MatrixSwap - DEX aggregator

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 11th, 2021 - October 18th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/11/2021	Roberto Reigada
0.2	Document Edits	10/18/2021	Roberto Reigada
0.3	Document Review	10/22/2021	Gabi Urrutia
1.0	Remediation Plan	11/04/2021	Roberto Reigada
1.1	Remediation Plan Edits	11/29/2021	Roberto Reigada
1.2	Remediation Plan Review	11/29/2021	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

MatrixSwap engaged Halborn to conduct a security audit on their smart contracts beginning on October 11th, 2021 and ending on October 18th, 2021. The security assessment was scoped to the smart contracts provided in the Github repository Matrixswap/Nebuchadnezzar - main branch

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is 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 the MatrixSwap team.

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 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. It’s quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was 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 contracts

- TokenPrice.sol
- Box.sol
- ApprovedTokens.sol
- AllRouterSwap.sol
- All contracts inherited by these contracts

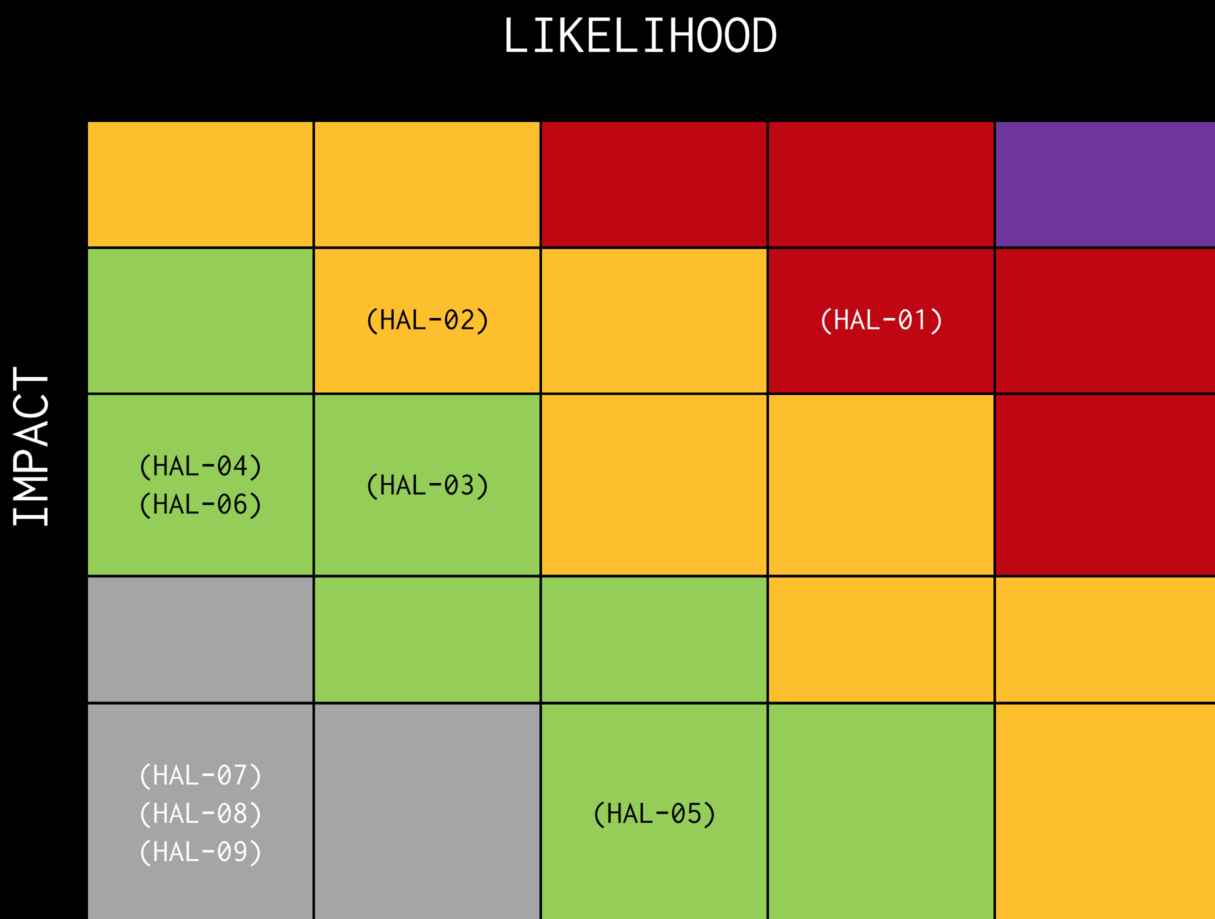
Commit ID: ab7712521e699c6a8a99275d51662ebecffd63b0

1st remediations Commit ID: a0ff1260d21d253bab3b4ee053d8bbaf79ea6f2c

2nd remediations Commit ID: 7fe0037de0bf9254b82e6726ff38dda1da2e57dc

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	4	3



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	High	SOLVED - 11/29/2021
HAL02 - UNCHECKED TRANSFER	Medium	SOLVED - 11/04/2021
HAL03 - UNUSED RETURN	Low	SOLVED - 11/04/2021
HAL04 - MISSING RE-ENTRANCY PROTECTION	Low	SOLVED - 11/04/2021
HAL05 - USE OF BLOCK.TIMESTAMP	Low	SOLVED - 11/04/2021
HAL06 - FLOATING PRAGMA	Low	SOLVED - 11/04/2021
HAL07 - EXTERNAL CALLS WITHIN A LOOP	Informational	SOLVED - 11/04/2021
HAL08 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 11/04/2021
HAL09 - USE OF ASSERT FUNCTION	Informational	SOLVED - 11/04/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - HIGH

Description:

Some tokens (like USDT) don't correctly implement the EIP20 standard and their transfer/transferFrom functions return void, instead of a success boolean. Calling these functions with the correct EIP20 function signatures will always revert as it is done in the contract AllRouterSwap.sol.

Tokens that don't correctly implement the latest EIP20 spec, like USDT, will be unusable in the smart contract as they revert the transaction because of the missing return value.

We recommend using OpenZeppelin's SafeERC20 versions with the safeTransfer and safeTransferFrom functions that handle the return value check as well as non-standard-compliant tokens. On the other hand, in the 1st remediations Commit ID a0ff1260d21d253bab3b4ee053d8bbaf79ea6f2c the same issue with the approve function was added into the code.

