// Smart Contract Security Assessment 12.03.2024 - 12.04.2024

Story Hunt Contracts

Story Hunt

HALBORN

Story Hunt Contracts - Story Hunt

Prepared by: HALBORN

Last Updated 12/17/2024

Date of Engagement by: December 3rd, 2024 - December 4th, 2024

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
5	0	0	0	1	4

TABLE OF CONTENTS

1. Introduction

Story Hunt engaged Halborn to conduct a security assessment on their smart contracts beginning on December 3rd, 2024 and ending on December 4th, 2024. The security assessment was scoped to the smart contracts provided to Halborn. Commit hashes and further details can be found in the Scope section of this report.

The Story Hunt codebase in scope mainly consists of a decentralized exchange based on a fork from UniswapV3 implementation.

2. Assessment Summary

Halborn was provided 2 days for the engagement and assigned 1 full-time security engineer to review the security of the smart contracts in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were mostly acknowledged by the Story Hunt team:

- Implement a two-step ownership transfer process and prevent ownership transfers to the zero address.
- Consider updating the Solidity compiler to the latest version to take advantage of enhanced security and functionality.
- Consider defining constants for magic numbers used.
- Consider providing informative error messages in all functions to inform users and developers why a transaction failed.

3. Test Approach And Methodology

`Halborn` performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture, purpose and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could led to arithmetic related vulnerabilities.
- Local testing with custom scripts (`Foundry`).
- Fork testing against main networks (`Foundry`).
- Static analysis of security for scoped contract, and imported functions (`Slither`).

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: v3-core

(b) Assessed Commit ID: 37f4eb0

(c) Items in scope:

- src/core/interfaces/callback/IStoryHuntV3FlashCallback.sol
- src/core/interfaces/callback/IStoryHuntV3MintCallback.sol
- src/core/interfaces/callback/IStoryHuntV3SwapCallback.sol
- src/core/interfaces/pool/IStoryHuntV3PoolActions.sol
- src/core/interfaces/pool/IStoryHuntV3PoolDerivedState.sol
- src/core/interfaces/pool/IStoryHuntV3PoolEvents.sol
- src/core/interfaces/pool/IStoryHuntV3PoolImmutables.sol
- src/core/interfaces/pool/IStoryHuntV3PoolOwnerActions.sol
- src/core/interfaces/pool/IStoryHuntV3PoolState.sol
- src/core/interfaces/IStoryHuntV3Factory.sol
- src/core/interfaces/IStoryHuntV3Pool.sol
- src/core/interfaces/IStoryHuntV3PoolDeployer.sol
- src/core/StoryHuntV3Factory.sol
- src/core/StoryHuntV3Pool.sol
- src/core/StoryHuntV3PoolDeployer.sol
- src/periphery/lens/StoryHuntInterfaceMulticall.sol
- src/periphery/libraries/PoolAddress.sol
- 0xstoryhunt/v3-periphery/tree/ff15b017440c9fd1f598804cd93b95c365145525

Out-of-Scope:

REMEDIATION COMMIT ID:

- 2ce68d1

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INSECURE OWNERSHIP TRANSFER MECHANISM	LOW	SOLVED - 12/14/2024
OUTDATED COMPILER VERSIONS	INFORMATIONAL	ACKNOWLEDGED - 12/14/2024
USE OF MAGIC NUMBERS	INFORMATIONAL	ACKNOWLEDGED - 12/14/2024
MISSING ERROR DESCRIPTIONS	INFORMATIONAL	ACKNOWLEDGED - 12/14/2024
UNOPTIMIZED FOR LOOP	INFORMATIONAL	ACKNOWLEDGED - 12/14/2024

7. FINDINGS & TECH DETAILS

7.1 INSECURE OWNERSHIP TRANSFER MECHANISM

// LOW

Description

The `setOwner()` function in the `StoryHuntV3Factory` contract implements an unsafe ownership transfer process that exposes the contract to potential security risks. The current implementation allows a single-step ownership transfer with the following shortcomings:

- No validation to prevent ownership transfers to the zero address.
- No confirmation required from the new owner.
- Immediate and irreversible ownership transfer.

