HighStreetMarket - NFT Pool

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 21st, 2021 - December 22nd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/21/2021	Roberto Reigada
0.2	Document Updates	12/21/2021	Roberto Reigada
0.3	Draft Review	12/23/2021	Gabi Urrutia
1.0	Remediation Plan	12/27/2021	Roberto Reigada
1.1	Remediation Plan Review	12/27/2021	Gabi Urrutia

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
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

HighStreetMarket engaged Halborn to conduct a security audit on their HighStreetNftPool.sol smart contract beginning on December 21st, 2021 and ending on December 22nd, 2021. The security assessment was scoped to the smart contract provided in the Github repository Highstreet-World/StakingPool

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned one full-time security engineer to audit the security of the smart contract. The security engineer a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit 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 addressed by HighStreetMarket team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding 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 bridge code 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 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 hotspots 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

IN-SCOPE:

The security assessment was scoped to the following smart contract:

- HighStreetNftPool.sol

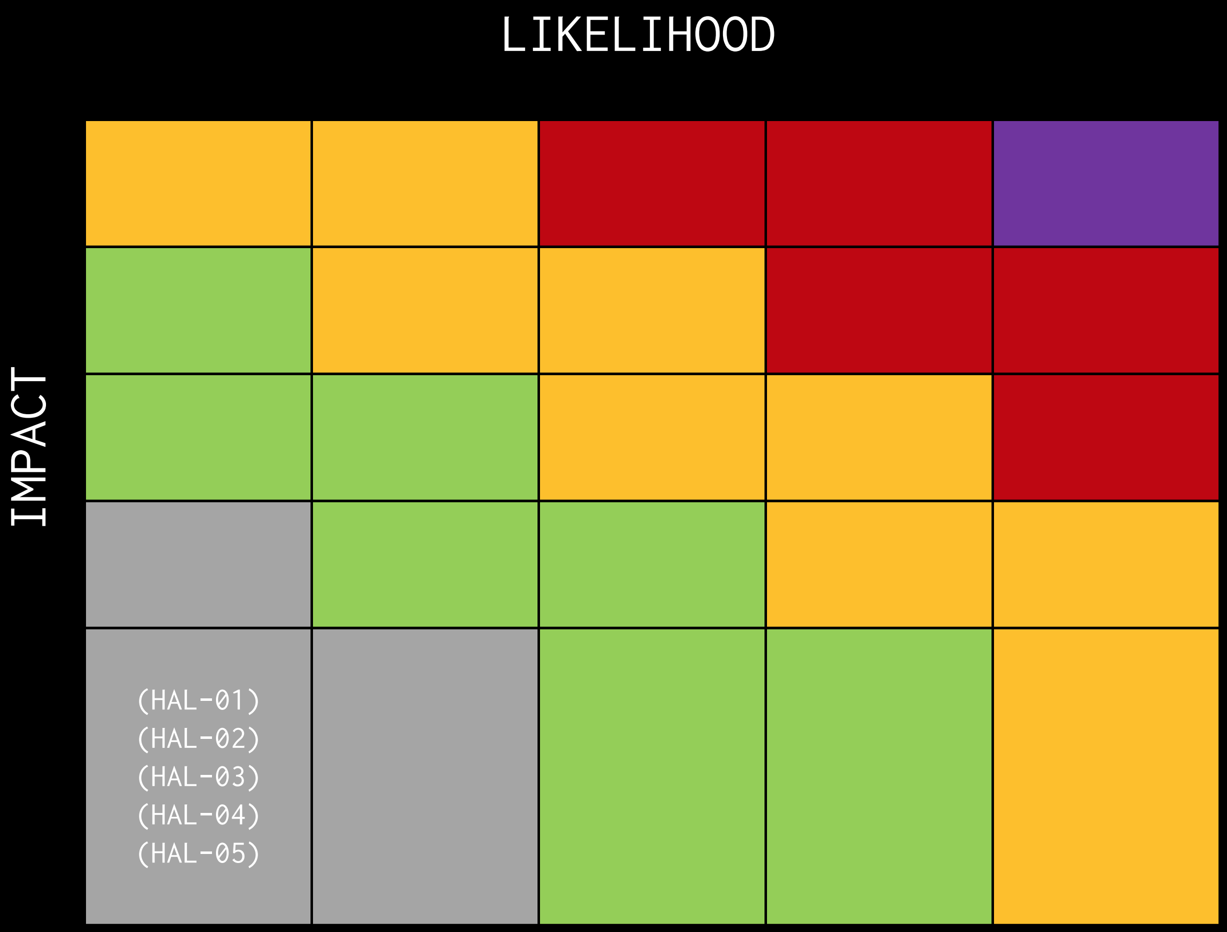
Commit ID: 8f88e62d1c88711f83c233cffefd5d66d1cb1589

