HALBORN

Pangolin - Exchange Contracts

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 28th, 2022 - March 1st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/28/2022	Roberto Reigada
0.2	Document Updates	03/01/2022	Roberto Reigada
0.3	Draft Review	03/01/2022	Gabi Urrutia
1.0	Remediation Plan	03/09/2022	Roberto Reigada
1.1	Remediation Plan Review	03/09/2022	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
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Pangolin engaged Halborn to conduct a security audit on their Exchange smart contracts beginning on February 28th, 2022 and ending on March 1st, 2022. The security assessment was scoped to some of the smart contracts provided in the GitHub repository pangolindex/exchange-contracts.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned a full-time security engineer to audit the security of the smart contracts. 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 mostly addressed by the Pangolin 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. 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.

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

1.4 SCOPE

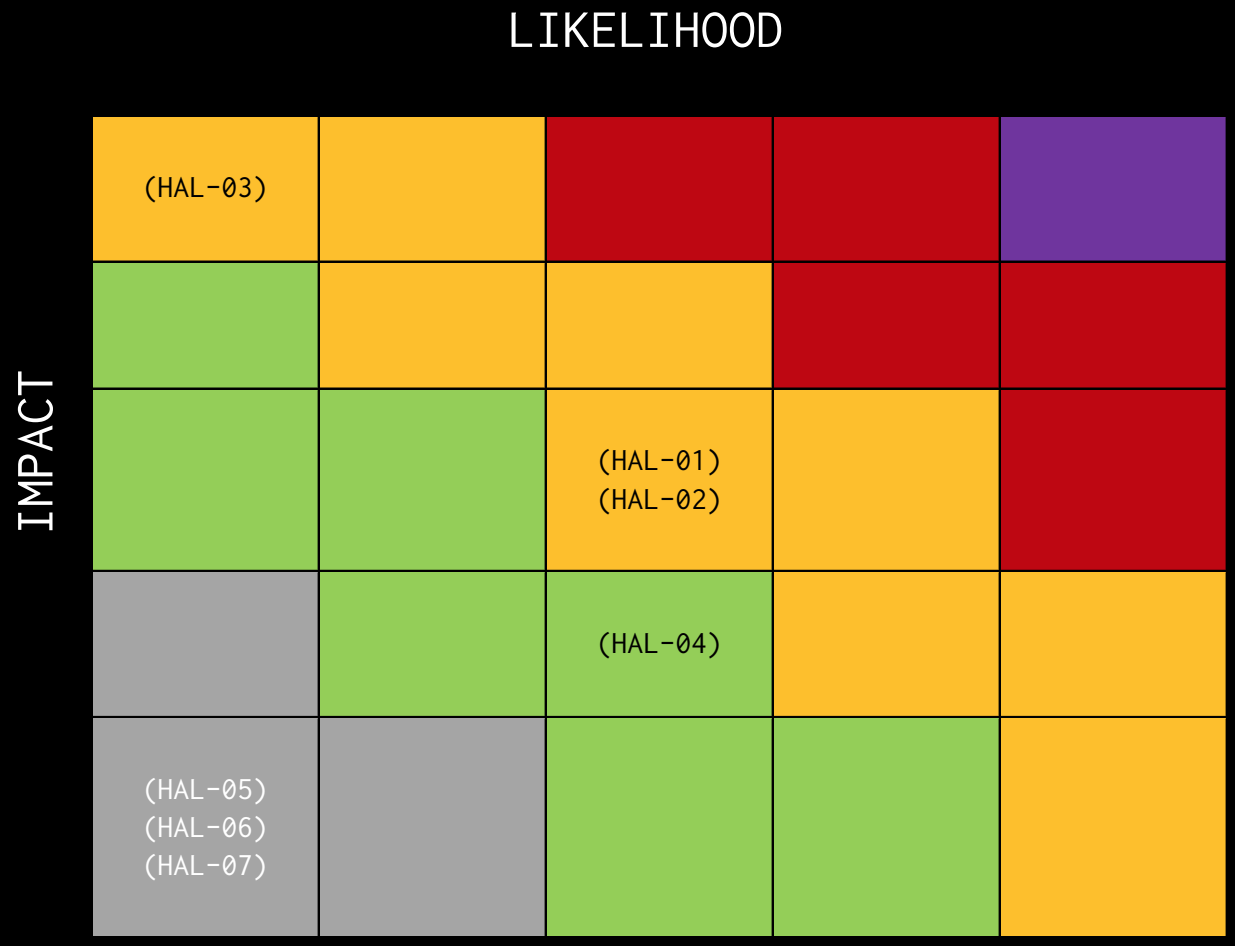
IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- RevenueDistributor.sol
- TreasuryVester.sol
- PNG.sol
- Airdrop.sol