This vulnerability could enable permanent loss of contract control through unintended transfers.

Code Location

```
54  function setOwner(address _owner) external override {
55      require(msg.sender == owner);
56      emit OwnerChanged(owner, _owner);
57      owner = _owner;
58  }
```

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (2.5)

Recommendation

Implement a two-step ownership transfer process, to add an extra layer of security to prevent accidental ownership transfers and prevent ownership transfers to the zero address.

Remediation

SOLVED: The **Story Hunt team** solved this finding in commit `2ce68d1` by following the mentioned recommendation.

Remediation Hash

https://github.com/0xstoryhunt/v3-core/commit/2ce68d1576957017f3e4521350fb224907dd1ffb

References

0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L54-L58

7.2 OUTDATED COMPILER VERSIONS

// INFORMATIONAL

Description

The Solidity compiler versions specified for the files in scope are `>=0.5.0` and `=0.7.6`, both of which are significantly outdated.

As of now, the latest Solidity version is `0.8.28`. Using outdated compiler versions can expose contracts to security vulnerabilities and prevent access to important language enhancements and optimizations introduced in more recent updates.

For a detailed overview of known bugs in various Solidity versions, refer to the official Solidity documentation. Additionally, a user-friendly visualization is available here.

BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:L/D:N/Y:N (0.5)

Recommendation

Consider updating the Solidity compiler to the latest version to take advantage of enhanced security and functionality.

Remediation

ACKNOWLEDGED: The **Story Hunt team** made a business decision to acknowledge this finding and not alter the contracts.

References

0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L2
0xstoryhunt/v3-periphery/contracts/lens/StoryHuntInterfaceMulticall.sol#L2
0xstoryhunt/v3-periphery/contracts/libraries/PoolAddress.sol#L2
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L2
0xstoryhunt/v3-core/contracts/StoryHuntV3PoolDeployer.sol#L2

7.3 USE OF MAGIC NUMBERS

// INFORMATIONAL

Description

Throughout the files in scope, particularly in `StoryHuntV3Factory` and `StoryHuntV3Pool` contracts, there are instances where magic numbers are used.

Magic numbers are direct numerical or string values used in the code without any explanation or context. This practice can lead to code maintainability issues, potential errors, and difficulty in understanding the code's logic.

Score

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

To improve the code's readability and facilitate refactoring, consider defining constants for magic numbers used, giving it clear and self-explanatory names.

Remediation

ACKNOWLEDGED: The **Story Hunt team** made a business decision to acknowledge this finding and not alter the contracts.

References

0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L63
0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L67
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L144
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L155
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L617
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L814
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L820
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L830-L838

7.4 MISSING ERROR DESCRIPTIONS

// INFORMATIONAL

Description

Throughout the files in scope, there are several instances where functions can revert without providing an error message. This can make it difficult for users and developers to understand why a transaction failed and how to resolve the issue.

Score

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Consider providing informative error messages in all functions to inform users and developers why a transaction failed.

Remediation

ACKNOWLEDGED: The **Story Hunt team** made a business decision to acknowledge this finding and not alter the contracts.

References

0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L40-L45
0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L55
0xstoryhunt/v3-core/contracts/StoryHuntV3Factory.sol#L62-L68
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L113
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L144
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L155
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L189
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L198
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L463
0xstoryhunt/v3-core/contracts/StoryHuntV3Pool.sol#L831-L834
0xstoryhunt/v3-periphery/contracts/libraries/PoolAddress.sol#L30

7.5 UNOPTIMIZED FOR LOOP

// INFORMATIONAL

Description

In the `StoryHuntInterfaceMulticall` contract, there is an instance of an unoptimized for loop declaration that may incur in higher gas costs than necessary.