Proof of Concept:

```
SwapHalbornTest (test_swap) deployed and initialized

Calling -> test_swap.swapToETHMock(erc20token.address, 10)
Transaction sent: 0x1b9bbf7b43d9c16ee90d979fbcee1a5e93921cfd5384a0b185b888c6fce4dca5
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 32
  SwapHalbornTest.swapToETHMock confirmed   Block: 13546090   Gas used: 46251 (0.69%)

Call trace for '0x1b9bbf7b43d9c16ee90d979fbcee1a5e93921cfd5384a0b185b888c6fce4dca5':
Initial call cost  [21572 gas]
SwapHalbornTest.swapToETHMock  0:436  [1437 / 24679 gas]
│
└── MyToken.approve  [CALL]  132:344  [1117 / 23242 gas]
    │   ├── address: erc20token.address
    │   ├── value: 0
    │   ├── input arguments:
    │   │   ├── spender: test_swap.address
    │   │   └── amount: 10
    │   └── return value: True
    │
    └── ERC20Upgradeable.approve  231:324  [50 / 22125 gas]
        └── ERC20Upgradeable._approve  239:316  [22075 gas]
```

```
Calling -> test_swap.swapToETHMock(usdt.address, 10)
Transaction sent: 0x0cd496436c7e8fa9790cb35c120029692476d3d098aec9a46b70f4dc25b8d451
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 33
  SwapHalbornTest.swapToETHMock confirmed (reverted)   Block: 13546091   Gas used: 47806 (0.71%)

Call trace for '0x0cd496436c7e8fa9790cb35c120029692476d3d098aec9a46b70f4dc25b8d451':
Initial call cost  [21572 gas]
SwapHalbornTest.swapToETHMock  0:398  [1293 / 26234 gas]
│
└── TetherToken.approve  [CALL]  132:352  [1885 / 24941 gas]
    │   ├── address: usdt.address
    │   ├── value: 0
    │   └── input arguments:
    │       ├── _spender: test_swap.address
    │       └── _value: 10
    │
    └── StandardToken.approve  233:345  [23056 gas]
```

Code Location:

- require(IERC20(inputToken).transfer(msg.sender, _amountIn), errorMessage); (AllRouterSwap.sol#296)
- require(IERC20(inputToken).transfer(msg.sender, _amountIn), errorMessage); (AllRouterSwap.sol#308)

1st remediations Commit ID a0ff1260d21d253bab3b4ee053d8bbaf79ea6f2c:

Listing 1: AllRouterSwap.sol

```
bool approved = IERC20Upgradeable(_tokenAddress).approve(address(
    _router), _amountIn.add(100000000000000000000));
require(approved, errorMessage);
```

Listing 2: AllRouterSwap.sol

```
bool approved = IERC20Upgradeable(_firstPath).approve(address(
    _router), _amountIn);
require(approved, errorMessage);
```

Listing 3: AllRouterSwap.sol

```
bool approved = IERC20Upgradeable(_tokenAddress).approve(address(
    _router), _amountIn.add(100000000000000000000));
require(approved, errorMessage);
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

It is recommended to use SafeERC20: safeTransfer and safeApprove.

Remediation Plan:

SOLVED: The MatrixSwap team correctly uses now safeTransfer and safeApprove. The contract is now compatible with non-standard ERC20 tokens like USDT. Fixed in commit ID 7fe0037de0bf9254b82e6726ff38dda1da2e57dc.

3.2 (HAL-02) UNCHECKED TRANSFER - MEDIUM

Description:

In the contract AllRouterSwap.sol the return value of some external transfer/transferFrom calls are not checked. Several tokens do not revert in case of failure and return false. If one of these tokens is used, a deposit would not revert if the transfer fails, and an attacker could deposit tokens for free.

Code Location:

- `IERC20(inputToken).transfer(owner,_fee)` (AllRouterSwap.sol#148)
- `IERC20(_firstPath).transfer(owner,_fee)` (AllRouterSwap.sol#176)
- `IERC20(_tokenAddress).transfer(owner,_fee)` (AllRouterSwap.sol#196)
- `IERC20(_token).transferFrom(msg.sender,address(this),_amountIn)` (AllRouterSwap.sol#211)
- `IERC20(inputToken).transferFrom(msg.sender,address(this),_amountIn[i])` (AllRouterSwap.sol#331)

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Remediation Plan:

SOLVED: The MatrixSwap team correctly uses now safeTransfer and safeTransferFrom.

3.3 (HAL-03) UNUSED RETURN - LOW

Description:

The return value of some external calls are not stored in a local or state variable. In the contract AllRouterSwap.sol there are instances were external methods are being called and the return values are ignored.

Code Location:

- IERC20(_tokenAddress).approve(address(_router),_amountIn.add (100000000000000000000)) (AllRouterSwap.sol#150)
- IERC20(_firstPath).approve(address(_router),_amountIn) (AllRouterSwap.sol#179)
- _router.swapExactTokensForETH(_amountIn,_minimumAmountOut,_path, msg.sender,block.timestamp + 180) (AllRouterSwap.sol#180-186)
- IERC20(_tokenAddress).approve(address(_router),_amountIn.add (100000000000000000000)) (AllRouterSwap.sol#199)

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Ensure that all the return values of the function calls are used. Add a return value check to avoid an unexpected crash of the contract.

Remediation Plan:

SOLVED: The MatrixSwap team correctly uses now safeApprove.

3.4 (HAL-04) MISSING RE-ENTRANCY PROTECTION - LOW

Description:

It was identified that the contract AllRouterSwap is missing nonReentrant guard in the public functions swap and swapToETH. Even if the functions are following the check-effects-interactions pattern we still recommend to use a mutex in order to be protected against cross-function reentrancy attacks. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against the Reentrancy attacks.

Code Location:

Listing 4: AllRouterSwap.sol

```
function swap(uint[] memory _amountIn, address[] memory _token,
    uint8[][] memory _swapRoute, bool[] memory _isEth, address
    _tokenTarget, bool _isMultiToSingleToken, bool _isSingleEth,
    uint[] memory _minimumAmountOut, bool _isNukeTx) public payable
{
    isSingleEth = _isSingleEth;
    isMultiToSingleToken = _isMultiToSingleToken;
    isMultiSwap = _token.length > 1;
    isNukeTx = _isNukeTx;

    for (uint8 i = 0; i < _token.length; i++)
    {
        inputToken = SwapLibrary._getToken(_token[i], _tokenTarget
            , _isMultiToSingleToken, true);
        outputToken = SwapLibrary._getToken(_token[i],
            _tokenTarget, _isMultiToSingleToken, false);

        if (!_isSingleEth && !_isMultiToSingleToken && _isEth[i])
        {
```

```
            _everySwapToEth(_amountIn[i], inputToken, _swapRoute[i], _minimumAmountOut[i]);
            continue;
        }
        else if (!(SwapLibrary._isInputAllEth(isMultiToSingleToken, isSingleEth) || (_isEth[i] && isMultiToSingleToken)))
        {
            IERC20(inputToken).transferFrom(msg.sender, address(this), _amountIn[i]);
        }