Commit ID: 30b5798c64f6aa29b92a5c2cf68cfb7ed0f7d506
Fixed Commit ID: bbbf14abf0283fa7ea3ccf07288fecdc177ed8f9

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	3	1	3



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - MINT FUNCTION IS MISSING MOVEDELEGATES CALL	Medium	SOLVED - 03/09/2022
HAL02 - LACK OF TRANSFEROWNERSHIP PATTERN	Medium	RISK ACCEPTED
HAL03 - DOS WITH BLOCK GAS LIMIT	Medium	SOLVED - 03/09/2022
HAL04 - MISSING ZERO ADDRESS CHECKS	Low	SOLVED - 03/09/2022
HAL05 - USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	SOLVED - 03/09/2022
HAL06 - UNNEEDED INITIALIZATION OF UINT VARIABLES TO ZERO	Informational	SOLVED - 03/09/2022
HAL07 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 03/09/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) MINT FUNCTION IS MISSING MOVEDELEGATES CALL - MEDIUM

Description:

In the Png contract, the function mint() does not call the _moveDelegates () function:

Listing 1: PNG.sol

```
184 function mint(address dst, uint rawAmount) external returns (bool)
    ↳ {
185     require(msg.sender == minter && minter != address(0), "Png::
        ↳ mint: unauthorized");
186     uint96 amount = safe96(rawAmount, "Png::mint: amount exceeds
        ↳ 96 bits");
187     _mintTokens(dst, amount);
188     return true;
189 }
```

Listing 2: PNG.sol

```
421 function _mintTokens(address dst, uint96 amount) internal {
422     require(dst != address(0), "Png::_mintTokens: cannot mint to
        ↳ the zero address");
423
424     totalSupply = SafeMath.add(totalSupply, uint(amount));
425     balances[dst] = add96(balances[dst], amount, "Png::_mintTokens
        ↳ : mint amount overflows");
426     emit Transfer(address(0), dst, amount);
427
428     require(totalSupply <= maxSupply, "Png::_mintTokens: mint
        ↳ result exceeds max supply");
429 }
```

This causes that every time Png tokens are minted the users will have to manually call delegate() passing their own address as parameter so their voting power is correctly accounted/updated in the smart contract:

```
Calling -> contract_PNG.mint(user1, 10_000000000000000000, {'from': owner})
Transaction sent: 0xdc53c5e669e119b3ec77f856795c4ff29d9c0584cbf947b68ca48f63ab9ec396
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  Png.mint confirmed   Block: 14294259   Gas used: 55905 (0.01%)

contract_PNG.balanceOf(user1) -> 10000000000000000000
contract_PNG.delegates(user1) -> 0x0000000000000000000000000000000000000000
contract_PNG.getCurrentVotes(user1) -> 0
Calling -> contract_PNG.delegate(user1.address, {'from': user1})
Transaction sent: 0x35cc7ecfa641e10966d85d055e94cadc4555ae2a6f4fd1d3a4f88e983136eb90
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  Png.delegate confirmed   Block: 14294261   Gas used: 91384 (0.02%)

contract_PNG.balanceOf(user1) -> 10000000000000000000
contract_PNG.delegates(user1) -> 0x2e27A9A350A318Cfd233968513c89f5Cf505587B
contract_PNG.getCurrentVotes(user1) -> 10000000000000000000
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to add the _moveDelegates() function call into the _mintTokens() function.

Remediation Plan:

SOLVED: Pangolin team solved the issue in the commit id bbbf14abf0283fa7ea3ccf07288fecdc177ed8f9.

3.2 (HAL-02) LACK OF TRANSFEROWNERSHIP PATTERN - MEDIUM

Description:

All functions that involve some kind of transfer of ownership require a single step, which is to set up the new privileged address. If this designated EOA account is not a valid account, it is very possible that the transfer of ownership will be made to an uncontrolled account, losing access to privileged functions.

Code location:

PNG.sol

Listing 3: PNG.sol (Line 267)