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Optimize the `for` loop declarations to reduce gas costs. Best practices include:

- The non-redundant initialization of the iterator with a default value (declaring simply `i` is equivalent to `i = 0` but more gas efficient),
- The use of the pre-increment operator (inside an `unchecked` block if using Solidity `>=0.8.0` and `<= 0.8.21` :`unchecked {++i}`, or simply `++i` if compiling with Solidity `>=0.8.22`).

Additionally, when reading from storage variables, it is recommended to reduce gas costs significantly by caching the array to read locally and iterate over it to avoid reading from storage on every iteration. Moreover, if there are several loops in the same function, the `i` variable can be re-used, to be able to set the value from non-zero to zero and reduce gas costs without additional variable declaration. For example:

```
uint256[] memory arrayInMemory = arrayInStorage;

uint256 i;
for (; i < arrayInMemory.length ;) {
  // code logic
  unchecked { ++i; }
}

delete i;

uint256[] memory arrayInMemory2 = arrayInStorage2;

for (; i < arrayInMemory2.length ;) {
  // code logic
```

```
    unchecked { ++i; }
  }
```

Remediation

ACKNOWLEDGED: The **Story Hunt team** made a business decision to acknowledge this finding and not alter the contracts.

References

0xstoryhunt/v3-periphery/contracts/lens/StoryHuntInterfaceMulticall.sol#L30

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.
The security team assessed all findings identified by the Slither software, however, findings with related to external dependencies are not included in the below results for the sake of report readability.

