRoundData(3540485)
  File "<console>", line 1, in <module>
  File "<console>", line 11, in underflowPoC
  File "brownie/network/contract.py", line 1905, in __call__
    return self.call(*args, block_identifier=block_identifier, override=override)
  File "brownie/network/contract.py", line 1696, in call
    raise VirtualMachineError(e) from None
  File "brownie/exceptions.py", line 106, in __init__
    raise ValueError(f"{self.message}: {err_msg}") from None
ValueError: execution reverted: 0x4e487b710000000000000000000000000000000000000000000000000000000000000011
```

Recommendation:

If retrieving historical price data from data feeds is an intended feature, adjusting the logic in the Utils.sol library to adjust to Chainlink's data feeds parameters is strongly recommended.

Remediation Plan:

SOLVED: The Neon Labs team solved this issue by gracefully handling requests for historical data for data feeds with historicalLength value of 0. It must be noted that every data feed from Chainlink's documentation has an historicalLength value of 0 at the time of writing this report.

Commit ID: e7b8f479c7105bae5245e11851acbe2a99b7c9e9

3.2 (HAL-02) MISSING CHECK FOR 0 TIMESTAMP - MEDIUM

Description:

Chainlink oracle return values are not handled properly. priceFeed includes the following parameters:
- roundId
- answer
- startedAt
- updatedAt
- answeredInRound

All those parameters should be sanity checked before updating the price. In this case, an incorrect price can be reported when timestamp == 0.

Code Location:

Listing 2: contracts/ChainlinkOracle.sol

```
function getRoundData(uint80 _roundId)
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        )
    {
        Utils.Round memory round = Utils.getRoundbyId(feedAddress,
 uint32(_roundId), historicalLength);

        return (
            round.roundId,
            round.answer,
            round.timestamp,
            round.timestamp,
```

```
            round.roundId
        );
    }
```

Listing 3: contracts/ChainlinkOracle.sol

```
function latestRoundData()
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        )
    {
        Utils.Round memory round = Utils.getLatestRound(
↳ feedAddress);

        return (
            round.roundId,
            round.answer,
            round.timestamp,
            round.timestamp,
            round.roundId
        );
    }
}
```

Scenario:

lastestRoundDate() calls a Chainlink oracle receiving the latestRoundData(). If there is a problem with Chainlink starting a new round and reaching consensus on the new value for the oracle (e.g. Chainlink nodes abandon the oracle, chain congestion, vulnerability/attacks on the Chainlink system) consumers of this contract may continue using incorrect data (if oracles are unable to submit no new round is started).

Recommendation:

Add a require to check if the price is on stale and revert both on line 37 and line 59.

Listing 4: contracts/ChainlinkOracle.sol

```
"require(
  timeStamp != 0,
  ""ChainlinkOracle::getLatestAnswer: round is not complete"
"
);"
```

Remediation Plan:

SOLVED: The Neon Labs team solved this issue by adding a require() statement to ensure that the returned timestamp is not 0.

Commit ID: 955c24d061275d3c17eb6ea0b505c5b177f97455

3.3 (HAL-03) FLOATING PRAGMA - LOW

Description:

The project contains many instances of floating pragma. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, either an outdated compiler version that might introduce bugs that affect the contract system negatively or a pragma version too recent which has not been extensively tested.

Code Location:

- ChainlinkOracle.sol#L2
- Utils.sol#L2
- QueryAccount.sol#L3

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider locking the pragma version with known bugs for the compiler version by removing the caret (^) symbol and using the same version for all contracts. When possible, do not use floating pragma in the final live deployment. Specifying a fixed compiler version ensures that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Remediation Plan:

SOLVED: The Neon Labs team solved this issue by locking pragma version to 0.8.19.

Commit ID: 4533b0f099dbfe9975933ccb5bc835c3749bf52a

3.4 (HAL-04) FOR LOOP CAN BE OPTIMIZED - INFORMATIONAL

Description:

It was observed all for loops in the protocol were not optimized properly. Suboptimal for loops can cost too much gas usage.

These for loops can be optimized as follows:

1. In Solidity (pragma 0.8.0 and later), adding unchecked keyword for arithmetical operations can reduce gas usage on contracts where underflow/underflow is unrealistic. It is possible to save gas by using this keyword on multiple code locations.
2. In all for loops, the index variable is incremented using +=. It is known that, in loops, using ++i costs less gas per iteration than +=. This also affects incremented variables within the loop code block.
3. Do not initialize index variables with 0, Solidity already initializes these uint variables as zero.

Check the Recommendation section for further details.

Code Location:

Listing 5: Utils.sol