        // 1 Router, 2 Tokens
        if (_swapRoute[i][0] == _swapRoute[i][2] && _swapRoute[i][1] == 0)
        {
            _twoTokensSwap(routerList[_swapRoute[i][0]-1], _amountIn[i], _isEth[i], _minimumAmountOut[i]);
        }

        // 1 Router, 3 Tokens
        else if (_swapRoute[i][0] == _swapRoute[i][2] && _swapRoute[i][1] > 0)
        {
            _threeTokensSwap(commonTokens[_swapRoute[i][1]-1], routerList[_swapRoute[i][0]-1], _amountIn[i], _isEth[i], _minimumAmountOut[i]);
        }

        // 2 Routers, 3 Tokens
        else if (_swapRoute[i][0] != _swapRoute[i][2])
        {
            _twoRoutersSwap(commonTokens[_swapRoute[i][1]-1], routerList[_swapRoute[i][0]-1], routerList[_swapRoute[i][2]-1], _amountIn[i], _isEth[i], _minimumAmountOut[i]);
        }
    }
}

function swapToETH(uint[] memory _amountIn, address[] memory _token, uint8[][] memory _swapRoute, uint[] memory _minimumAmountOut) public
{
    isMultiSwap = _token.length > 1;
```

```
    for (uint i = 0; i < _token.length; i++)
    {
        inputToken = _token[i];
        outputToken = addressWETH;
        _everySwapToEth(_amountIn[i], _token[i], _swapRoute[i],
            _minimumAmountOut[i]);
    }
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

We recommend using ReentrancyGuard through the nonReentrant modifier.

Remediation Plan:

SOLVED: The MatrixSwap team correctly added the nonReentrant modifier.

3.5 (HAL-05) USE OF BLOCK.TIMESTAMP - LOW

Description:

During a manual review, we noticed the use of block.timestamp. The contract developers should be aware that this does not mean current time. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. The use of block.timestamp creates a risk that miners could perform time manipulation to influence price oracles. Miners can modify the timestamp by up to 900 seconds.

Code Location:

- block.timestamp + 180 (AllRouterSwap.sol#139)
- block.timestamp + 180 (AllRouterSwap.sol#156)
- block.timestamp + 180 (AllRouterSwap.sol#185)
- block.timestamp + 180 (AllRouterSwap.sol#205)

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

Use block.number instead of block.timestamp or now to reduce the risk of Maximal Extractable Value (MEV) attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

SOLVED: The MatrixSwap team is not using block.timestamp anymore in the smart contract.

3.6 (HAL-06) FLOATING PRAGMA - LOW

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Code Location:

Listing 5

```
utils/SafeMath.sol:3:pragma solidity ^0.8.0;
utils/Initializable.sol:3:pragma solidity ^0.8.0;
TokenPrice.sol:2:pragma solidity ^0.8.0;
library/SwapLibrary.sol:2:pragma solidity ^0.8.0;
Box.sol:3:pragma solidity ^0.8.0;
ApprovedTokens.sol:2:pragma solidity ^0.8.7;
AllRouterSwap.sol:2:pragma solidity ^0.8.7;
AdminBox.sol:3:pragma solidity ^0.8.0;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider locking the pragma version. It is not recommended to use a floating pragma in production. It is possible to lock the pragma by fixing the version both in truffle-config.js for Truffle framework or in hardhat.config.js for HardHat framework.

Remediation Plan:

SOLVED: The MatrixSwap team correctly locked the pragma version to the 0.8.7 version.

3.7 (HAL-07) EXTERNAL CALLS WITHIN A LOOP - INFORMATIONAL

Description:

Calls inside a loop might lead to a Denial of Service attack. If the i variable iterates up to a very high value or is reset by the external functions called, this could cause a Denial of Service.

Code Location:

ApprovedTokens.sol

Listing 6: ApprovedTokens.sol (Lines 29,31,35)

```
23 function searchApprovedTokens(address[] memory _tokens)
24     external
25     view
26     returns (address[] memory)
27 {
28     address[] memory _approvedTokens = new address[](_tokens.
           length);
29     for (uint16 i = 0; i < _tokens.length; i++) {
30         console.log("token:", _tokens[i]);
31         uint256 _allowance = IERC20(_tokens[i]).allowance(
32             msg.sender,
33             swapContract
34         );
35         uint256 _balance = IERC20(_tokens[i]).balanceOf(msg.sender
               );
36         // console.log("_allowance:", _allowance);
37         // console.log("_balance:", _balance);
38
39         if (_allowance > _balance) {
40             _approvedTokens[i] = (_tokens[i]);
41         }
42     }
43     return _approvedTokens;
44 }
```

AllRouterSwap.sol

Listing 7: AllRouterSwap.sol (Lines 319,321,322,326,329,331,337,343,349,357,361)

```
312 function swap(uint[] memory _amountIn, address[] memory _token, uint8[][] memory _swapRoute, bool[] memory _isEth, address _tokenTarget, bool _isMultiToSingleToken, bool _isSingleEth, uint[] memory _minimumAmountOut, bool _isNukeTx) public payable
313 {
314     isSingleEth = _isSingleEth;
315     isMultiToSingleToken = _isMultiToSingleToken;
316     isMultiSwap = _token.length > 1;
317     isNukeTx = _isNukeTx;
318 
319     for (uint8 i = 0; i < _token.length; i++)
320     {
321         inputToken = SwapLibrary._getToken(_token[i], _tokenTarget, _isMultiToSingleToken, true);
322         outputToken = SwapLibrary._getToken(_token[i], _tokenTarget, _isMultiToSingleToken, false);
323 
324         if (!_isSingleEth && !_isMultiToSingleToken && _isEth[i])
325         {
326             _everySwapToEth(_amountIn[i], inputToken, _swapRoute[i], _minimumAmountOut[i]);
327             continue;
328         }
329         else if (!(SwapLibrary._isInputAllEth(isMultiToSingleToken, isSingleEth) || (_isEth[i] && isMultiToSingleToken)))
330         {
331             IERC20(inputToken).transferFrom(msg.sender, address(this), _amountIn[i]);
332         }
333 
334         // 1 Router, 2 Tokens
335         if (_swapRoute[i][0] == _swapRoute[i][2] && _swapRoute[i][1] == 0)
336         {
337             _twoTokensSwap(routerList[_swapRoute[i][0]-1], _amountIn[i], _isEth[i], _minimumAmountOut[i]);
338         }
339 
340         // 1 Router, 3 Tokens
341         else if (_swapRoute[i][0] == _swapRoute[i][2] &&
```