```
264 function setMinter(address newMinter) external returns (bool) {
265     require(msg.sender == admin, "Png::setMinter: unauthorized");
266     emit MinterChanged(minter, newMinter);
267     minter = newMinter;
268     return true;
269 }
```

Listing 4: PNG.sol (Line 280)

```
276 function setAdmin(address newAdmin) external returns (bool) {
277     require(msg.sender == admin, "Png::setAdmin: unauthorized");
278     require(newAdmin != address(0), "Png::setAdmin: cannot make
        ↳ zero address the admin");
279     emit AdminChanged(admin, newAdmin);
280     admin = newAdmin;
281     return true;
282 }
```

Airdrop.sol

Listing 5: Airdrop.sol (Line 73)

```
70 function setOwner(address owner_) external {
71     require(owner_ != address(0), 'Airdrop::setOwner: invalid new
       ↳ owner');
72     require(msg.sender == owner, 'Airdrop::setOwner: unauthorized'
       ↳ );
73     owner = owner_;
74 }
```

Listing 6: Airdrop.sol (Line 81)

```
79     function setWhitelister(address addr) external {
80         require(msg.sender == owner, 'Airdrop::setWhitelister:
           ↳ unauthorized');
81         whitelister = addr;
82     }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account must call an acceptOwnership() function for the transfer of ownership to succeed. This ensures the nominated EOA account is a valid and active account.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk:
- Airdrop events will last for a short time, and it is expected that ownership will only be transferred once from the deployer to the multisig.
- PNG Ownership will only be transferred from the deployer to the Timelock

contract (contracts/governance/Timelock.sol). This Timelock contract has the suggested two-step ownership transfer pattern, and any additional change of ownership would likely go through Timelock rather than PNG.

3.3 (HAL-03) DOS WITH BLOCK GAS LIMIT - MEDIUM

Description:

In the contract TreasuryVester the distribute() function is used to distribute the tokens to recipients based on their allocation:

TreasuryVester.sol

Listing 7: TreasuryVester.sol (Line 149)

```
70 function distribute() public {
71     require(vestingEnabled, "TreasuryVester::distribute: vesting
       ↳ is not enabled");
72     require(
73         block.timestamp >= lastUpdate + VESTING_CLIFF,
74         "TreasuryVester::distribute: it is too early to distribute
           ↳ "
75     );
76     lastUpdate = block.timestamp;
77
78     // defines a vesting schedule that lasts for 30 months
79     if (step % STEPS_TO_SLASH == 0) {
80         uint slash = step / STEPS_TO_SLASH;
81         if (slash < 5) {
82             _vestingPercentage = _initialVestingPercentages[slash
                ↳ ];
83         } else if (slash < 12) {
84             _vestingPercentage -= 20;
85         } else if (slash < 20) {
86             _vestingPercentage -= 15;
87         } else if (slash < 30) {
88             _vestingPercentage -= 10;
89         } else {
90            revert("TreasuryVester::distribute: vesting is over");
91         }
92         _vestingAmount = getVestingAmount();
93     }
94     step++;
95
96     // distributes _vestingAmount of tokens to recipients based on
```

```
        ↳  their allocation
 97     for (uint i; i < _recipientsLength; i++) {
 98         Recipient memory recipient = _recipients[i];
 99         uint amount = recipient.allocation * _vestingAmount /
            ↳ DENOMINATOR;
100         if (!recipient.isMiniChef) {
101             // simply mints or transfer tokens to regular
                ↳ recipients
102             vestedToken.mint(recipient.account, amount);
103         } else {
104             // calls fund rewards of minichef after minting tokens
                ↳  to self
105             vestedToken.mint(address(this), amount);
106             vestedToken.approve(recipient.account, amount);
107             IMiniChefV2(recipient.account).fundRewards(amount,
                ↳ VESTING_CLIFF);
108         }
109     }
110     emit TokensVested(_vestingAmount);
111 }
```

As the length of recipients is not limited, in case that there are too many recipients, the block gas limit could be reached, causing miners to not respond to all distribute() calls, thus blocking the main purpose of the smart contract.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to add a require statement in the setRecipients() ↳ function of the TreasuryVester contract that limits the number of recipients.

Remediation Plan:

SOLVED: Pangolin team solved the issue in the commit id 1e2f374c6728b998a22045a673be0ba14156b9c1.

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

Description:

In the TreasuryVester and Airdrop contracts, the constructors are missing address validation. On the other hand, in some functions it is also critical to perform this validation. For example, in the TreasuryVester ↳ .setRecipients() function, if a 0 address recipient was set the distribute() call would always fail, since most tokens cannot be minted at the 0 address.

