Seascape - Block Lords - Import & Export

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 15th, 2022 - August 20th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/15/2022	Roberto Reigada
0.2	Document Updates	08/20/2022	Roberto Reigada
0.3	Document Updates	08/20/2022	Luis Arroyo
0.4	Draft Review	08/20/2022	Gabi Urrutia
0.5	Document Updates	09/06/2022	Luis Arroyo
0.6	Draft Review	09/12/2022	Kubilay Onur Gungor
1.0	Remediation Plan	09/19/2022	Luis Arroyo
1.1	Remediation Plan Review	09/19/2022	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
Kubilay Onur Gungor	Halborn	Kubilay.Gungor@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com
Luis Arroyo	Halborn	Luis.Arroyo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Seascape engaged Halborn to conduct a security audit on their smart contracts beginning on August 15th, 2022 and ending on August 20th, 2022. The security assessment was scoped to the smart contract provided in the GitHub repository blocklords3d/smartcontracts/

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts 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 Seascape 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 code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security 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 contracts

- ImportExportManager.sol
- ImportExportElasticNft.sol

1st Commit ID: f64fa27b972cd6697b8c851b5586b455c165aec6

2nd Commit ID: 5cda6c52f94583c4d44d84e1f36770f30f984246

EXECUTIVE OVERVIEW

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	3	0

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨	🟨 (HAL-01)	🟥	🟥
🟩 (HAL-02) (HAL-03)	🟩	🟨	🟨	🟥
⬜	🟩 (HAL-04)	🟩	🟨	🟨
⬜	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - UNUSABLE CONTRACT	Medium	SOLVED - 09/19/2022
HAL02 - UNDEFINED VARIABLES ARE USED	Low	SOLVED - 09/19/2022
HAL03 - UNDEFINED IMPORTS ARE USED	Low	SOLVED - 09/19/2022
HAL04 - MISSING ZERO ADDRESS CHECKS	Low	SOLVED - 09/19/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNUSABLE CONTRACT - MEDIUM

Description:

The ImportExportElasticNft.sol smart contract constructor initializes the nft variable to the same value. This cause nft value to be equal to 0x0000000000000000000000000000000000000000. Once deployed, this means that the contract cannot be used since there is no way to modify this variable.

Code Location:

Listing 1: ImportExportElasticNft.sol (Line 19)

```
18    constructor(address nft) SecureContract(true, true) {
19        nft = _nft;
20        owner = msg.sender;
21        verifier = msg.sender;
22    }
```

Proof of Concept:

To replicate this issue:
- Deploy the smart contract with any address as NFT.
- check the NFT address of the contract.

Listing 2: pentest.js

```
elastic = await (await (await hre.ethers.getContractFactory("ImportExportElasticNft"))
            .deploy(block.address)).deployed();

console.log('[+] nft addr: ' + await elastic.nft());
```

Listing 3: Output

```
  Elastic Testing
[+] nft add: 0x0000000000000000000000000000000000000000
```

Risk Level:

Likelihood - 3
Impact - 4

Recommendation:

It is recommended to use the parameters of the constructor correctly to set the correct addresses when deploying the smart contract and avoid mentioned scenarios.

Remediation Plan:

SOLVED: The SeaScape team now assigns correctly the _nft parameter to the nft state variable.

3.2 (HAL-02) UNDEFINED VARIABLES ARE USED - LOW

Description:

The ImportExportElasticNft.sol smart contract uses undefined variables, resulting in contracts which do not compile.

Code Location:

- nftExportNonce (ImportExportElasticNft.sol#59,65)

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to declare all used variables.

Remediation Plan:

SOLVED: The SeaScape team has implemented and declared the mapping nftExportNonce.

3.3 (HAL-03) UNDEFINED IMPORTS ARE USED - LOW

Description:

The ImportExportElasticNft.sol smart contract uses undefined castings referring to other contracts. The name of these contracts is not correct.

Code Location:

- Blocklords (ImportExportElasticNft.sol#46,47,67)

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to define all castings correctly to other contracts.

Remediation Plan:

SOLVED: The SeaScape team has renamed the Blocklord contract to BlockLords to correct this import.

3.4 (HAL-04) MISSING ZERO ADDRESS CHECKS - LOW

Description:

It has been detected that some functions, such as exportNft(), and exportToken() of the smart contracts, are missing address validation. Every input address should be checked not to be zero, especially the ones that could lead to rendering the contract unusable, lock tokens, etc. This is considered a best practice.