```
                _swapRoute[i][1] > 0)
        {
            _threeTokensSwap(commonTokens[_swapRoute[i][1]-1],
                routerList[_swapRoute[i][0]-1], _amountIn[i],
                _isEth[i], _minimumAmountOut[i]);
        }

        // 2 Routers, 3 Tokens
        else if (_swapRoute[i][0] != _swapRoute[i][2])
        {
            _twoRoutersSwap(commonTokens[_swapRoute[i][1]-1],
                routerList[_swapRoute[i][0]-1], routerList[
                _swapRoute[i][2]-1], _amountIn[i], _isEth[i],
                _minimumAmountOut[i]);
        }
    }
}

function swapToETH(uint[] memory _amountIn, address[] memory
    _token, uint8[][] memory _swapRoute, uint[] memory
    _minimumAmountOut) public
{
    isMultiSwap = _token.length > 1;
    for (uint i = 0; i < _token.length; i++)
    {
        inputToken = _token[i];
        outputToken = addressWETH;
        _everySwapToEth(_amountIn[i], _token[i], _swapRoute[i],
            _minimumAmountOut[i]);
    }
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If possible, use pull over push strategy for external calls or limit the max. size of the arrays being iterated.

Remediation Plan:

SOLVED: The MatrixSwap team correctly limited the maximum iterations of the loops by casting the i variable to uint8 and adding a require statement. For example:

Listing 8: AllRouterSwap.sol (Lines 381,387)

```
379 function swapToETH(uint[] memory _amountIn, address[] memory
        _token, uint8[][] memory _swapRoute, uint[] memory
        _minimumAmountOut, uint[] memory _uintArray) external
        nonReentrant
380 {
381     require(_token.length < uint8(40), "Block gas limit exceeded")
            ;
382     // uint[] memory _uintArray = [_discount, _deadline]
383     isMultiSwap = _token.length > 1;
384     discount = _uintArray[0];
385     deadline = _uintArray[1];
386
387     for (uint8 i = 0; i < _token.length; i++)
388     {
389         inputToken = _token[i];
390         outputToken = addressWETH;
391         _everySwapToEth(_amountIn[i], _token[i], _swapRoute[i],
                _minimumAmountOut[i]);
392     }
393 }
```

3.8 (HAL-08) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the contract Box.sol there are functions marked as public but they are never directly called within the same contract or in any of its descendants:

Box.sol
- store(uint256) (Box.sol#12-15)
- retrieve() (Box.sol#18-20)
- decrement() (Box.sol#22-25)

AllRouterSwap.sol
- initialize(address,address,address[],address[]) (AllRouterSwap.sol#64-77)
- changeOwner(address) (AllRouterSwap.sol#88-91)
- addCommonToken(address) (AllRouterSwap.sol#93-96)
- showOwner() (AllRouterSwap.sol#98-101)
- swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (AllRouterSwap.sol#312-352)
- swapToETH(uint256[],address[],uint8[][],uint256[]) (AllRouterSwap.sol#354-363)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the function is not intended to be called internally or by descendants, it is better to mark all these functions as external to reduce gas costs.

Remediation Plan:

SOLVED: The MatrixSwap team set all the mentioned functions as external to reduce gas costs.

3.9 (HAL-09) USE OF ASSERT FUNCTION - INFORMATIONAL

Description:

In the contract AllRouterSwap.sol the function assert is used. As per Solidity documentation:

The assert function creates an error of type Panic(uint256). Assert should only be used to test for internal errors, and to check invariants. Properly functioning code should never create a Panic, not even on invalid external input.

Code Location:

Listing 9: Assertl (Lines 125)

```
121 if (SwapLibrary._isWrapUnwrap(_path[0], _path[1], addressWETH))
122 {
123     WETH = IWETH(_path[0]);
124     WETH.deposit{value: _amountIn}();
125     assert(WETH.transfer(msg.sender, _amountIn));
126 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use a require statement instead.

Remediation Plan:

SOLVED: The MatrixSwap team is correctly using now a require statement.

MANUAL TESTING

4.1 TESTING CONTRACT INITIALIZATION FRONT-RUNNING

As the contracts Swap and AdminBox make use of initialize functions we have checked if they could be front-run but this is not the case as they have a constructor:

“Do not leave an implementation contract uninitialized. An uninitialized implementation contract can be taken over by an attacker, which may impact the proxy. You can either invoke the initializer manually, or you can include a constructor to automatically mark it as initialized when it is deployed”:

AdminBox contract

```
contract AdminBox is Initializable {
    uint256 private _value;
    address private _admin;
    address[] private _admin2;
    uint256 private _addedNumber;

    // Emitted when the stored value changes
    event ValueChanged(uint256 value);

    function initialize(address admin, address[] memory admin2) public initializer {
        _admin = admin;
        _admin2 = admin2;
        _addedNumber = 69;
    }

    /// @custom:oz-upgrades-unsafe-allow constructor
    constructor() initializer {}
```

Swap contract

```
    function initialize(address _owner, address _admin, address[] memory _routerList, addre
    {
        errorMessage = "Tx Fail";
        amount = new uint[](2);

        owner = _owner;
        admin = _admin;
        commonTokens = _commonTokens;

        for (uint8 i = 0; i < _routerList.length; i++)
        {
            routerList.push(IUniswapV2Router02(_routerList[i]));
        }
    }

    /// @custom:oz-upgrades-unsafe-allow constructor
    constructor() initializer {}
```

Below we can see how they were automatically initialized right after being deployed:

```
>>> swapcontract = owner.deploy(Swap)
Transaction sent: 0x9eb75cb104ab52298324ddb309a334805eddc5aecaccf46bdd22f25d5f2de4d0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Swap.constructor confirmed   Block: 13449735   Gas used: 2882072 (42.88%)
  Swap deployed at: 0x201d8D501EDAcA18dB06DccBBAadf485c634fbF2

>>> swapcontract.initialize(owner.address, user1.address, ['0xA102072A4C07F06EC3B4900FDC4C7B80b6c57429', '0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506',
 '0xa5E0829CaCEd8fFDD4De3c43696c57F7D7A678ff', '0xC0788A3aD43d79aa53B09c2EaCc313A787d1d607', '0x6466CDC2615f111dc6Dbd0Ef6fDD209F5Ff8d184',
 '0x93bcDc45f7e62f89a8e901DC4A0E2c6C427D9F25', '0xC60aE14F2568b102F8Ca6266e8799112846DD088', '0x3a1D87f206D12415f5b0A33E786967680AAb4f6d',
 '0x5C6EC38fb0e2609672BDf628B1fD605A523E5923', '0x94930a328162957FF1dd48900aF67B5439336cBD'],
 ['0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270', '0x7ceB23fD6bC0adD59E62ac25578270cFf1b9f619', '0x2791Bca1f2de4661ED88A30C99A7a9449Aa84174',
 '0xc2132D05D31c914a87C6611C10748AEb04B58e8F', '0x8f3Cf7ad23Cd3CaDbD9735AFf958023239c6A063'])
Transaction sent: 0xf892920f7834e6fc30de0f88b38d858a01229aca5cc853c02056561e83bbe7cf
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  Swap.initialize confirmed (Initializable: contract is already initialized)   Block: 13449736   Gas used: 33166 (0.49%)