Each address should be validated and checked to be non-zero.

Code location:

TreasuryVester.sol

Listing 8: TreasuryVester.sol (Line 87)

```
78 constructor(
79     address newVestedToken,
80     uint newStartingBalance,
81     Recipient[] memory newRecipients,
82     address newGuardian
83 ) {
84     require(newStartingBalance > 0, "TreasuryVester::Constructor:
       ↳ invalid starting balance");
85     require(newGuardian != address(0), "TreasuryVester::
       ↳ Constructor: invalid guardian address");
86     guardian = newGuardian;
87     vestedToken = IPng(newVestedToken);
88     startingBalance = newStartingBalance;
89     setRecipients(newRecipients);
90 }
```

Listing 9: TreasuryVester.sol (Line 187)

```
170 function setRecipients(Recipient[] memory newRecipients) public
    ↳ onlyOwner {
171     _recipientsLength = newRecipients.length;
172     require(
173         _recipientsLength != 0,
174         "TreasuryVester::setRecipients: invalid recipient number"
175     );
176     uint allocations;
177     for (uint i; i < _recipientsLength; ++i) {
178         Recipient memory recipient = newRecipients[i];
179         require(
180             recipient.account != address(0),
181             "TreasuryVester::setRecipients: invalid recipient
                ↳ address"
182         );
183         require(
184             recipient.allocation != 0,
185             "TreasuryVester::setRecipients: invalid recipient
                ↳ allocation"
186         );
187         _recipients[i] = recipient;
188         allocations += recipient.allocation;
189     }
190     require(
191         allocations == DENOMINATOR,
192         "TreasuryVester::setRecipients: invalid total allocation"
193     );
194     emit RecipientsChanged(newRecipients);
195 }
```

Airdrop.sol

Listing 10: Airdrop.sol (Lines 45-47)

```
38 constructor(
39     uint supply_,
40     address png_,
41     address owner_,
42     address remainderDestination_
43 ) {
44     airdropSupply = supply_;
45     png = png_;
```

```
46      owner = owner_;
47      remainderDestination = remainderDestination_;
48  }
```

Listing 11: Airdrop.sol (Line 62)

```
57  function setRemainderDestination(address remainderDestination_)
    ↳ external {
58      require(
59          msg.sender == owner,
60          'Airdrop::setRemainderDestination: unauthorized'
61      );
62      remainderDestination = remainderDestination_;
63  }
```

PNG.sol

Listing 12: PNG.sol (Line 267)

```
264 function setMinter(address newMinter) external returns (bool) {
265     require(msg.sender == admin, "Png::setMinter: unauthorized");
266     emit MinterChanged(minter, newMinter);
267     minter = newMinter;
268     return true;
269 }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to validate that each address input is non-zero.

Remediation Plan:

SOLVED: Pangolin team solved the issue in the commit id 8810acc38f27bf9de25a492ffe41d3c54c657c5f.

3.5 (HAL-05) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

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

Code Location:

TreasuryVester.sol
- Line 149: `for (uint i; i < _recipientsLength; i++){`

Airdrop.sol
- Line 178: `for (uint i = 0; i < addrs.length; i++){`

Proof of Concept:

For example, based in the following test contract:

Listing 13: Test.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.9;
3
4 contract test {
5     function postiincrement(uint256 iterations) public {
6         for (uint256 i = 0; i < iterations; i++) {
7         }
8     }
9     function preiincrement(uint256 iterations) public {
10        for (uint256 i = 0; i < iterations; ++i) {
11        }
12    }
13 }
```