OUT-OF-SCOPE:

Other smart contracts in the repository, external libraries and economical attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	5



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - AMOUNT PARAMETER CAN BE REMOVED	Informational	SOLVED - 12/23/2021
HAL-02 - EMERGENCYWITHDRAW FUNCTION DOES NOT PROVIDE ANY UTILITY	Informational	SOLVED - 12/23/2021
HAL-03 - SOME FUNCTIONS CAN BE REMOVED	Informational	SOLVED - 12/23/2021
HAL-04 - GAS OVER-CONSUMPTION IN LOOPS	Informational	SOLVED - 12/23/2021
HAL-05 - UNNECESSARY INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	SOLVED - 12/23/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) AMOUNT PARAMETER CAN BE REMOVED - INFORMATIONAL

Description:

The contract HighStreetNftPool contains the function unstakeReward(uint256 _depositId, uint256 _amount) which allows a user to unstake a specified amount of rewards. The _amount parameter here does not provide any utility, as there is no incentive for the user to just claim a part of his total rewards.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the _amount parameter from the unstakeReward() and _unstakeReward() functions. The full reward amount should be sent to the user instead. This will simplify the logic of the functions, saving some gas.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 079f3c357f653d57452059a13d66ec994c80753a

3.2 (HAL-02) EMERGENCYWITHDRAW FUNCTION DOES NOT PROVIDE ANY UTILITY - INFORMATIONAL

Description:

The contract HighStreetNftPool contains the functions emergencyWithdraw() and _emergencyWithdraw():

Listing 1: emergencyWithdraw - external (Lines 441)

```
439 function emergencyWithdraw(uint256[] calldata _listIds) external
        nonReentrant {
440     // delegate call to an internal function
441     _emergencyWithdraw(msg.sender, _listIds);
442 }
```

Listing 2: emergencyWithdraw - internal

```
function _emergencyWithdraw(
    address _staker,
    uint256[] calldata _listIds
) internal virtual {
    require(_listIds.length > 0, "zero amount");
    // limit the max nft transfer.
    require(_listIds.length <= 40, "length exceeds limitation");

    // get a link to user data struct, we will write to it later
    User storage user = users[_staker];
    uint16[] memory list = user.list;
    uint256 amount = _listIds.length;
    require(user.tokenAmount >= amount, "amount exceeds stake");

    // update smart contract state
    _sync();

    // update user record
    user.tokenAmount -= amount;
    user.subYieldRewards = tokenToReward(user.tokenAmount,
        yieldRewardsPerToken);
    usersLockingAmount = usersLockingAmount - amount;

    uint256 index;
    uint256[] memory nfts = new uint256[](_listIds.length);
    for(uint i =0; i < _listIds.length; i++) {
        index = _listIds[i];
        if(UINT_16_MAX == list[index]) {
            nfts[i] = 0;
        } else {
            nfts[i] = uint256(list[index]);
        }
        IERC721(poolToken).safeTransferFrom(address(this), _staker, nfts[i]);
        if (user.tokenAmount  != 0) {
            delete user.list[index];
        }
    }

    if (user.tokenAmount  == 0) {
        delete user.list;
    }

    emit EmergencyWithdraw(msg.sender, amount, nfts);
}
```

The function _emergencyWithdraw() contains the same code as the _unstake() function, and it is not providing any extra functionality. Furthermore, there is no locking applied to the staked NFTs which means that the user has no restrictions to retrieve them. There is no need for any kind of emergencyWithdraw function in the smart contract.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the emergencyWithdraw() and _emergencyWithdraw() functions.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID a9e1c25ec2de219e1f66c31b51d7b7a7997b15b3

3.3 (HAL-03) SOME FUNCTIONS CAN BE REMOVED - INFORMATIONAL

Description:

The contract HighStreetNftPool contains the function mintYieldTo():

Listing 3: mintYieldTo

```
function mintYieldTo(address _to, uint256 _amount) internal {
    // transfer HIGH tokens as required
    transferHighToken(_to, _amount);

    emit MintYield(_to, _amount);
}
```

This mintYieldTo() function, all it does is calling another internal function: transferHighToken():

Listing 4: transferHighToken

```
function transferHighToken(address _to, uint256 _value) internal {
    // just delegate call to the target
    SafeERC20.safeTransfer(IERC20(HIGH), _to, _value);
}
```

At the same time, the transferHighToken() function just calls the SafeERC20.safeTransfer(IERC20(HIGH), _to, _value); function.

mintYieldTo() and transferHighToken() can be removed and instead SafeERC20.safeTransfer can be used directly to save some gas.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the mintYieldTo() and transferHighToken() functions and instead use directly SafeERC20.safeTransfer.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 8c6269aff32496a7e75630a63abbf80f5015d18a

3.4 (HAL-04) GAS OVER-CONSUMPTION IN LOOPS - INFORMATIONAL

Description:

In all the loops, the counter variable is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++.