Output

The findings obtained as a result of the Slither scan were reviewed, and the majority were not included in the report because they were determined as false positives.

```
StoryHuntV3Pool.initialize(uint160) (src/core/StoryHuntV3Pool.sol#241-259) uses a dangerous strict equality:
        - require(bool,string)(slot0.sqrtPriceX96 == 0,AI) (src/core/StoryHuntV3Pool.sol#242)
StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710) uses a dangerous strict equality:
        - state.sqrtPriceX96 == step.sqrtPriceNextX96 (src/core/StoryHuntV3Pool.sol#617)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in StoryHuntV3Pool.collectProtocol(address,uint128,uint128) (src/core/StoryHuntV3Pool.sol#762-782):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (src/core/StoryHuntV3Pool.sol#773)
        State variables written after the call(s):
        - protocolFees.token1 -= amount1 (src/core/StoryHuntV3Pool.sol#777)
        StoryHuntV3Pool.protocolFees (src/core/StoryHuntV3Pool.sol#87) can be used in cross function reentrancies:
        - StoryHuntV3Pool.collectProtocol(address,uint128,uint128) (src/core/StoryHuntV3Pool.sol#762-782)
        - StoryHuntV3Pool.flash(address,uint256,uint256,bytes) (src/core/StoryHuntV3Pool.sol#713-751)
        - StoryHuntV3Pool.protocolFees (src/core/StoryHuntV3Pool.sol#87)
        - StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710)
Reentrancy in StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710):
        External calls:
        - TransferHelper.safeTransfer(token1,recipient,uint256(-amount1)) (src/core/StoryHuntV3Pool.sol#695)
        - IStoryHuntV3SwapCallback(msg.sender).storyHuntV3SwapCallback(amount0,amount1,data) (src/core/StoryHuntV3Pool.sol#698)
        - TransferHelper.safeTransfer(token0,recipient,uint256(-amount0)) (src/core/StoryHuntV3Pool.sol#701)
        - IStoryHuntV3SwapCallback(msg.sender).storyHuntV3SwapCallback(amount0,amount1,data) (src/core/StoryHuntV3Pool.sol#704)
        State variables written after the call(s):
        - slot0.unlocked = true (src/core/StoryHuntV3Pool.sol#709)
        StoryHuntV3Pool.slot0 (src/core/StoryHuntV3Pool.sol#74) can be used in cross function reentrancies:
        - StoryHuntV3Pool._modifyPosition(StoryHuntV3Pool.ModifyPositionParams) (src/core/StoryHuntV3Pool.sol#276-320)
        - StoryHuntV3Pool._updatePosition(address,int24,int24,int128,int24) (src/core/StoryHuntV3Pool.sol#327-405)
        - StoryHuntV3Pool.flash(address,uint256,uint256,bytes) (src/core/StoryHuntV3Pool.sol#713-751)
        - StoryHuntV3Pool.increaseObservationCardinalityNext(uint16) (src/core/StoryHuntV3Pool.sol#231-237)
        - StoryHuntV3Pool.initialize(uint160) (src/core/StoryHuntV3Pool.sol#241-259)
        - StoryHuntV3Pool.lock() (src/core/StoryHuntV3Pool.sol#104-109)
        - StoryHuntV3Pool.observe(uint32[]) (src/core/StoryHuntV3Pool.sol#224-228)
        - StoryHuntV3Pool.setFeeProtocol(uint8,uint8) (src/core/StoryHuntV3Pool.sol#754-759)
        - StoryHuntV3Pool.slot0 (src/core/StoryHuntV3Pool.sol#74)
        - StoryHuntV3Pool.snapshotCumulativesInside(int24,int24) (src/core/StoryHuntV3Pool.sol#156-221)
        - StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StoryHuntV3Pool.mint(address,int24,int24,uint128,bytes).balance0Before (src/core/StoryHuntV3Pool.sol#424) is a local variable never initialized
StoryHuntV3Pool._updatePosition(address,int24,int24,int128,int24).flippedUpper (src/core/StoryHuntV3Pool.sol#341) is a local variable never initialized
StoryHuntV3Pool.mint(address,int24,int24,uint128,bytes).balance1Before (src/core/StoryHuntV3Pool.sol#425) is a local variable never initialized
StoryHuntV3Pool._updatePosition(address,int24,int24,int128,int24).flippedLower (src/core/StoryHuntV3Pool.sol#340) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