Code Location:

Listing 4: ImportExportManager.sol (Lines 108,133)

```
108     function exportNft(address nft, uint nftId, uint8 _v, bytes32
 ↳ _r, bytes32 _s) external {
109         require(supportedNfts[nft], "unsupported token");
110
111         /// Validation of quality
112         /// message is generated as owner + amount + last time
 ↳ stamp + quality
113         bytes memory _prefix = "\x19Ethereum Signed Message:\n32";
114         bytes32 _messageNoPrefix =
115         keccak256(abi.encodePacked(msg.sender, nft, address(this),
 ↳  block.chainid, nftId, nftExportNonce[msg.sender]));
116         bytes32 _message = keccak256(abi.encodePacked(_prefix,
 ↳ _messageNoPrefix));
117         address _recover = ecrecover(_message, _v, _r, _s);
118
119         require(_recover == verifier, "verification failed");
120
121         nftExportNonce[msg.sender]++;
122
123         address accountHodler = accountHodlerOf(msg.sender);
124
125         if (address(accountHodler).codehash == 0) {
126             require(deploy(accountHodler, msg.sender), "Failed to
 ↳ deploy the contract");
127             AccountHodler(accountHodler).initialize(owner);
```

```
        }

        AccountHodler(accountHodler).exportNft(nft, msg.sender, nftId);
    }

    function exportToken(address token, uint amount, uint fee, uint8 _v, bytes32 _r, bytes32 _s) external {
        require(supportedTokens[token], "unsupported token");

        /// Validation of quality
        /// message is generated as owner + amount + last time stamp + quality
        bytes memory _prefix = "\x19Ethereum Signed Message:\n32";
        bytes32 _messageNoPrefix =
        keccak256(abi.encodePacked(msg.sender, token, address(this), block.chainid, amount, fee, tokenExportNonce[msg.sender]));
        bytes32 _message = keccak256(abi.encodePacked(_prefix, _messageNoPrefix));
        address _recover = ecrecover(_message, _v, _r, _s);

        require(_recover == verifier, "verification failed");

        tokenExportNonce[msg.sender]++;

        address accountHodler = accountHodlerOf(msg.sender);

        if (address(accountHodler).codehash == 0) {
            require(deploy(accountHodler, msg.sender), "Failed to deploy the contract");
            AccountHodler(accountHodler).initialize(owner);
        }

        AccountHodler(accountHodler).exportToken(token,msg.sender, feeReceiver, amount, fee);
    }
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

It is recommended to validate that each address inputs in the exportNft(), exportToken(), supportNft(), supportToken() and similar functions from the smart contracts are non-zero.

Remediation Plan:

SOLVED: The SeaScape team now validates the inputs where an address is used to verify they are non-zero before performing any functionality.

AUTOMATED TESTING

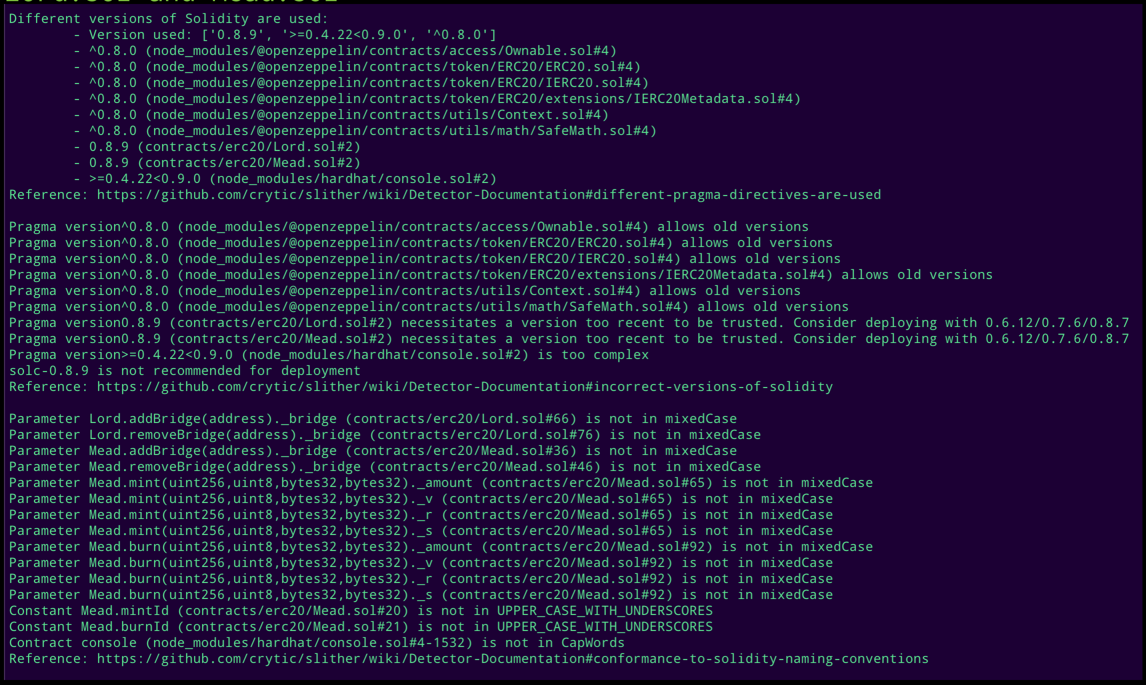
4.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

Lord.sol and Mead.sol

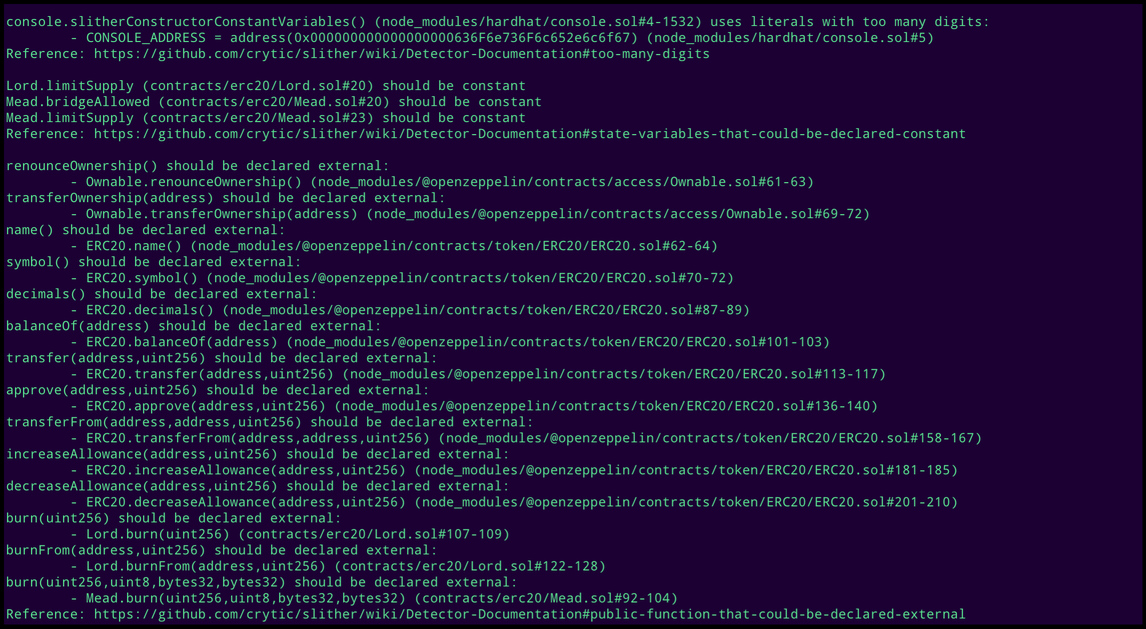