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: 0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05'>
```

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: Pangolin team solved the issue in the commit id 4aa439b03eac34bebf40c6b72e9afeb2d0fa2333.

3.6 (HAL-06) UNNEEDED INITIALIZATION OF UINT VARIABLES TO ZERO - INFORMATIONAL

Description:

Since `i` is an `uint`, it is already initialized to `0`. `uint i = 0` reassigns the `0` to `i` which wastes gas. The same applies to the `lower` state variable shown below.

Code Location:

Airdrop.sol
- Line 178: `for (uint i = 0; i < addrs.length; i++){`

PNG.sol
- Line 373: `uint32 lower = 0;`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended not to initialize uint variables to `0` to save gas. For example:
`for (uint i; i < addrs.length; ++i){`

Remediation Plan:

SOLVED: Pangolin team solved the issue in the commit id `87781621d7294afeafd7a33916b4016dc1f3ed34`.

3.7 (HAL-07) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the TreasuryVester and PNG contracts, there are some functions marked as public that are never called directly within the contract itself or in any of their descendants:

TreasuryVester.sol
- distribute() (TreasuryVester.sol#122-163)

PNG.sol
- delegate() (PNG.sol#314-316)
- delegateBySig() (PNG.sol#327-336)
- getPriorVotes() (PNG.sol#355-387)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by their descendants, it is better to mark them as external to reduce gas costs.

Remediation Plan:

SOLVED: Pangolin team solved the issue in the commit id f1fff4b75db0fe70450c55234418bd445834029d.

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:

RevenueDistributor.sol

```
RevenueDistributor.setRecipients(RevenueDistributor.Recipient[]).i (contracts/RevenueDistributor.sol#58) is a local variable never initialized
RevenueDistributor.distributeToken(address).i (contracts/RevenueDistributor.sol#37) is a local variable never initialized
RevenueDistributor.getRecipients().i (contracts/RevenueDistributor.sol#27) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in RevenueDistributor.distributeToken(address) (contracts/RevenueDistributor.sol#33-49):
        External calls:
        - IERC20(token).safeTransfer(_recipients[_recipientsLength - 1].account,IERC20(token).balanceOf(address(this))) (contracts/RevenueDistributor.sol#44-47)
        Event emitted after the call(s):
        - TokenDistributed(token,amount) (contracts/RevenueDistributor.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#29-36) 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/access/Ownable.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/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.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 (contracts/RevenueDistributor.sol#2) allows old versions
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#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TreasuryVester.sol

```
TreasuryVester.setRecipients(TreasuryVester.Recipient[]).i (contracts/TreasuryVester.sol#177) is a local variable never initialized
TreasuryVester.getRecipients().i (contracts/TreasuryVester.sol#99) is a local variable never initialized
TreasuryVester.distribute().i (contracts/TreasuryVester.sol#149) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) ignores return value by vestedToken.mint(recipient.account,amount) (contracts/TreasuryVester.sol#154)
TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) ignores return value by vestedToken.mint(address(this),amount) (contracts/TreasuryVester.sol#157)
TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) ignores return value by vestedToken.approve(recipient.account,amount) (contracts/TreasuryVester.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) has external calls inside a loop: vestedToken.mint(recipient.account,amount) (contracts/TreasuryVester.sol#154)
TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) has external calls inside a loop: vestedToken.mint(address(this),amount) (contracts/TreasuryVester.sol#157)
TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) has external calls inside a loop: vestedToken.approve(recipient.account,amount) (contracts/TreasuryVester.sol#158)
TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) has external calls inside a loop: IMiniChefV2(recipient.account).fundRewards(amount,VESTING_CLIFF) (contracts/TreasuryVester.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163):
        External calls:
        - vestedToken.mint(recipient.account,amount) (contracts/TreasuryVester.sol#154)
        - vestedToken.mint(address(this),amount) (contracts/TreasuryVester.sol#157)
```

```
        - vestedToken.approve(recipient.account,amount) (contracts/TreasuryVester.sol#158)
        - IMiniChefV2(recipient.account).fundRewards(amount,VESTING_CLIFF) (contracts/TreasuryVester.sol#159)
        Event emitted after the call(s):
        - TokensVested(_vestingAmount) (contracts/TreasuryVester.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastUpdate + VESTING_CLIFF,TreasuryVester::distribute: it is too early to distribute) (contracts/TreasuryVester.sol#124-127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#90-96) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#88-98) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#21-27) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#29-36) 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/access/Ownable.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/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.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 (contracts/TreasuryVester.sol#3) allows old versions
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#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
distribute() should be declared external:
        - TreasuryVester.distribute() (contracts/TreasuryVester.sol#122-163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Airdrop.sol