```
for (uint8 i = 0; i < length; i++) {
    bytesArray[i] = _bytes[i];
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to apply the following pattern for Solidity pragma version 0.8.0 and later.

Listing 6: Possible Suggestion

```
for (uint256 i; i < arrayLength; ) {
    . . .
    unchecked {
     ++i
    }
```

Remediation Plan:

ACKNOWLEDGED: The Neon Labs team acknowledged this issue.

3.5 (HAL-05) USE CALLDATA INSTEAD OF MEMORY FOR IMMUTABLE FUNCTION ARGUMENTS - INFORMATIONAL

Description:

Mark data types as calldata instead of memory where possible. This makes it so that the data is not automatically loaded into memory. If the data passed into the function does not need to be changed (like updating values in an array), it can be passed in as calldata. The one exception to this is if the argument must later be passed into another function that takes an argument that specifies memory storage.

Code Location:

QueryAccount.sol:78
QueryAccount.sol:84
Utils.sol:181
Utils.sol:187
Utils.sol:200

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to mark the data type as calldata instead of memory.

Remediation Plan:

ACKNOWLEDGED: The Neon Labs team acknowledged this issue.

3.6 (HAL-06) PACKING STORAGE VARIABLES INCREASES GAS EFFICIENCY - INFORMATIONAL

Description:

In Ethereum, you pay gas for every storage slot you use. A slot is of 256 bits, and you can pack as many variables as you can try to fit. Packing is done by the Solidity compiler and optimizer automatically.

Storage variables should be strictly packed to consume less gas as a best-practice.

Code Location:

Listing 7: Utils.sol

```
struct Round {
        uint32 roundId;
        int128 answer;
        uint32 timestamp;
    }
```

Listing 8: Utils.sol

```
struct Header {
    uint8 decimals;
    string description;
    uint8 version;
    uint32 latestRoundId;
    uint32 liveLength;
    uint32 liveCursor;
    uint32 historicalCursor;
    uint8 granularity;
}
```

Listing 9: ChainlinkOracle.sol

```
uint8 public decimals;
uint256 public feedAddress;
string public description;
uint256 public version;
uint32 private historicalLength;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider packing those variables as suggested below:

Listing 10: ChainlinkOracle.sol

```
uint8 public decimals;
uint32 private historicalLength;
string public description;
uint256 public feedAddress;
uint256 public version;
```

Listing 11: Utils.sol

```
struct Round {
    uint32 roundId;
    uint32 timestamp;
    int128 answer;
}
```

Listing 12: Utils.sol

```
struct Header {
    uint8 decimals;
    uint8 granularity;
    uint8 version;
```

```
    uint32 latestRoundId;
    uint32 liveLength;
    uint32 liveCursor;
    uint32 historicalCursor;
    string description;
}
```

Remediation Plan:

ACKNOWLEDGED: The Neon Labs team acknowledged this issue.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIs and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