StoryHuntV3Pool.observe(uint32[]) (src/core/StoryHuntV3Pool.sol#224-228) ignores return value by observations.observe(_blockTimestamp(),secondsAgos,slot0.tick,slot0.observationIndex,liquidity,slot0.observationCardinality) (src/core/StoryHuntV3Pool.sol#227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
StoryHuntV3Factory.setOwner(address)._owner (src/core/StoryHuntV3Factory.sol#50) lacks a zero-check on :
                - owner = _owner (src/core/StoryHuntV3Factory.sol#53)
StoryHuntInterfaceMulticall.multicall(StoryHuntInterfaceMulticall.Call[]).target (src/periphery/lens/StoryHuntInterfaceMulticall.sol#31) lacks a zero-check on :
                - (success,ret) = target.call{gas: gasLimit}(callData) (src/periphery/lens/StoryHuntInterfaceMulticall.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StoryHuntV3Pool.flash(address,uint256,uint256,bytes) (src/core/StoryHuntV3Pool.sol#713-751):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (src/core/StoryHuntV3Pool.sol#722)
        - TransferHelper.safeTransfer(token1,recipient,amount1) (src/core/StoryHuntV3Pool.sol#723)
        - IStoryHuntV3FlashCallback(msg.sender).storyHuntV3FlashCallback(fee0,fee1,data) (src/core/StoryHuntV3Pool.sol#725)
        State variables written after the call(s):
        - feeGrowthGlobal0X128 += FullMath.mulDiv(paid0 - fees0,FixedPoint128.Q128,_liquidity) (src/core/StoryHuntV3Pool.sol#741)
        - feeGrowthGlobal1X128 += FullMath.mulDiv(paid1 - fees1,FixedPoint128.Q128,_liquidity) (src/core/StoryHuntV3Pool.sol#747)
```

```
        - protocolFees.token0 += uint128(fees0) (src/core/StoryHuntV3Pool.sol#740)
        - protocolFees.token1 += uint128(fees1) (src/core/StoryHuntV3Pool.sol#746)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StoryHuntV3Pool.collect(address,int24,int24,uint128,uint128) (src/core/StoryHuntV3Pool.sol#436-459):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (src/core/StoryHuntV3Pool.sol#451)
        - TransferHelper.safeTransfer(token1,recipient,amount1) (src/core/StoryHuntV3Pool.sol#455)
        Event emitted after the call(s):
        - Collect(msg.sender,recipient,tickLower,tickUpper,amount0,amount1) (src/core/StoryHuntV3Pool.sol#458)
Reentrancy in StoryHuntV3Pool.collectProtocol(address,uint128,uint128) (src/core/StoryHuntV3Pool.sol#762-782):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (src/core/StoryHuntV3Pool.sol#773)
        - TransferHelper.safeTransfer(token1,recipient,amount1) (src/core/StoryHuntV3Pool.sol#778)
        Event emitted after the call(s):
        - CollectProtocol(msg.sender,recipient,amount0,amount1) (src/core/StoryHuntV3Pool.sol#781)
Reentrancy in StoryHuntV3Pool.flash(address,uint256,uint256,bytes) (src/core/StoryHuntV3Pool.sol#713-751):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (src/core/StoryHuntV3Pool.sol#722)
        - TransferHelper.safeTransfer(token1,recipient,amount1) (src/core/StoryHuntV3Pool.sol#723)
        - IStoryHuntV3FlashCallback(msg.sender).storyHuntV3FlashCallback(fee0,fee1,data) (src/core/StoryHuntV3Pool.sol#725)
        Event emitted after the call(s):
        - Flash(msg.sender,recipient,amount0,amount1,paid0,paid1) (src/core/StoryHuntV3Pool.sol#750)
Reentrancy in StoryHuntV3Pool.mint(address,int24,int24,uint128,bytes) (src/core/StoryHuntV3Pool.sol#409-433):
        External calls:
        - IStoryHuntV3MintCallback(msg.sender).storyHuntV3MintCallback(amount0,amount1,data) (src/core/StoryHuntV3Pool.sol#428)
        Event emitted after the call(s):
        - Mint(msg.sender,recipient,tickLower,tickUpper,amount,amount0,amount1) (src/core/StoryHuntV3Pool.sol#432)
Reentrancy in StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710):
        External calls:
        - TransferHelper.safeTransfer(token1,recipient,uint256(-amount1)) (src/core/StoryHuntV3Pool.sol#695)
        - IStoryHuntV3SwapCallback(msg.sender).storyHuntV3SwapCallback(amount0,amount1,data) (src/core/StoryHuntV3Pool.sol#698)