Code Location:

HighStreetNftPool.sol

- Line 307: for(uint256 i = pageStart; i < pageEnd; i++){
- Line 350: for(uint256 i = pageStart; i < pageEnd; i++){
- Line 520: for(uint i =0; i < _nftIds.length; i++){
- Line 571: for(uint i =0; i < _listIds.length; i++){
- Line 663: for(uint i =0; i < _listIds.length; i++){

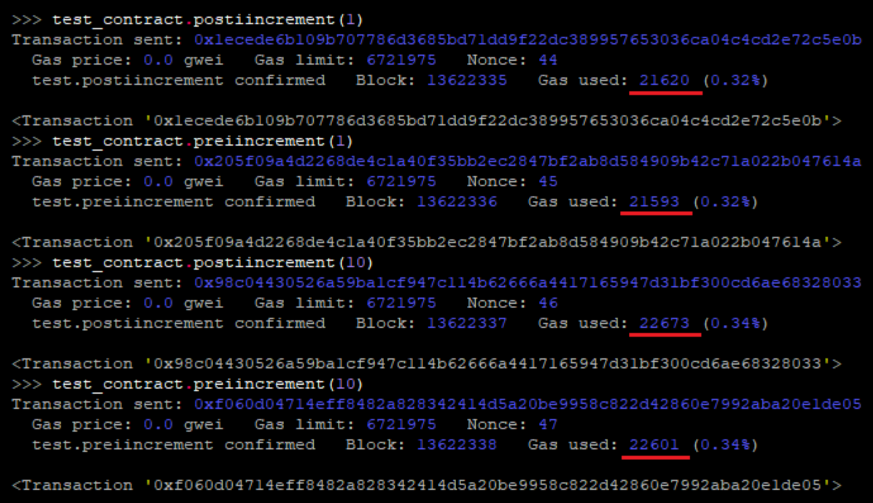
Proof of Concept:

For example, based in the following test contract:

Listing 5: Test.sol

```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.9;

contract test {
    function postiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; i++) {
        }
    }
    function preiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; ++i) {
        }
    }
}
```

We can see the difference in the gas costs:



```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use ++i instead of i++ to increment the value of an uint variable inside a loop. This is not applicable outside of loops.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID a4017a4a5eeedb24d3792744cf71c5d0dcf5394c

3.5 (HAL-05) UNNECESSARY INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

As i is an uint, it is already initialized to 0. uint i = 0 reassigns the 0 to i which wastes gas.

Code Location:

HighStreetNftPool.sol
- Line 519: uint256 addedAmount = 0;
- Line 520: for(uint i =0; i < _nftIds.length; i++){
- Line 571: for(uint i =0; i < _listIds.length; i++){
- Line 663: for(uint i =0; i < _listIds.length; i++){

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize i variable to 0 to save some gas. For example:

```
for(uint i; i < _nftIds.length; i++){
```

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 8f715b12b6fcc4ce25f27fc2357ab268a1c35472

AUTOMATED TESTING

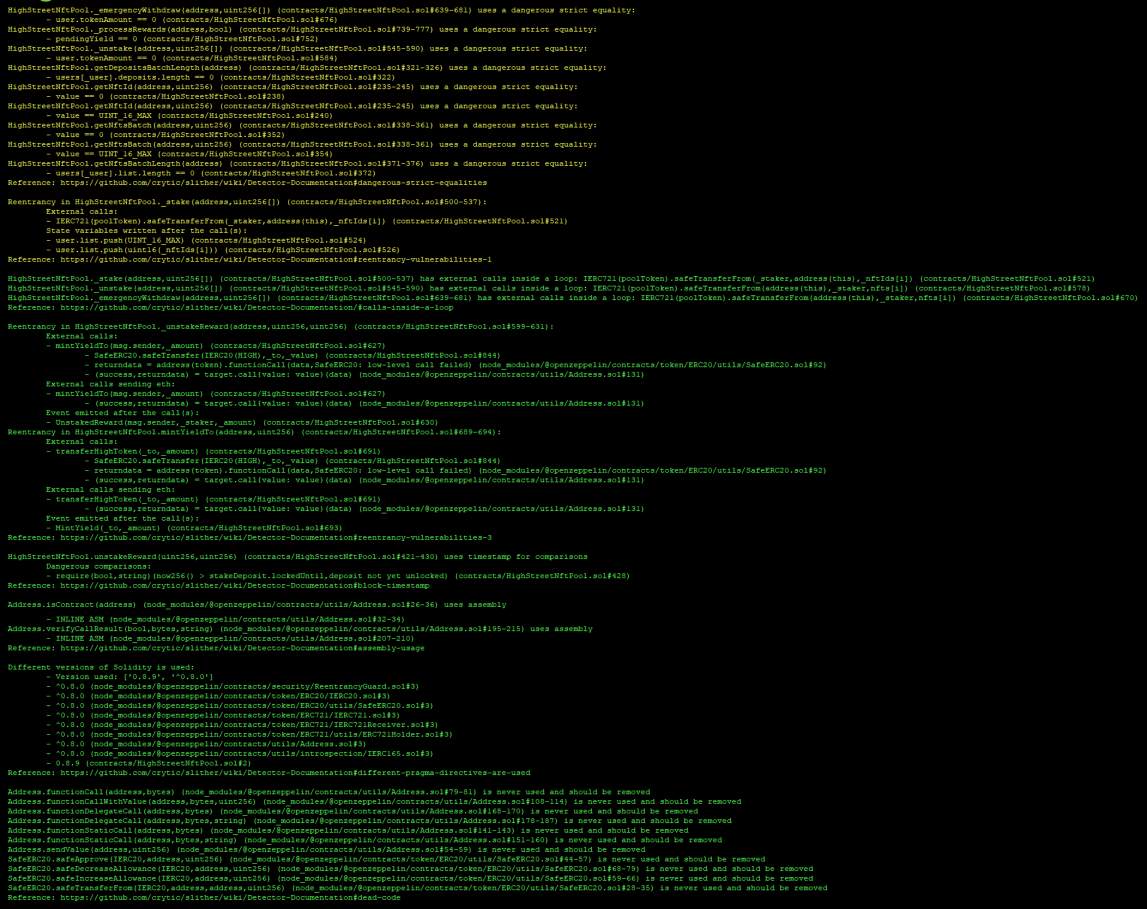
4.1 STATIC ANALYSIS REPORT

Description:

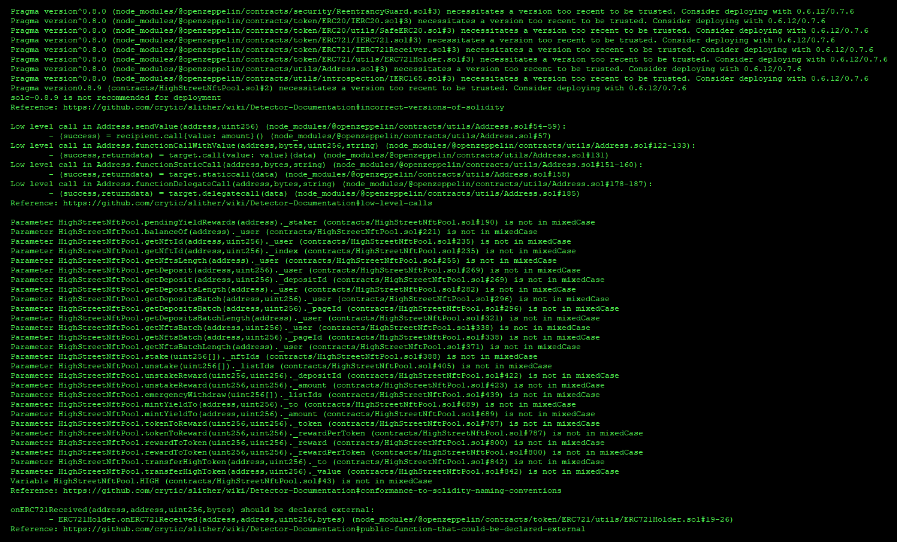
Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats, Slither was run on the all-scoped 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.

Slither results:

HighStreetNftPool.sol