<Transaction '0xf892920f7834e6fc30de0f88b38d858a01229aca5cc853c02056561e83bbe7cf'>
>>> adminbox = owner.deploy(AdminBox)
Transaction sent: 0x97aa59423ae610348982a303236ec78e45c35fa1529264c1c570708a2f0f5f43
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  AdminBox.constructor confirmed   Block: 13449737   Gas used: 341605 (5.08%)
  AdminBox deployed at: 0xbb7826c89f7EE9f6807e6B6243855d788b358074

>>> adminbox.initialize(owner.address, [user1.address])
Transaction sent: 0x6582a5dc3bd2ff2ba4ad8e9990dc05fa84d7c92d3d20117c9d00246f5fd5bc6c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  AdminBox.initialize confirmed (Initializable: contract is already initialized)   Block: 13449738   Gas used: 24714 (0.37%)

<Transaction '0x6582a5dc3bd2ff2ba4ad8e9990dc05fa84d7c92d3d20117c9d00246f5fd5bc6c'>
```

MANUAL TESTING

4.2 TESTING CONTRACT UPGRADEABILITY

In this test we wanted to test the upgradeability of the contract. In order to do that we have created a SwapV2 contract which adds a new simple getter function:

Listing 10: SwapV2 - getisSingleEth()

```
function get_isSingleEth() public view returns (bool){
    return isSingleEth;
}
```

SwapV1 was deployed. As the next step, the owner of the SwapV1 contract was updated by calling changeOwner function. The new owner is accounts [1]. Then using a proxy upgrade pattern, we upgraded the Swap contract to the SwapV2, checked that the owner of the SwapV2 contract was still accounts[1] and called the new function:

```
>>> account = accounts[0]
>>> swap = Swap.deploy({"from": account})
Transaction sent: 0xd402acf4cd58f6ea6d941d8383dbd49c14ae3db7108726f08c5f28916ca21622
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Swap.constructor confirmed   Block: 13455454   Gas used: 2928244 (43.56%)
  Swap deployed at: 0x98c7497555271F2E08E32ab0Fb1E4d17EE51876E

>>> proxy_admin = ProxyAdmin.deploy({"from": account})
Transaction sent: 0x581928659c9f40a9ba5ae178f83812c45c802e5387b2a3510f8a9ed9af0fd703
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  ProxyAdmin.constructor confirmed   Block: 13455455   Gas used: 471462 (7.01%)
  ProxyAdmin deployed at: 0xb8CAf26Fb113D011A021BE0D77Ff5562dcb91A72

>>> swap_encoded_initializer_function = encode_function_data(swap.initialize, account.address, account.address, [], [])
>>> proxy = TransparentUpgradeableProxy.deploy(swap.address, proxy_admin.address, swap_encoded_initializer_function, {"from": account, "gas_limit": 1000000})
Transaction sent: 0x593925237dce77060c44d7fbeb38cdf9fe225c1f20d77bd9b867e05516b0eecc
  Gas price: 0.0 gwei   Gas limit: 1000000   Nonce: 2
  TransparentUpgradeableProxy.constructor confirmed   Block: 13455456   Gas used: 704818 (70.48%)
  TransparentUpgradeableProxy deployed at: 0x9f61590eD0321cDe9099680FE1BFe98A45CA4Cf7

>>> proxy_box = Contract.from_abi("Swap", proxy.address, Swap.abi)
>>> proxy_box.changeOwner(accounts[1].address, {'from': account})
Transaction sent: 0x55fd72027414be3c944640efd6615a5772b4b5fa3747077497decf6f4540eec9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  Transaction confirmed   Block: 13455457   Gas used: 31275 (0.47%)

<Transaction '0x55fd72027414be3c944640efd6615a5772b4b5fa3747077497decf6f4540eec9'>
>>> output.redd("proxy_box.showOwner() -> " + str(proxy_box.showOwner()))
proxy_box.showOwner() -> 0x8f95a4d83337d1aBcbBAe42d503D551d2f0E5bfa
>>> a[1]
<Account '0x8f95a4d83337d1aBcbBAe42d503D551d2f0E5bfa'>
>>> swap_v2 = SwapV2.deploy({"from": account})
Transaction sent: 0x75da008fab729be775ae72196b982a1817fc5c52284df40c4eadb2ca8080aa77
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  SwapV2.constructor confirmed   Block: 13455458   Gas used: 2897292 (43.10%)
  SwapV2 deployed at: 0x87CED4EbCB72766A111909e757D97249862eBd46

>>> upgrade(account, proxy, swap_v2, proxy_admin_contract=proxy_admin)
Transaction sent: 0x7cd0b1130011049 1f802f95f0289079e20a8333b51926c68ea6d6af87a5073a2
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 5
  ProxyAdmin.upgrade confirmed   Block: 13455459   Gas used: 33800 (0.50%)

<Transaction '0x7cd0b11300110491f802f95f0289079e20a8333b51926c68ea6d6af87a5073a2'>
>>> proxy_box = Contract.from_abi("SwapV2", proxy.address, SwapV2.abi)
>>> output.redd("proxy_box.get_isSingleEth({'from': account}) -> " + str(proxy_box.get_isSingleEth({'from': account})))
proxy_box.get_isSingleEth({'from': account}) -> False
>>> output.redd("proxy_box.showOwner() -> " + str(proxy_box.showOwner()))
proxy_box.showOwner() -> 0x8f95a4d83337d1aBcbBAe42d503D551d2f0E5bfa
>>> a[1]
<Account '0x8f95a4d83337d1aBcbBAe42d503D551d2f0E5bfa'>
```

It is also possible, as included in the test cases of the project, to upgrade the contract using the OpenZeppelin Upgrades Plugins.

AUTOMATED TESTING

AUTOMATED TESTING

5.1 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:

TokenPrice.sol