        - TransferHelper.safeTransfer(token0,recipient,uint256(-amount0)) (src/core/StoryHuntV3Pool.sol#701)
        - IStoryHuntV3SwapCallback(msg.sender).storyHuntV3SwapCallback(amount0,amount1,data) (src/core/StoryHuntV3Pool.sol#704)
        Event emitted after the call(s):
        - Swap(msg.sender,recipient,amount0,amount1,state.sqrtPriceX96,state.liquidity,state.tick) (src/core/StoryHuntV3Pool.sol#708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StoryHuntInterfaceMulticall.multicall(StoryHuntInterfaceMulticall.Call[]) (src/periphery/lens/StoryHuntInterfaceMulticall.sol#27-41) tries to limit the gas of an external call that controls implicit decoding
        (success,ret) = target.call{gas: gasLimit}(callData) (src/periphery/lens/StoryHuntInterfaceMulticall.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#return-bomb
INFO:Detectors:
StoryHuntV3Pool.initialize(uint160) (src/core/StoryHuntV3Pool.sol#241-259) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(slot0.sqrtPriceX96 == 0,AI) (src/core/StoryHuntV3Pool.sol#242)
StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710) uses timestamp for comparisons
        Dangerous comparisons:
        - state.amountSpecifiedRemaining ≠ 0 && state.sqrtPriceX96 ≠ sqrtPriceLimitX96 (src/core/StoryHuntV3Pool.sol#572)
        - step.tickNext < TickMath.MIN_TICK (src/core/StoryHuntV3Pool.sol#580)
        - step.tickNext > TickMath.MAX_TICK (src/core/StoryHuntV3Pool.sol#582)
        - cache.feeProtocol > 0 (src/core/StoryHuntV3Pool.sol#607)
        - state.liquidity > 0 (src/core/StoryHuntV3Pool.sol#614)
        - state.sqrtPriceX96 == step.sqrtPriceNextX96 (src/core/StoryHuntV3Pool.sol#617)
        - state.sqrtPriceX96 ≠ step.sqrtPriceStartX96 (src/core/StoryHuntV3Pool.sol#649)
        - state.tick ≠ slot0Start.tick (src/core/StoryHuntV3Pool.sol#656)
        - cache.liquidityStart ≠ state.liquidity (src/core/StoryHuntV3Pool.sol#677)
        - state.protocolFee > 0 (src/core/StoryHuntV3Pool.sol#683)
        - state.protocolFee > 0 (src/core/StoryHuntV3Pool.sol#686)
        - amount1 < 0 (src/core/StoryHuntV3Pool.sol#695)
        - require(bool,string)(balance0Before.add(uint256(amount0)) ≤ balance0(),IIA) (src/core/StoryHuntV3Pool.sol#699)
        - amount0 < 0 (src/core/StoryHuntV3Pool.sol#701)
        - require(bool,string)(balance1Before.add(uint256(amount1)) ≤ balance1(),IIA) (src/core/StoryHuntV3Pool.sol#705)
```

```
        - (step.sqrtPriceNextX96 < sqrtPriceLimitX96) (src/core/StoryHuntV3Pool.sol#590-596)
        - (step.sqrtPriceNextX96 > sqrtPriceLimitX96) (src/core/StoryHuntV3Pool.sol#590-596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
StoryHuntV3Pool.swap(address,bool,int256,uint160,bytes) (src/core/StoryHuntV3Pool.sol#529-710) has a high cyclomatic complexity (28).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Version constraint ≥0.5.0 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - DirtyBytesArrayToStorage
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching
        - EmptyByteArrayCopy
        - DynamicArrayCleanup
        - ImplicitConstructorCallvalueCheck
        - TupleAssignmentMultiStackSlotComponents
        - MemoryArrayCreationOverflow
        - privateCanBeOverridden
        - SignedArrayStorageCopy
        - ABIEncoderV2StorageArrayWithMultiSlotElement
        - DynamicConstructorArgumentsClippedABIV2
        - UninitializedFunctionPointerInConstructor
        - IncorrectEventSignatureInLibraries
        - ABIEncoderV2PackedStorage.
It is used by:
        - ≥0.5.0 (lib/solidity-lib/contracts/libraries/AddressStringUtil.sol#3)
        - ≥0.5.0 (lib/solidity-lib/contracts/libraries/SafeERC20Namer.sol#3)
        - ≥0.5.0 (src/core/interfaces/IERC20Minimal.sol#2)
        - ≥0.5.0 (src/core/interfaces/IStoryHuntV3Factory.sol#2)
        - ≥0.5.0 (src/core/interfaces/IStoryHuntV3Pool.sol#2)