```
Airdrop.constructor(uint256,address,address,address).png_ (contracts/Airdrop.sol#40) lacks a zero-check on :
                - png = png_ (contracts/Airdrop.sol#45)
Airdrop.constructor(uint256,address,address,address).owner_ (contracts/Airdrop.sol#41) lacks a zero-check on :
                - owner = owner_ (contracts/Airdrop.sol#46)
Airdrop.constructor(uint256,address,address,address).remainderDestination_ (contracts/Airdrop.sol#42) lacks a zero-check on :
                - remainderDestination = remainderDestination_ (contracts/Airdrop.sol#47)
Airdrop.setRemainderDestination(address).remainderDestination_ (contracts/Airdrop.sol#57) lacks a zero-check on :
                - remainderDestination = remainderDestination_ (contracts/Airdrop.sol#62)
Airdrop.setWhitelister(address).addr (contracts/Airdrop.sol#79) lacks a zero-check on :
                - whitelister = addr (contracts/Airdrop.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Airdrop.claim() (contracts/Airdrop.sol#139-155):
        External calls:
        - require(bool,string)(IPNG(png).transfer(msg.sender,amountToClaim),Airdrop::claim: Transfer failed) (contracts/Airdrop.sol#149-152)
        Event emitted after the call(s):
        - PngClaimed(msg.sender,amountToClaim) (contracts/Airdrop.sol#154)
Reentrancy in Airdrop.endClaiming() (contracts/Airdrop.sol#117-131):
        External calls:
        - require(bool,string)(IPNG(png).transfer(remainderDestination,amount),Airdrop::endClaiming: Transfer failed) (contracts/Airdrop.sol#125-128)
        Event emitted after the call(s):
        - ClaimingOver() (contracts/Airdrop.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Airdrop.whitelistAddresses(address[],uint256[]) (contracts/Airdrop.sol#162-188) has costly operations inside a loop:
        - totalAllocated = totalAllocated + pngOut - withdrawAmount[addr] (contracts/Airdrop.sol#181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Pragma version^0.8.0 (contracts/Airdrop.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

PNG.sol

```
Png._writeCheckpoint(address,uint32,uint96,uint96) (contracts/PNG.sol#449-460) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/PNG.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Png.setMinter(address).newMinter (contracts/PNG.sol#264) lacks a zero-check on :
                - minter = newMinter (contracts/PNG.sol#267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Png.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#148-167) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= deadline,Png::permit: signature expired) (contracts/PNG.sol#162)
Png.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#327-336) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,Png::delegateBySig: signature expired) (contracts/PNG.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Png.getChainId() (contracts/PNG.sol#483-487) uses assembly
        - INLINE ASM (contracts/PNG.sol#485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

SafeMath.add(uint256,uint256,string) (contracts/libraries/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/libraries/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/libraries/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/libraries/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libraries/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/libraries/SafeMath.sol#85-97) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/libraries/SafeMath.sol#107-119) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

delegate(address) should be declared external:
        - Png.delegate(address) (contracts/PNG.sol#314-316)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - Png.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#327-336)
getPriorVotes(address,uint256) should be declared external:
        - Png.getPriorVotes(address,uint256) (contracts/PNG.sol#355-387)
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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

RevenueDistributor.sol

Report for contracts/RevenueDistributor.sol
https://dashboard.mythx.io/#/console/analyses/fce0ef23-02d0-42e9-974e-142923d60c2b

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
27	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
28	(SWC-110) Assert Violation	Unknown	Out of bounds array access
37	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
37	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
37	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
40	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
40	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
45	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
45	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
58	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
59	(SWC-110) Assert Violation	Unknown	Out of bounds array access
66	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

TreasuryVester.sol

Report for contracts/TreasuryVester.sol
https://dashboard.mythx.io/#/console/analyses/81a0a8a1-4592-4686-988c-6ce686715dec

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
99	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
100	(SWC-110) Assert Violation	Unknown	Out of bounds array access
112	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
112	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
125	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
131	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
132	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
134	(SWC-110) Assert Violation	Unknown	Out of bounds array access

136	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
138	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
140	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
146	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
149	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
151	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
151	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
177	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
178	(SWC-110) Assert Violation	Unknown	Out of bounds array access
188	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

Airdrop.sol

Report for contracts/Airdrop.sol
https://dashboard.mythx.io/#/console/analyses/912e3b2e-3bc1-4a88-be0a-fa2f9d0d0033

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

PNG.sol

Report for PNG.sol
https://dashboard.mythx.io/#/console/analyses/d19141d8-1adb-42d4-b220-0d8239bec541

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
356	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
450	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

- `block.number` is not used as a source of randomness.
- The pragmas are set in the `hardhat.config.js` file.
- Integer Overflows and Underflows flagged by MythX are false positives, as the contracts are using Solidity `^0.8.0` version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- Assert violations are false positives.

THANK YOU FOR CHOOSING

// HALBORN