```
BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-226) performs a multiplication on the result of a division:
	- sstore(uint256,uint256)(_preBytes,fslot_concatStorage_asm_0 + mload(uint256)(_postBytes + 0x20) / 0x100 ** 32 - mlength_concatStorage_asm_0 * 0x100 ** 32 - newlength_concatStorage_asm_0 + mlength_co
ncatStorage_asm_0 * 2) (solidity-bytes-utils/contracts/BytesLib.sol#115-140)
BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-226) performs a multiplication on the result of a division:
	- sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (solidity-bytes-utils/contracts/BytesLib.sol#189)
BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-226) performs a multiplication on the result of a division:
	- sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (solidity-bytes-utils/contracts/BytesLib.sol#223)
BytesLib.equalStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#439-509) performs a multiplication on the result of a division:
	- fslot_equalStorage_asm_0 = fslot_equalStorage_asm_0 / 0x100 * 0x100 (solidity-bytes-utils/contracts/BytesLib.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sc_concatStorage_asm_0 = keccak256(uint256,uint256)(0x0,0x20) + slength_concatStorage_asm_0 / 32 (solidity-bytes-utils/contracts/BytesLib.sol#195)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#161)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#
91-226) potentially used before declaration: submod_concatStorage_asm_0 = 32 - slengthmod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#204)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#205)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#161)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#
91-226) potentially used before declaration: mc_concatStorage_asm_0 = _postBytes + submod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#205)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#163)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-
226) potentially used before declaration: end_concatStorage_asm_0 = _postBytes + mlength_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#206)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#164)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91
-226) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStorage_asm_0 - 1 (solidity-bytes-utils/contracts/BytesLib.sol#207)
Variable 'BytesLib.concatStorage(bytes,bytes).submod_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#161)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#
91-226) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** submod_concatStorage_asm_0 - 1 (solidity-bytes-utils/contracts/BytesLib.sol#207)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (solidity-by
tes-utils/contracts/BytesLib.sol#209)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#164)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91
-226) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (solidity-
bytes-utils/contracts/BytesLib.sol#209)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,sload(uint256)(sc_concatStorage_asm_0) + mload(uint256)(mc_concatStorage_asm_0) & mask_concatStorage_asm_0) (solidity-by
tes-utils/contracts/BytesLib.sol#209)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (solidity-bytes-utils/contracts/BytesLib.sol#212)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (solidity-bytes-utils/contracts/BytesLib.sol#213)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#214)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#163)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-
226) potentially used before declaration: mc_concatStorage_asm_0 < end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#214)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#221)
Variable 'BytesLib.concatStorage(bytes,bytes).end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#163)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-
226) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#221)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#164)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91
-226) potentially used before declaration: mask_concatStorage_asm_0 = 0x100 ** mc_concatStorage_asm_0 - end_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#221)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (solidity-bytes-utils/cont
racts/BytesLib.sol#223)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (solidity-bytes-utils/cont
racts/BytesLib.sol#223)
```

```
racts/BytesLib.sol#223)
Variable 'BytesLib.concatStorage(bytes,bytes).mask_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#164)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91
-226) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0) / mask_concatStorage_asm_0 * mask_concatStorage_asm_0) (solidity-bytes-utils/co
ntracts/BytesLib.sol#223)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0)) (solidity-bytes-utils/contracts/BytesLib.sol#218)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sstore(uint256,uint256)(sc_concatStorage_asm_0,mload(uint256)(mc_concatStorage_asm_0)) (solidity-bytes-utils/contracts/BytesLib.sol#218)
Variable 'BytesLib.concatStorage(bytes,bytes).sc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#147)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: sc_concatStorage_asm_0 = sc_concatStorage_asm_0 + 1 (solidity-bytes-utils/contracts/BytesLib.sol#215)
Variable 'BytesLib.concatStorage(bytes,bytes).mc_concatStorage_asm_0 (solidity-bytes-utils/contracts/BytesLib.sol#162)' in BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-2
26) potentially used before declaration: mc_concatStorage_asm_0 = mc_concatStorage_asm_0 + 0x20 (solidity-bytes-utils/contracts/BytesLib.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