        - ≥0.5.0 (src/core/interfaces/IStoryHuntV3PoolDeployer.sol#2)
        - ≥0.5.0 (src/core/interfaces/callback/IStoryHuntV3FlashCallback.sol#2)
        - ≥0.5.0 (src/core/interfaces/callback/IStoryHuntV3MintCallback.sol#2)
        - ≥0.5.0 (src/core/interfaces/callback/IStoryHuntV3SwapCallback.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolActions.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolDerivedState.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolEvents.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolImmutables.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolOwnerActions.sol#2)
        - ≥0.5.0 (src/core/interfaces/pool/IStoryHuntV3PoolState.sol#2)
        - ≥0.5.0 (src/core/libraries/BitMath.sol#2)
        - ≥0.5.0 (src/core/libraries/LiquidityMath.sol#2)
        - ≥0.5.0 (src/core/libraries/SafeCast.sol#2)
        - ≥0.5.0 (src/core/libraries/SqrtPriceMath.sol#2)
        - ≥0.5.0 (src/core/libraries/SwapMath.sol#2)
        - ≥0.5.0 (src/core/libraries/TickBitmap.sol#2)
        - ≥0.5.0 (src/core/libraries/UnsafeMath.sol#2)
        - ≥0.5.0 (src/periphery/base/SelfPermit.sol#2)
        - ≥0.5.0 (src/periphery/interfaces/INonfungibleTokenPositionDescriptor.sol#2)
        - ≥0.5.0 (src/periphery/interfaces/IPeripheryImmutableState.sol#2)
        - ≥0.5.0 (src/periphery/interfaces/external/IERC1271.sol#2)
        - ≥0.5.0 (src/periphery/interfaces/external/IERC20PermitAllowed.sol#2)
        - ≥0.5.0 (src/periphery/lens/TickLens.sol#2)
        - ≥0.5.0 (src/periphery/libraries/LiquidityAmounts.sol#2)
        - ≥0.5.0 (src/periphery/libraries/PoolAddress.sol#2)
        - ≥0.5.0 (src/periphery/libraries/PositionKey.sol#2)
        - ≥0.5.0 (src/periphery/libraries/SqrtPriceMathPartial.sol#2)
Version constraint =0.7.6 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
```

```
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching.
It is used by:
        - =0.7.6 (src/core/NoDelegateCall.sol#2)
        - =0.7.6 (src/core/StoryHuntV3Factory.sol#2)
        - =0.7.6 (src/core/StoryHuntV3Pool.sol#2)
        - =0.7.6 (src/core/StoryHuntV3PoolDeployer.sol#2)
        - =0.7.6 (src/periphery/NonfungiblePositionManager.sol#2)
        - =0.7.6 (src/periphery/NonfungibleTokenPositionDescriptor.sol#2)
        - =0.7.6 (src/periphery/SwapRouter.sol#2)
        - =0.7.6 (src/periphery/base/BlockTimestamp.sol#2)
        - =0.7.6 (src/periphery/base/ERC721Permit.sol#2)
        - =0.7.6 (src/periphery/base/LiquidityManagement.sol#2)
        - =0.7.6 (src/periphery/base/Multicall.sol#2)
        - =0.7.6 (src/periphery/base/PeripheryImmutableState.sol#2)
        - =0.7.6 (src/periphery/base/PeripheryValidation.sol#2)
        - =0.7.6 (src/periphery/base/PoolInitializer.sol#2)
        - =0.7.6 (src/periphery/examples/PairFlash.sol#2)
        - =0.7.6 (src/periphery/interfaces/external/IWETH9.sol#2)
        - =0.7.6 (src/periphery/lens/Quoter.sol#2)
        - =0.7.6 (src/periphery/lens/QuoterV2.sol#2)
        - =0.7.6 (src/periphery/lens/StoryHuntInterfaceMulticall.sol#2)
        - =0.7.6 (src/periphery/libraries/CallbackValidation.sol#2)
        - =0.7.6 (src/periphery/libraries/HexStrings.sol#2)
        - =0.7.6 (src/periphery/libraries/TokenRatioSortOrder.sol#2)
solc-0.7.6 is an outdated solc version. Use a more recent version (at least 0.8.0), if possible.
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
The following unused import(s) in src/periphery/libraries/NFTDescriptor.sol should be removed:
        -import "src/core/interfaces/IStoryHuntV3Pool.sol"; (src/periphery/libraries/NFTDescriptor.sol#13)

        -import "src/core/libraries/BitMath.sol"; (src/periphery/libraries/NFTDescriptor.sol#10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-imports
INFO:Slither:. analyzed (103 contracts with 94 detectors), 25 result(s) found
```

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.