```
HighStreetNftPool._emergencyWithdraw(address,uint256[]) (contracts/HighStreetNftPool.sol#639-681) uses a dangerous strict equality:
        - user.tokenAmount == 0 (contracts/HighStreetNftPool.sol#676)
HighStreetNftPool._processRewards(address,bool) (contracts/HighStreetNftPool.sol#739-777) uses a dangerous strict equality:
        - pendingYield == 0 (contracts/HighStreetNftPool.sol#752)
HighStreetNftPool._unstake(address,uint256[]) (contracts/HighStreetNftPool.sol#545-590) uses a dangerous strict equality:
        - user.tokenAmount == 0 (contracts/HighStreetNftPool.sol#584)
HighStreetNftPool.getDepositsBatchLength(address) (contracts/HighStreetNftPool.sol#321-326) uses a dangerous strict equality:
        - users[_user].deposits.length == 0 (contracts/HighStreetNftPool.sol#322)
HighStreetNftPool.getNftId(address,uint256) (contracts/HighStreetNftPool.sol#235-245) uses a dangerous strict equality:
        - value == 0 (contracts/HighStreetNftPool.sol#238)
HighStreetNftPool.getNftId(address,uint256) (contracts/HighStreetNftPool.sol#235-245) uses a dangerous strict equality:
        - value == UINT_16_MAX (contracts/HighStreetNftPool.sol#240)
HighStreetNftPool.getNftsBatch(address,uint256) (contracts/HighStreetNftPool.sol#338-361) uses a dangerous strict equality:
        - value == 0 (contracts/HighStreetNftPool.sol#352)
HighStreetNftPool.getNftsBatch(address,uint256) (contracts/HighStreetNftPool.sol#338-361) uses a dangerous strict equality:
        - value == UINT_16_MAX (contracts/HighStreetNftPool.sol#354)
HighStreetNftPool.getNftsBatchLength(address) (contracts/HighStreetNftPool.sol#371-376) uses a dangerous strict equality:
        - users[_user].list.length == 0 (contracts/HighStreetNftPool.sol#372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in HighStreetNftPool._stake(address,uint256[]) (contracts/HighStreetNftPool.sol#500-537):
        External calls:
        - IERC721(poolToken).safeTransferFrom(_staker,address(this),_nftIds[i]) (contracts/HighStreetNftPool.sol#521)
        State variables written after the call(s):
        - user.list.push(UINT_16_MAX) (contracts/HighStreetNftPool.sol#524)
        - user.list.push(uint16(_nftIds[i])) (contracts/HighStreetNftPool.sol#526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

HighStreetNftPool._stake(address,uint256[]) (contracts/HighStreetNftPool.sol#500-537) has external calls inside a loop: IERC721(poolToken).safeTransferFrom(_staker,address(this),_nftIds[i]) (contracts/HighStreetNftPool.sol#521)
HighStreetNftPool._unstake(address,uint256[]) (contracts/HighStreetNftPool.sol#545-590) has external calls inside a loop: IERC721(poolToken).safeTransferFrom(address(this),_staker,nfts[i]) (contracts/HighStreetNftPool.sol#578)
HighStreetNftPool._emergencyWithdraw(address,uint256[]) (contracts/HighStreetNftPool.sol#639-681) has external calls inside a loop: IERC721(poolToken).safeTransferFrom(address(this),_staker,nfts[i]) (contracts/HighStreetNftPool.sol#670)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in HighStreetNftPool._unstakeReward(address,uint256,uint256) (contracts/HighStreetNftPool.sol#599-631):
        External calls:
        - mintYieldTo(msg.sender,_amount) (contracts/HighStreetNftPool.sol#627)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetNftPool.sol#844)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - mintYieldTo(msg.sender,_amount) (contracts/HighStreetNftPool.sol#627)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - UnstakedReward(msg.sender,_staker,_amount) (contracts/HighStreetNftPool.sol#630)
Reentrancy in HighStreetNftPool.mintYieldTo(address,uint256) (contracts/HighStreetNftPool.sol#689-694):
        External calls:
        - transferHighToken(_to,_amount) (contracts/HighStreetNftPool.sol#691)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetNftPool.sol#844)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferHighToken(_to,_amount) (contracts/HighStreetNftPool.sol#691)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - MintYield(_to,_amount) (contracts/HighStreetNftPool.sol#693)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

HighStreetNftPool.unstakeReward(uint256,uint256) (contracts/HighStreetNftPool.sol#421-430) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now256() > stakeDeposit.lockedUntil,deposit not yet unlocked) (contracts/HighStreetNftPool.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly

        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - 0.8.9 (contracts/HighStreetNftPool.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/HighStreetNftPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter HighStreetNftPool.pendingYieldRewards(address)._staker (contracts/HighStreetNftPool.sol#190) is not in mixedCase
Parameter HighStreetNftPool.balanceOf(address)._user (contracts/HighStreetNftPool.sol#221) is not in mixedCase
Parameter HighStreetNftPool.getNftId(address,uint256)._user (contracts/HighStreetNftPool.sol#235) is not in mixedCase
Parameter HighStreetNftPool.getNftId(address,uint256)._index (contracts/HighStreetNftPool.sol#235) is not in mixedCase
Parameter HighStreetNftPool.getNftsLength(address)._user (contracts/HighStreetNftPool.sol#255) is not in mixedCase
Parameter HighStreetNftPool.getDeposit(address,uint256)._user (contracts/HighStreetNftPool.sol#269) is not in mixedCase
Parameter HighStreetNftPool.getDeposit(address,uint256)._depositId (contracts/HighStreetNftPool.sol#269) is not in mixedCase
Parameter HighStreetNftPool.getDepositsLength(address)._user (contracts/HighStreetNftPool.sol#282) is not in mixedCase
Parameter HighStreetNftPool.getDepositsBatch(address,uint256)._user (contracts/HighStreetNftPool.sol#296) is not in mixedCase
Parameter HighStreetNftPool.getDepositsBatch(address,uint256)._pageId (contracts/HighStreetNftPool.sol#296) is not in mixedCase
Parameter HighStreetNftPool.getDepositsBatchLength(address)._user (contracts/HighStreetNftPool.sol#321) is not in mixedCase
Parameter HighStreetNftPool.getNftsBatch(address,uint256)._user (contracts/HighStreetNftPool.sol#338) is not in mixedCase
Parameter HighStreetNftPool.getNftsBatch(address,uint256)._pageId (contracts/HighStreetNftPool.sol#338) is not in mixedCase
Parameter HighStreetNftPool.getNftsBatchLength(address)._user (contracts/HighStreetNftPool.sol#371) is not in mixedCase
Parameter HighStreetNftPool.stake(uint256[])._nftIds (contracts/HighStreetNftPool.sol#388) is not in mixedCase
Parameter HighStreetNftPool.unstake(uint256[])._listIds (contracts/HighStreetNftPool.sol#405) is not in mixedCase
Parameter HighStreetNftPool.unstakeReward(uint256,uint256)._depositId (contracts/HighStreetNftPool.sol#422) is not in mixedCase
Parameter HighStreetNftPool.unstakeReward(uint256,uint256)._amount (contracts/HighStreetNftPool.sol#423) is not in mixedCase
Parameter HighStreetNftPool.emergencyWithdraw(uint256[])._listIds (contracts/HighStreetNftPool.sol#439) is not in mixedCase
Parameter HighStreetNftPool.mintYieldTo(address,uint256)._to (contracts/HighStreetNftPool.sol#689) is not in mixedCase
Parameter HighStreetNftPool.mintYieldTo(address,uint256)._amount (contracts/HighStreetNftPool.sol#689) is not in mixedCase
Parameter HighStreetNftPool.tokenToReward(uint256,uint256)._token (contracts/HighStreetNftPool.sol#787) is not in mixedCase
Parameter HighStreetNftPool.tokenToReward(uint256,uint256)._rewardPerToken (contracts/HighStreetNftPool.sol#787) is not in mixedCase
Parameter HighStreetNftPool.rewardToToken(uint256,uint256)._reward (contracts/HighStreetNftPool.sol#800) is not in mixedCase
Parameter HighStreetNftPool.rewardToToken(uint256,uint256)._rewardPerToken (contracts/HighStreetNftPool.sol#800) is not in mixedCase
Parameter HighStreetNftPool.transferHighToken(address,uint256)._to (contracts/HighStreetNftPool.sol#842) is not in mixedCase
Parameter HighStreetNftPool.transferHighToken(address,uint256)._value (contracts/HighStreetNftPool.sol#842) is not in mixedCase
Variable HighStreetNftPool.HIGH (contracts/HighStreetNftPool.sol#43) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#19-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither. All the reentrancies flagged are false positives. All the external functions are protected with the nonReentrant modifier.