BytesLib.concat(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#13-89) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#23-86)
BytesLib.concatStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#91-226) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#92-225)
BytesLib.slice(bytes,uint256,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#228-295) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#242-292)
BytesLib.toAddress(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#297-306) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#301-303)
BytesLib.toUint8(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#308-317) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#312-314)
BytesLib.toUint16(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#319-328) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#323-325)
BytesLib.toUint32(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#330-339) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#334-336)
BytesLib.toUint64(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#341-350) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#345-347)
BytesLib.toUint96(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#352-361) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#356-358)
BytesLib.toUint128(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#363-372) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#367-369)
BytesLib.toUint256(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#374-383) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#378-380)
BytesLib.toBytes32(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#385-394) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#389-391)
BytesLib.equal(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#396-437) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#399-434)
BytesLib.equalStorage(bytes,bytes) (solidity-bytes-utils/contracts/BytesLib.sol#439-509) uses assembly
	- INLINE ASM (solidity-bytes-utils/contracts/BytesLib.sol#449-506)
QueryAccount.to_uint256(bytes) (libraries/external/QueryAccount.sol#78-82) uses assembly
	- INLINE ASM (libraries/external/QueryAccount.sol#79-81)
QueryAccount.to_bool(bytes) (libraries/external/QueryAccount.sol#84-88) uses assembly
	- INLINE ASM (libraries/external/QueryAccount.sol#85-87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
	- Version used: ['>=0.7.0', '>=0.8.0<0.9.0', '^0.8.0']
	- >=0.7.0 (libraries/external/QueryAccount.sol#3)
	- >=0.8.0<0.9.0 (solidity-bytes-utils/contracts/BytesLib.sol#9)
	- ^0.8.0 (@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2)
	- ^0.8.0 (ChainlinkOracle.sol#2)
	- ^0.8.0 (libraries/Utils.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
QueryAccount.executable(uint256) (libraries/external/QueryAccount.sol#54-57) is never used and should be removed
QueryAccount.lamports(uint256) (libraries/external/QueryAccount.sol#45-48) is never used and should be removed
QueryAccount.owner(uint256) (libraries/external/QueryAccount.sol#27-30) is never used and should be removed
QueryAccount.rent_epoch(uint256) (libraries/external/QueryAccount.sol#63-66) is never used and should be removed
QueryAccount.to_bool(bytes) (libraries/external/QueryAccount.sol#84-88) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2) allows old versions
Pragma version>=0.8.0<0.9.0 (solidity-bytes-utils/contracts/BytesLib.sol#9) is too complex
Pragma version^0.8.0 (ChainlinkOracle.sol#2) allows old versions
Pragma version>=0.7.0 (libraries/external/QueryAccount.sol#3) allows old versions
Pragma version^0.8.0 (libraries/Utils.sol#2) allows old versions
solc-0.8.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in QueryAccount.cache(uint256,uint64,uint64) (libraries/external/QueryAccount.sol#18-21):
        - (success) = precompiled.staticcall(abi.encodeWithSignature(cache(uint256,uint64,uint64),solana_address,offset,len)) (libraries/external/QueryAccount.sol#19)
Low level call in QueryAccount.owner(uint256) (libraries/external/QueryAccount.sol#27-30):