```
Different versions of Solidity are used:
        - Version used: ['0.8.9', '>=0.4.22<0.9.0', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
        - 0.8.9 (contracts/erc20/Lord.sol#2)
        - 0.8.9 (contracts/erc20/Mead.sol#2)
        - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) allows old versions
Pragma version0.8.9 (contracts/erc20/Lord.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/erc20/Mead.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Lord.addBridge(address)._bridge (contracts/erc20/Lord.sol#66) is not in mixedCase
Parameter Lord.removeBridge(address)._bridge (contracts/erc20/Lord.sol#76) is not in mixedCase
Parameter Mead.addBridge(address)._bridge (contracts/erc20/Mead.sol#36) is not in mixedCase
Parameter Mead.removeBridge(address)._bridge (contracts/erc20/Mead.sol#46) is not in mixedCase
Parameter Mead.mint(uint256,uint8,bytes32,bytes32)._amount (contracts/erc20/Mead.sol#65) is not in mixedCase
Parameter Mead.mint(uint256,uint8,bytes32,bytes32)._v (contracts/erc20/Mead.sol#65) is not in mixedCase
Parameter Mead.mint(uint256,uint8,bytes32,bytes32)._r (contracts/erc20/Mead.sol#65) is not in mixedCase
Parameter Mead.mint(uint256,uint8,bytes32,bytes32)._s (contracts/erc20/Mead.sol#65) is not in mixedCase
Parameter Mead.burn(uint256,uint8,bytes32,bytes32)._amount (contracts/erc20/Mead.sol#92) is not in mixedCase
Parameter Mead.burn(uint256,uint8,bytes32,bytes32)._v (contracts/erc20/Mead.sol#92) is not in mixedCase
Parameter Mead.burn(uint256,uint8,bytes32,bytes32)._r (contracts/erc20/Mead.sol#92) is not in mixedCase
Parameter Mead.burn(uint256,uint8,bytes32,bytes32)._s (contracts/erc20/Mead.sol#92) is not in mixedCase
Constant Mead.mintId (contracts/erc20/Mead.sol#20) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Mead.burnId (contracts/erc20/Mead.sol#21) is not in UPPER_CASE_WITH_UNDERSCORES
Contract console (node_modules/hardhat/console.sol#4-1532) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