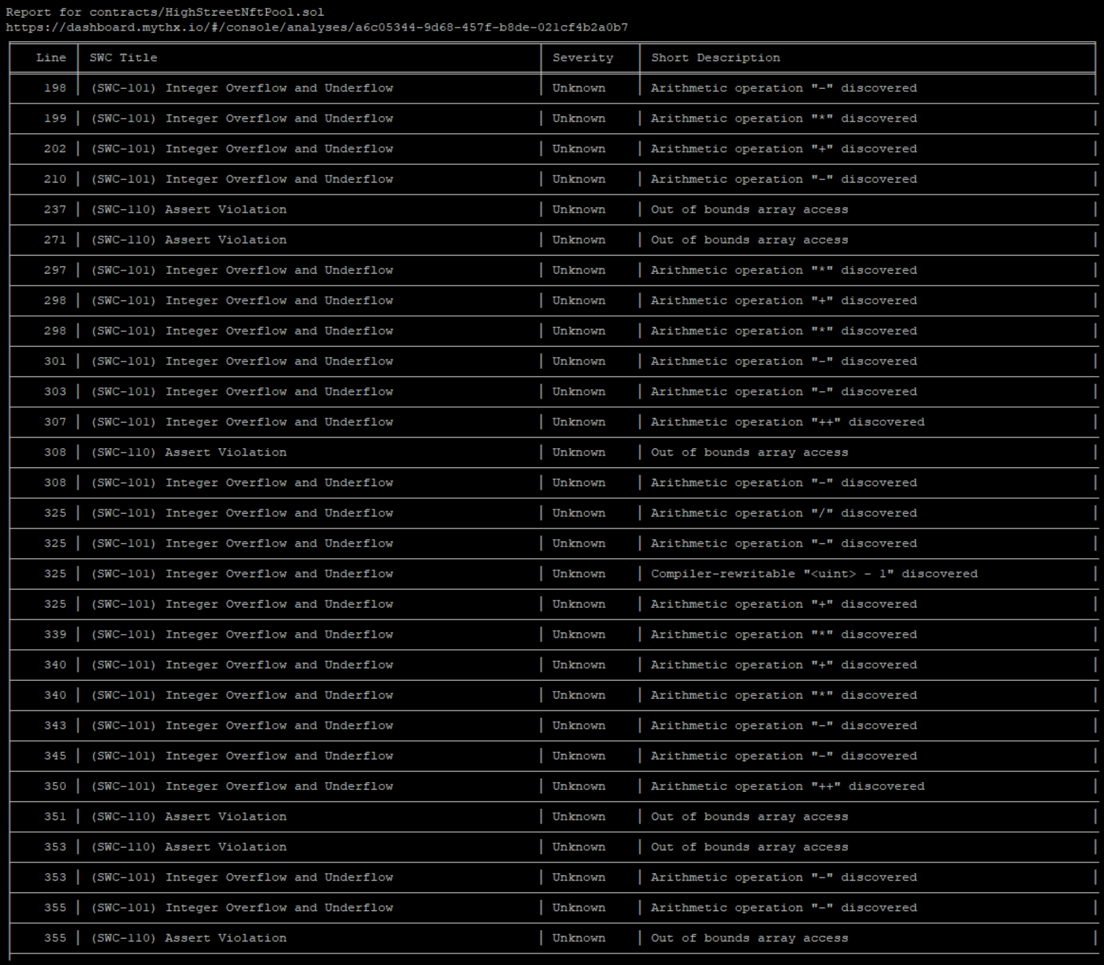
4.2 AUTOMATED SECURITY SCAN

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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

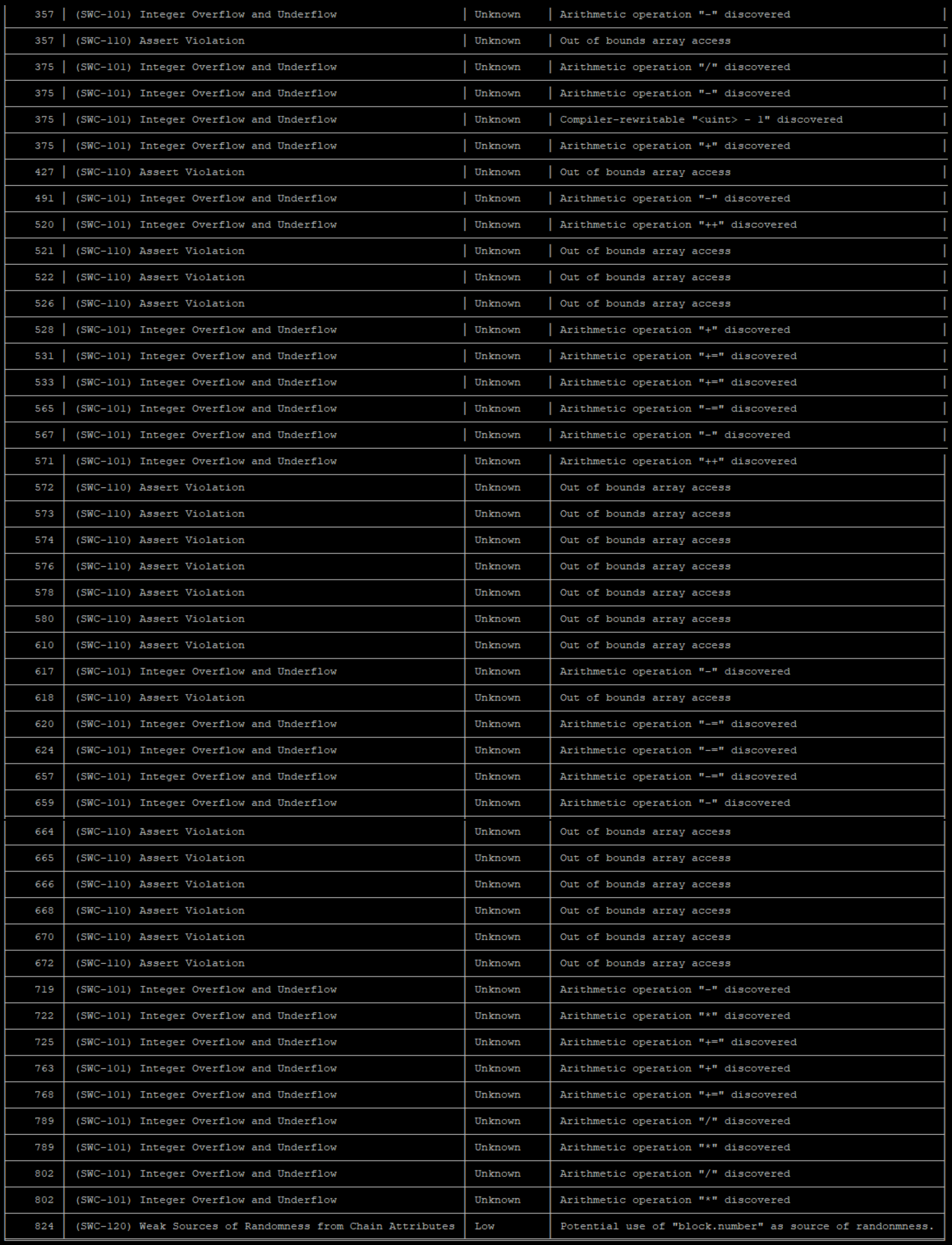
MythX results:

HighStreetNftPool.sol



```
Report for contracts/HighStreetNftPool.sol
https://dashboard.mythx.io/#/console/analyses/a6c05344-9d68-457f-b8de-021cf4b2a0b7
```

Line	SWC Title	Severity	Short Description
198	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
199	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
202	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
210	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
237	(SWC-110) Assert Violation	Unknown	Out of bounds array access
271	(SWC-110) Assert Violation	Unknown	Out of bounds array access
297	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
298	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
298	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
301	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
303	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
307	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
308	(SWC-110) Assert Violation	Unknown	Out of bounds array access
308	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
325	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
325	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
325	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
325	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
339	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
340	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
340	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
343	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
345	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
350	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
351	(SWC-110) Assert Violation	Unknown	Out of bounds array access
353	(SWC-110) Assert Violation	Unknown	Out of bounds array access
353	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
355	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
355	(SWC-110) Assert Violation	Unknown	Out of bounds array access



357	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
357	(SWC-110) Assert Violation	Unknown	Out of bounds array access
375	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
375	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
375	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
375	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
427	(SWC-110) Assert Violation	Unknown	Out of bounds array access
491	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
520	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
521	(SWC-110) Assert Violation	Unknown	Out of bounds array access
522	(SWC-110) Assert Violation	Unknown	Out of bounds array access
526	(SWC-110) Assert Violation	Unknown	Out of bounds array access
528	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
531	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
533	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
565	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
567	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
571	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
572	(SWC-110) Assert Violation	Unknown	Out of bounds array access
573	(SWC-110) Assert Violation	Unknown	Out of bounds array access
574	(SWC-110) Assert Violation	Unknown	Out of bounds array access
576	(SWC-110) Assert Violation	Unknown	Out of bounds array access
578	(SWC-110) Assert Violation	Unknown	Out of bounds array access
580	(SWC-110) Assert Violation	Unknown	Out of bounds array access
610	(SWC-110) Assert Violation	Unknown	Out of bounds array access
617	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
618	(SWC-110) Assert Violation	Unknown	Out of bounds array access
620	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
624	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
657	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
659	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
664	(SWC-110) Assert Violation	Unknown	Out of bounds array access
665	(SWC-110) Assert Violation	Unknown	Out of bounds array access
666	(SWC-110) Assert Violation	Unknown	Out of bounds array access
668	(SWC-110) Assert Violation	Unknown	Out of bounds array access
670	(SWC-110) Assert Violation	Unknown	Out of bounds array access
672	(SWC-110) Assert Violation	Unknown	Out of bounds array access
719	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
722	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
725	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
763	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
768	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
789	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
789	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
802	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
802	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
824	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

- Integer Overflows and Underflows flagged by MythX are false positives, as the contract is using Solidity 0.8.10 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- Assert violations are false positives.

- block.number is used but not as a source of randomness.

THANK YOU FOR CHOOSING

HALBORN