        - (success,result) = precompiled.staticcall(abi.encodeWithSignature(owner(uint256),solana_address)) (libraries/external/QueryAccount.sol#28)
Low level call in QueryAccount.length(uint256) (libraries/external/QueryAccount.sol#36-39):
        - (success,result) = precompiled.staticcall(abi.encodeWithSignature(length(uint256),solana_address)) (libraries/external/QueryAccount.sol#37)
Low level call in QueryAccount.lamports(uint256) (libraries/external/QueryAccount.sol#45-48):
        - (success,result) = precompiled.staticcall(abi.encodeWithSignature(lamports(uint256),solana_address)) (libraries/external/QueryAccount.sol#46)
Low level call in QueryAccount.executable(uint256) (libraries/external/QueryAccount.sol#54-57):
        - (success,result) = precompiled.staticcall(abi.encodeWithSignature(executable(uint256),solana_address)) (libraries/external/QueryAccount.sol#55)
Low level call in QueryAccount.rent_epoch(uint256) (libraries/external/QueryAccount.sol#63-66):
        - (success,result) = precompiled.staticcall(abi.encodeWithSignature(rent_epoch(uint256),solana_address)) (libraries/external/QueryAccount.sol#64)
Low level call in QueryAccount.data(uint256,uint64,uint64) (libraries/external/QueryAccount.sol#74-76):
        - precompiled.staticcall(abi.encodeWithSignature(data(uint256,uint64,uint64),solana_address,offset,len)) (libraries/external/QueryAccount.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BytesLib.concat(bytes,bytes)._preBytes (solidity-bytes-utils/contracts/BytesLib.sol#14) is not in mixedCase
Parameter BytesLib.concat(bytes,bytes)._postBytes (solidity-bytes-utils/contracts/BytesLib.sol#15) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._preBytes (solidity-bytes-utils/contracts/BytesLib.sol#91) is not in mixedCase
Parameter BytesLib.concatStorage(bytes,bytes)._postBytes (solidity-bytes-utils/contracts/BytesLib.sol#91) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#229) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#230) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._length (solidity-bytes-utils/contracts/BytesLib.sol#231) is not in mixedCase
Parameter BytesLib.toAddress(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#297) is not in mixedCaseParameter BytesLib.toAddress(bytes,uint256)._start (solidity-bytes-utils/contracts/Bytes
Lib.sol#297) is not in mixedCaseParameter BytesLib.toUint8(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#308) is not in mixedCase
Parameter BytesLib.toUint8(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#308) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#319) is not in mixedCase
Parameter BytesLib.toUint16(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#319) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#330) is not in mixedCase
Parameter BytesLib.toUint32(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#330) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#341) is not in mixedCase
Parameter BytesLib.toUint64(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#341) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#352) is not in mixedCase
Parameter BytesLib.toUint96(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#352) is not in mixedCase
Parameter BytesLib.toUint128(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#363) is not in mixedCaseParameter BytesLib.toUint128(bytes,uint256)._start (solidity-bytes-utils/contracts/Bytes
Lib.sol#363) is not in mixedCaseParameter BytesLib.toUint256(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#374) is not in mixedCaseParameter BytesLib.toUint256(bytes,uint256)._start (soli
dity-bytes-utils/contracts/BytesLib.sol#374) is not in mixedCaseParameter BytesLib.toBytes32(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#385) is not in mixedCaseParameter BytesLib.toByt
es32(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#385) is not in mixedCaseParameter BytesLib.equal(bytes,bytes)._preBytes (solidity-bytes-utils/contracts/BytesLib.sol#396) is not in mixe
dCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (solidity-bytes-utils/contracts/BytesLib.sol#396) is not in mixedCase
```