```
INFO:Detectors:
TokenPrice._getPriceOutputCommonToken(uint256,address[]).amounts (contracts/TokenPrice.sol#181) is a local variable never initialized
TokenPrice._getPriceDiffRouters(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#160) is a local variable never initialized
TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#129) is a local variable never initialized
TokenPrice._getPriceTwoTokens(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#108) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
TokenPrice._getPriceTwoTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#103-116) ignores return value by _router.getAmountsOut(_amountIn,_path) (contracts/TokenPrice.sol#108-115)
TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#118-142) ignores return value by _router.getAmountsOut(_amountIn,_newPath) (contracts/TokenPrice.sol#129-140)
TokenPrice._getPriceDiffRouters(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#144-175) ignores return value by _routerIn.getAmountsOut(_amountIn,_newPath) (contracts/TokenPrice.sol#160-173)
TokenPrice._getPriceOutputCommonToken(uint256,address[]) (contracts/TokenPrice.sol#177-194) ignores return value by routerList[i].getAmountsOut(_amountIn,_path) (contracts/TokenPrice.sol#181-192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#118-142) has external calls inside a loop: _router.getAmountsOut(_amountIn,_newPath) (contracts/TokenPrice.sol#129-140)
TokenPrice._getPriceDiffRouters(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#144-175) has external calls inside a loop: _routerIn.getAmountsOut(_amountIn,_newPath) (contracts/TokenPrice.sol#160-173)
TokenPrice._getPriceOutputCommonToken(uint256,address[]) (contracts/TokenPrice.sol#177-194) has external calls inside a loop: routerList[i].getAmountsOut(_amountIn,_path) (contracts/TokenPrice.sol#181-192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Variable 'TokenPrice._getPriceTwoTokens(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#108)' in TokenPrice._getPriceTwoTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#103-116) potentially u
sed before declaration: _price[0] = amounts[1] (contracts/TokenPrice.sol#110)
Variable 'TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#129)' in TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#118-142) potential
ly used before declaration: _price[0] < amounts[2] (contracts/TokenPrice.sol#131)
Variable 'TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#129)' in TokenPrice._getPriceThreeTokens(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#118-142) potential
ly used before declaration: _price[0] = amounts[2] (contracts/TokenPrice.sol#133)
Variable 'TokenPrice._getPriceDiffRouters(IUniswapV2Router02,uint256,address[]).amounts (contracts/TokenPrice.sol#160)' in TokenPrice._getPriceDiffRouters(IUniswapV2Router02,uint256,address[]) (contracts/TokenPrice.sol#144-175) potential
ly used before declaration: fetchedPrice = _getPriceOutputCommonToken(amounts[1],_passedPath) (contracts/TokenPrice.sol#162)
Variable 'TokenPrice._getPriceOutputCommonToken(uint256,address[]).amounts (contracts/TokenPrice.sol#181)' in TokenPrice._getPriceOutputCommonToken(uint256,address[]) (contracts/TokenPrice.sol#177-194) potentially used before declaration
: _price[0] < amounts[1] (contracts/TokenPrice.sol#183)
Variable 'TokenPrice._getPriceOutputCommonToken(uint256,address[]).amounts (contracts/TokenPrice.sol#181)' in TokenPrice._getPriceOutputCommonToken(uint256,address[]) (contracts/TokenPrice.sol#177-194) potentially used before declaration
: _price[0] = amounts[1] (contracts/TokenPrice.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Pragma version^0.8.0 (contracts/TokenPrice.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
INFO:Detectors:
Parameter TokenPrice.getPrice(uint256,address[])._amountIn (contracts/TokenPrice.sol#44) is not in mixedCase
Parameter TokenPrice.getPrice(uint256,address[])._path (contracts/TokenPrice.sol#44) is not in mixedCase
Variable TokenPrice.ETH (contracts/TokenPrice.sol#12) is not in mixedCase
Variable TokenPrice.WETH (contracts/TokenPrice.sol#13) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Box.sol

```
INFO:Detectors:
Pragma version^0.8.0 (contracts/Box.sol#3) 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
INFO:Detectors:
store(uint256) should be declared external:
        - Box.store(uint256) (contracts/Box.sol#12-15)
retrieve() should be declared external:
        - Box.retrieve() (contracts/Box.sol#18-20)
decrement() should be declared external:
        - Box.decrement() (contracts/Box.sol#22-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ApprovedTokens.sol

```
INFO:Detectors:
ApprovedTokens.searchApprovedTokens(address[]) (contracts/ApprovedTokens.sol#23-44) has external calls inside a loop: _allowance = IERC20(_tokens[i]).allowance(msg.sender,swapContract) (contracts/ApprovedTokens.sol#31-34)
ApprovedTokens.searchApprovedTokens(address[]) (contracts/ApprovedTokens.sol#23-44) has external calls inside a loop: _balance = IERC20(_tokens[i]).balanceOf(msg.sender) (contracts/ApprovedTokens.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version^0.8.7 (contracts/ApprovedTokens.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
INFO:Detectors:
Parameter ApprovedTokens.searchApprovedTokens(address[])._tokens (contracts/ApprovedTokens.sol#23) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