AUTOMATED TESTING



```
console.slitherConstructorConstantVariables() (node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (node_modules/hardhat/console.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Lord.limitSupply (contracts/erc20/Lord.sol#20) should be constant
Mead.bridgeAllowed (contracts/erc20/Mead.sol#20) should be constant
Mead.limitSupply (contracts/erc20/Mead.sol#23) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-63)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#69-72)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
burn(uint256) should be declared external:
        - Lord.burn(uint256) (contracts/erc20/Lord.sol#107-109)
burnFrom(address,uint256) should be declared external:
        - Lord.burnFrom(address,uint256) (contracts/erc20/Lord.sol#122-128)
burn(uint256,uint8,bytes32,bytes32) should be declared external:
        - Mead.burn(uint256,uint8,bytes32,bytes32) (contracts/erc20/Mead.sol#92-104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.

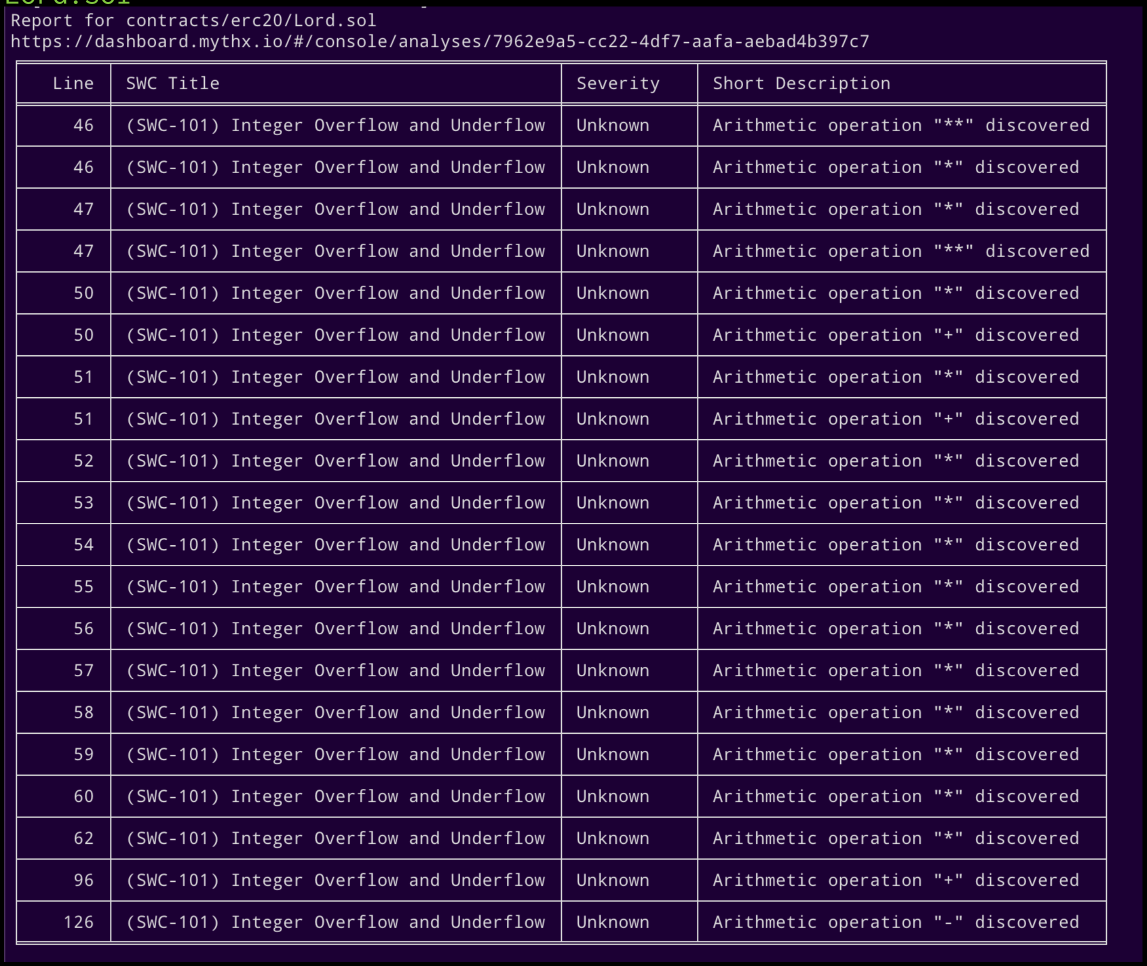
4.2 AUTOMATED SECURITY SCAN

Description:

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

MythX results:

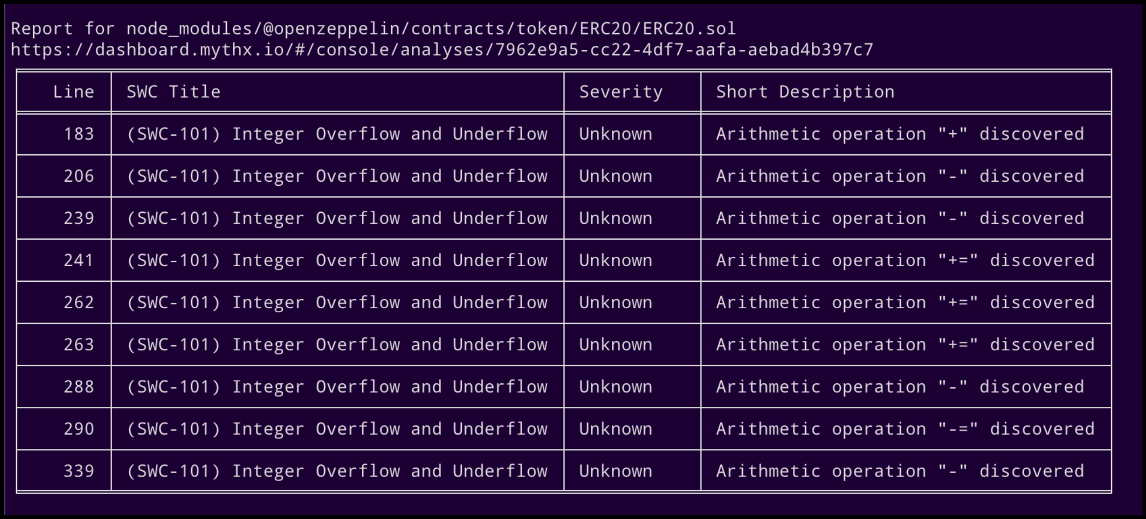
Lord.sol