```
Parameter BytesLib.toUint128(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#363) is not in mixedCaseParameter BytesLib.toUint128(bytes,uint256)._start (solidity-bytes-utils/contracts/Bytes
Lib.sol#363) is not in mixedCaseParameter BytesLib.toUint256(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#374) is not in mixedCaseParameter BytesLib.toUint256(bytes,uint256)._start (soli
dity-bytes-utils/contracts/BytesLib.sol#374) is not in mixedCaseParameter BytesLib.toBytes32(bytes,uint256)._bytes (solidity-bytes-utils/contracts/BytesLib.sol#385) is not in mixedCaseParameter BytesLib.toByt
es32(bytes,uint256)._start (solidity-bytes-utils/contracts/BytesLib.sol#385) is not in mixedCaseParameter BytesLib.equal(bytes,bytes)._preBytes (solidity-bytes-utils/contracts/BytesLib.sol#396) is not in mixe
dCase
Parameter BytesLib.equal(bytes,bytes)._postBytes (solidity-bytes-utils/contracts/BytesLib.sol#396) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._preBytes (solidity-bytes-utils/contracts/BytesLib.sol#440) is not in mixedCase
Parameter BytesLib.equalStorage(bytes,bytes)._postBytes (solidity-bytes-utils/contracts/BytesLib.sol#441) is not in mixedCase
Parameter ChainlinkOracle.getRoundData(uint80)._roundId (ChainlinkOracle.sol#25) is not in mixedCase
Parameter QueryAccount.cache(uint256,uint64,uint64).solana_address (libraries/external/QueryAccount.sol#18) is not in mixedCase
Parameter QueryAccount.owner(uint256).solana_address (libraries/external/QueryAccount.sol#27) is not in mixedCase
Parameter QueryAccount.length(uint256).solana_address (libraries/external/QueryAccount.sol#36) is not in mixedCase
Parameter QueryAccount.lamports(uint256).solana_address (libraries/external/QueryAccount.sol#45) is not in mixedCase
Parameter QueryAccount.executable(uint256).solana_address (libraries/external/QueryAccount.sol#54) is not in mixedCase
Function QueryAccount.rent_epoch(uint256) (libraries/external/QueryAccount.sol#63-66) is not in mixedCase
Parameter QueryAccount.rent_epoch(uint256).solana_address (libraries/external/QueryAccount.sol#63) is not in mixedCase
Parameter QueryAccount.data(uint256,uint64,uint64).solana_address (libraries/external/QueryAccount.sol#74) is not in mixedCase
Function QueryAccount.to_uint256(bytes) (libraries/external/QueryAccount.sol#78-82) is not in mixedCase
Function QueryAccount.to_bool(bytes) (libraries/external/QueryAccount.sol#84-88) is not in mixedCase
Constant QueryAccount.precompiled (libraries/external/QueryAccount.sol#10) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Utils.getRoundbyId(uint256,uint32,uint32)._roundId (libraries/Utils.sol#96) is not in mixedCase
Parameter Utils.bytesToString(bytes)._bytes (libraries/Utils.sol#200) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BytesLib.toAddress(bytes,uint256) (solidity-bytes-utils/contracts/BytesLib.sol#297-306) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (solidity-bytes-utils/contracts/BytesLib.sol#302)
Utils.readLittleEndianSigned128(uint128) (libraries/Utils.sol#221-226) uses literals with too many digits:
        - input = ((input << 32) & 0xFFFFFFFF00000000FFFFFFFF00000000) | ((input >> 32) & 0x00000000FFFFFFFF00000000FFFFFFFF) (libraries/Utils.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ChainlinkOracle.decimals (ChainlinkOracle.sol#8) should be immutable
ChainlinkOracle.description (ChainlinkOracle.sol#10) should be immutable
ChainlinkOracle.feedAddress (ChainlinkOracle.sol#9) should be immutable
ChainlinkOracle.historicalLength (ChainlinkOracle.sol#12) should be immutable
ChainlinkOracle.version (ChainlinkOracle.sol#11) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
. analyzed (5 contracts with 84 detectors), 113 result(s) found
```

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

Results:

Report for contracts/ChainlinkOracle.sol
https://dashboard.mythx.io/#/console/analyses/f508c56a-bfa4-4d78-b93d-cd819bd63308

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Figure 1: MythX Result - 1

Report for contracts/libraries/Utils.sol
https://dashboard.mythx.io/#/console/analyses/0a1868b8-bd92-4759-b88f-88427d8fd92d

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Figure 2: MythX Result - 2

The findings obtained as a result of the MythX scan were examined, and these findings were included in the report.

THANK YOU FOR CHOOSING

HALBORN