AllRouterSwap.sol

```
INFO:Detectors:
Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159) sends eth to arbitrary user
	Dangerous calls:
	- WETH.deposit{value: _amountIn}() (contracts/AllRouterSwap.sol#124)
	- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188) sends eth to arbitrary user
	Dangerous calls:
	- (success) = msg.sender.call{value: _amountIn}() (contracts/AllRouterSwap.sol#167)
Swap._returnBalance(bool,uint256) (contracts/AllRouterSwap.sol#299-310) sends eth to arbitrary user
	Dangerous calls:
	- (ethSuccess) = msg.sender.call{value: _amountIn}() (contracts/AllRouterSwap.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Swap._twoRoutersSwap(address,IUniswapV2Router02,IUniswapV2Router02,uint256,bool,uint256) (contracts/AllRouterSwap.sol#283-292):
	External calls:
	- amount = _swapToken(_firstRouter,_amountIn,SwapLibrary._path2(inputToken,_secondPath),address(this),inputToken,_isEth,0,true) (contracts/AllRouterSwap.sol#290)
		- WETH.deposit{value: _amountIn}() (contracts/AllRouterSwap.sol#124)
		- assert(bool)(WETH.transfer(msg.sender,_amountIn)) (contracts/AllRouterSwap.sol#125)
		- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
		- _router.swapExactETHForTokens{value: _amountIn}(_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#135-140)
		- IERC20(inputToken).transfer(owner,_fee) (contracts/AllRouterSwap.sol#148)
		- IERC20(_tokenAddress).approve(address(_router),_amountIn.add(1000000000000000000000)) (contracts/AllRouterSwap.sol#150)
		- _router.swapExactTokensForTokens(_amountIn,_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#151-157)
	- _swapToken(_secondRouter,amount[amount.length - 1],SwapLibrary._path2(_secondPath,outputToken),msg.sender,_secondPath,_isEth,_minimumAmountOut,false) (contracts/AllRouterSwap.sol#291)
		- WETH.deposit{value: _amountIn}() (contracts/AllRouterSwap.sol#124)
		- assert(bool)(WETH.transfer(msg.sender,_amountIn)) (contracts/AllRouterSwap.sol#125)
		- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
		- _router.swapExactETHForTokens{value: _amountIn}(_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#135-140)
		- IERC20(inputToken).transfer(owner,_fee) (contracts/AllRouterSwap.sol#148)
		- IERC20(_tokenAddress).approve(address(_router),_amountIn.add(1000000000000000000000)) (contracts/AllRouterSwap.sol#150)
		- _router.swapExactTokensForTokens(_amountIn,_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#151-157)
	External calls sending eth:
	- amount = _swapToken(_firstRouter,_amountIn,SwapLibrary._path2(inputToken,_secondPath),address(this),inputToken,_isEth,0,true) (contracts/AllRouterSwap.sol#290)
		- WETH.deposit{value: _amountIn}() (contracts/AllRouterSwap.sol#124)
		- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
		- _router.swapExactETHForTokens{value: _amountIn}(_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#135-140)
	- _swapToken(_secondRouter,amount[amount.length - 1],SwapLibrary._path2(_secondPath,outputToken),msg.sender,_secondPath,_isEth,_minimumAmountOut,false) (contracts/AllRouterSwap.sol#291)
		- WETH.deposit{value: _amountIn}() (contracts/AllRouterSwap.sol#124)
		- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
		- _router.swapExactETHForTokens{value: _amountIn}(_minimumAmountOut,_path,_to,block.timestamp + 180) (contracts/AllRouterSwap.sol#135-140)
	State variables written after the call(s):
	- _swapToken(_secondRouter,amount[amount.length - 1],SwapLibrary._path2(_secondPath,outputToken),msg.sender,_secondPath,_isEth,_minimumAmountOut,false) (contracts/AllRouterSwap.sol#291)
		- WETH = IWETH(_path[0]) (contracts/AllRouterSwap.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159) ignores return value by IERC20(inputToken).transfer(owner,_fee) (contracts/AllRouterSwap.sol#148)
Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188) ignores return value by IERC20(_firstPath).transfer(owner,_fee) (contracts/AllRouterSwap.sol#176)
Swap._swapToEthInitial(IUniswapV2Router02,address,uint256,address[],uint256) (contracts/AllRouterSwap.sol#190-207) ignores return value by IERC20(_tokenAddress).transfer(owner,_fee) (contracts/AllRouterSwap.sol#196)
Swap._everySwapToEth(uint256,address,uint8[],uint256) (contracts/AllRouterSwap.sol#209-230) ignores return value by IERC20(_token).transferFrom(msg.sender,address(this),_amountIn) (contracts/AllRouterSwap.sol#211)
Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (contracts/AllRouterSwap.sol#312-352) ignores return value by IERC20(inputToken).transferFrom(msg.sender,address(this),_amountIn[i]) (contracts/AllRouterSwa
p.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159) ignores return value by IERC20(_tokenAddress).approve(address(_router),_amountIn.add(1000000000000000000000)) (c
ontracts/AllRouterSwap.sol#150)
Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188) ignores return value by IERC20(_firstPath).approve(address(_router),_amountIn) (contracts/AllRouterSwap.sol#179)
Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188) ignores return value by _router.swapExactTokensForETH(_amountIn,_minimumAmountOut,_path,msg.sender,block.timestamp +
 180) (contracts/AllRouterSwap.sol#180-186)
Swap._swapToEthInitial(IUniswapV2Router02,address,uint256,address[],uint256) (contracts/AllRouterSwap.sol#190-207) ignores return value by IERC20(_tokenAddress).approve(address(_router),_amountIn.add(1000000000000000000000)) (contracts/Al
lRouterSwap.sol#199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Swap.initialize(address,address,address[],address[])._owner (contracts/AllRouterSwap.sol#64) lacks a zero-check on :
		- owner = _owner (contracts/AllRouterSwap.sol#69)
Swap.initialize(address,address,address[],address[])._admin (contracts/AllRouterSwap.sol#64) lacks a zero-check on :
		- admin = _admin (contracts/AllRouterSwap.sol#70)
Swap.changeOwner(address)._owner (contracts/AllRouterSwap.sol#88) lacks a zero-check on :
		- owner = _owner (contracts/AllRouterSwap.sol#90)
Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._tokenTarget (contracts/AllRouterSwap.sol#312) lacks a zero-check on :
		- inputToken = SwapLibrary._getToken(_token[i],_tokenTarget,_isMultiToSingleToken,true) (contracts/AllRouterSwap.sol#321)
		- outputToken = SwapLibrary._getToken(_token[i],_tokenTarget,_isMultiToSingleToken,false) (contracts/AllRouterSwap.sol#322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (contracts/AllRouterSwap.sol#312-352) has external calls inside a loop: IERC20(inputToken).transferFrom(msg.sender,address(this),_amountIn[i]) (contracts/Al
lRouterSwap.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159) uses timestamp for comparisons
	Dangerous comparisons:
	- assert(bool)(WETH.transfer(msg.sender,_amountIn)) (contracts/AllRouterSwap.sol#125)
	- _fee > 0 (contracts/AllRouterSwap.sol#129)
	- _fee > 0 (contracts/AllRouterSwap.sol#145)
Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(success,errorMessage) (contracts/AllRouterSwap.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used:
	- Version used: ['^0.8.0', '^0.8.7']
	- ^0.8.7 (contracts/AllRouterSwap.sol#2)
	- ^0.8.0 (contracts/library/SwapLibrary.sol#2)
	- ^0.8.0 (contracts/utils/Initializable.sol#3)
	- ^0.8.0 (contracts/utils/SafeMath.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (contracts/AllRouterSwap.sol#312-352) has costly operations inside a loop:
	- inputToken = SwapLibrary._getToken(_token[i],_tokenTarget,_isMultiToSingleToken,true) (contracts/AllRouterSwap.sol#321)
Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (contracts/AllRouterSwap.sol#312-352) has costly operations inside a loop:
	- outputToken = SwapLibrary._getToken(_token[i],_tokenTarget,_isMultiToSingleToken,false) (contracts/AllRouterSwap.sol#322)
Swap.swapToETH(uint256[],address[],uint8[][],uint256[]) (contracts/AllRouterSwap.sol#354-363) has costly operations inside a loop:
	- inputToken = _token[i] (contracts/AllRouterSwap.sol#359)
Swap.swapToETH(uint256[],address[],uint8[][],uint256[]) (contracts/AllRouterSwap.sol#354-363) has costly operations inside a loop:
	- outputToken = addressWETH (contracts/AllRouterSwap.sol#360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
SafeMath.div(uint256,uint256,string) (contracts/utils/SafeMath.sol#190-199) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/utils/SafeMath.sol#150-152) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/utils/SafeMath.sol#216-225) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/utils/SafeMath.sol#167-176) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/utils/SafeMath.sol#21-27) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/utils/SafeMath.sol#63-68) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/utils/SafeMath.sol#75-80) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/utils/SafeMath.sol#46-56) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/utils/SafeMath.sol#34-39) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.7 (contracts/AllRouterSwap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/library/SwapLibrary.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/utils/Initializable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/utils/SafeMath.sol#3) 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
INFO:Detectors:
Low level call in Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159):
	- (success) = owner.call{value: _fee}() (contracts/AllRouterSwap.sol#132)
Low level call in Swap._swapToETH(address,address,IUniswapV2Router02,uint256,address[],uint256,bool) (contracts/AllRouterSwap.sol#161-188):
	- (success) = msg.sender.call{value: _amountIn}() (contracts/AllRouterSwap.sol#167)
Low level call in Swap._returnBalance(bool,uint256) (contracts/AllRouterSwap.sol#299-310):
	- (ethSuccess) = msg.sender.call{value: _amountIn}() (contracts/AllRouterSwap.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Swap.initialize(address,address,address[],address[])._owner (contracts/AllRouterSwap.sol#64) is not in mixedCase
Parameter Swap.initialize(address,address,address[],address[])._admin (contracts/AllRouterSwap.sol#64) is not in mixedCase
Parameter Swap.initialize(address,address,address[],address[])._routerList (contracts/AllRouterSwap.sol#64) is not in mixedCase
Parameter Swap.initialize(address,address,address[],address[])._commonTokens (contracts/AllRouterSwap.sol#64) is not in mixedCase
Parameter Swap.changeOwner(address)._owner (contracts/AllRouterSwap.sol#88) is not in mixedCase
Parameter Swap.addCommonToken(address)._commonToken (contracts/AllRouterSwap.sol#93) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._amountIn (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._token (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._swapRoute (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._isEth (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._tokenTarget (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._isMultiToSingleToken (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._isSingleEth (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._minimumAmountOut (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool)._isNukeTx (contracts/AllRouterSwap.sol#312) is not in mixedCase
Parameter Swap.swapToETH(uint256[],address[],uint8[][],uint256[])._amountIn (contracts/AllRouterSwap.sol#354) is not in mixedCase
Parameter Swap.swapToETH(uint256[],address[],uint8[][],uint256[])._token (contracts/AllRouterSwap.sol#354) is not in mixedCase
Parameter Swap.swapToETH(uint256[],address[],uint8[][],uint256[])._swapRoute (contracts/AllRouterSwap.sol#354) is not in mixedCase
Parameter Swap.swapToETH(uint256[],address[],uint8[][],uint256[])._minimumAmountOut (contracts/AllRouterSwap.sol#354) is not in mixedCase
Constant Swap.addressETH (contracts/AllRouterSwap.sol#44) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Swap.addressWETH (contracts/AllRouterSwap.sol#45) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Swap.WETH (contracts/AllRouterSwap.sol#48) is not in mixedCase
Function IUniswapV2Router02.WETH() (contracts/library/SwapLibrary.sol#28-29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Swap.addCommonToken(address)._commonToken (contracts/AllRouterSwap.sol#93) is too similar to Swap.commonTokens (contracts/AllRouterSwap.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
Swap._swapToken(IUniswapV2Router02,uint256,address[],address,address,bool,uint256,bool) (contracts/AllRouterSwap.sol#103-159) uses literals with too many digits:
        - IERC20(_tokenAddress).approve(address(_router),_amountIn.add(100000000000000000000)) (contracts/AllRouterSwap.sol#150)
Swap._swapToEthInitial(IUniswapV2Router02,address,uint256,address[],uint256) (contracts/AllRouterSwap.sol#190-207) uses literals with too many digits:
        - IERC20(_tokenAddress).approve(address(_router),_amountIn.add(100000000000000000000)) (contracts/AllRouterSwap.sol#199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Swap.addressETH (contracts/AllRouterSwap.sol#44) is never used in Swap (contracts/AllRouterSwap.sol#39-366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
initialize(address,address,address[],address[]) should be declared external:
        - Swap.initialize(address,address,address[],address[]) (contracts/AllRouterSwap.sol#64-77)
changeOwner(address) should be declared external:
        - Swap.changeOwner(address) (contracts/AllRouterSwap.sol#88-91)
addCommonToken(address) should be declared external:
        - Swap.addCommonToken(address) (contracts/AllRouterSwap.sol#93-96)
showOwner() should be declared external:
        - Swap.showOwner() (contracts/AllRouterSwap.sol#98-101)
swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) should be declared external:
        - Swap.swap(uint256[],address[],uint8[][],bool[],address,bool,bool,uint256[],bool) (contracts/AllRouterSwap.sol#312-352)
swapToETH(uint256[],address[],uint8[][],uint256[]) should be declared external:
        - Swap.swapToETH(uint256[],address[],uint8[][],uint256[]) (contracts/AllRouterSwap.sol#354-363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- The external calls inside a loop flagged by Slither in the TokenPrice contract are limited to 5 and 10 as the size of commonTokens array is 5 and the size of routerList is 10 so there is no issue here. On the other hand, searchApprovedTokens function in the contract ApprovedTokens.sol is not limited and was correctly flagged by Slither.
- Some public functions were flagged as they are never called within the same contract, hence they can be declared external to reduce gas costs.
- The flagged reentrancies in AllRouterSwap.sol are false positives although we still recommend using the ReentrancyGuard.
- Unchecked transfers and unused returns were correctly flagged in the AllRouterSwap.sol contract.

AUTOMATED TESTING

5.2 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:

TokenPrice.sol

Report for contracts/TokenPrice.sol
https://dashboard.mythx.io/#/console/analyses/490419b2-81a1-4291-a0c2-77f5b440027a

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

Box.sol

Report for contracts/Box.sol
https://dashboard.mythx.io/#/console/analyses/211deedf-32b6-4343-b9e0-3a4288ca79de

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

ApprovedTokens.sol

Report for contracts/ApprovedTokens.sol
https://dashboard.mythx.io/#/console/analyses/04cfdf79-ab35-43d3-8a7b-729eca473481

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

AllRouterSwap.sol

Report for AllRouterSwap.sol
https://dashboard.mythx.io/#/console/analyses/0f578a13-1ee4-4502-952c-8d6bec18c9e4

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
48	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

- No relevant findinds by MythX.

THANK YOU FOR CHOOSING

HALBORN