```
Report for contracts/erc20/Lord.sol
https://dashboard.mythx.io/#/console/analyses/7962e9a5-cc22-4df7-aafa-aebad4b397c7

| Line | SWC Title                                 | Severity | Short Description                   |
|------|-------------------------------------------|----------|-------------------------------------|
|   46 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "**" discovered |
|   46 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   47 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   47 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "**" discovered |
|   50 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   50 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "+" discovered  |
|   51 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   51 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "+" discovered  |
|   52 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   53 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   54 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   55 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   56 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   57 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   58 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   59 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   60 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   62 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "*" discovered  |
|   96 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "+" discovered  |
|  126 | (SWC-101) Integer Overflow and Underflow  | Unknown  | Arithmetic operation "-" discovered  |
```

Mead.sol



```
Report for node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
https://dashboard.mythx.io/#/console/analyses/7962e9a5-cc22-4df7-aafa-aebad4b397c7
╒════════╤════════════════════════════════════════╤════════════╤═════════════════════════════════════╕
│   Line │ SWC Title                              │ Severity   │ Short Description                   │
╞════════╪════════════════════════════════════════╪════════════╪═════════════════════════════════════╡
│    183 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "+" discovered  │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    206 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "-" discovered  │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    239 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "-" discovered  │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    241 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "+=" discovered │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    262 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "+=" discovered │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    263 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "+=" discovered │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    288 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "-" discovered  │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    290 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "-=" discovered │
├────────┼────────────────────────────────────────┼────────────┼─────────────────────────────────────┤
│    339 │ (SWC-101) Integer Overflow and Underflow │ Unknown    │ Arithmetic operation "-" discovered  │
╘════════╧════════════════════════════════════════╧════════════╧═════════════════════════════════════╛
```

- No major issues found by MythX.

THANK YOU FOR CHOOSING

HALBORN