Findings

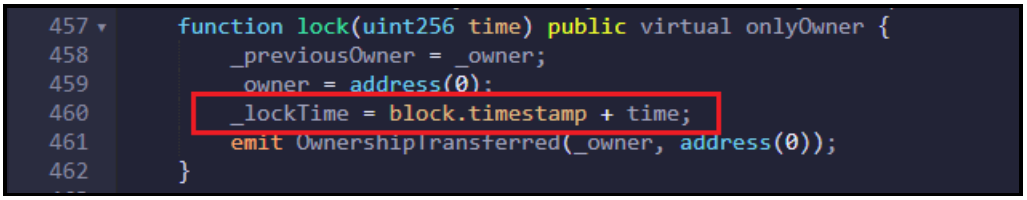
1. **Integer Arithmetic Bugs**

    - SWC ID: 101
    - Severity: High
    - Estimated Gas Usage: 1169 – 1545
    - Relationships: CWE-682: Incorrect Calculation
    - Description: The arithmetic operator can overflow. It is possible to cause an integer overflow or underflow in the arithmetic operation.

```
720     function calculateTopUpClaim(
721         uint256 currentRecipientBalance,
722         uint256 basedRewardCycleBlock,
723         uint256 threshHoldTopUpRate,
724         uint256 amount
725     ) public view returns (uint256) {
726         if (currentRecipientBalance == 0) {
727             return block.timestamp + basedRewardCycleBlock;
728         }
729         else {
730             uint256 rate = amount.mul(100).div(currentRecipientBalance);
731 
732             if (uint256(rate) >= threshHoldTopUpRate) {
733                 uint256 incurCycleBlock = basedRewardCycleBlock.mul(uint256(rate)).div(100);
734 
735                 if (incurCycleBlock >= basedRewardCycleBlock) {
736                     incurCycleBlock = basedRewardCycleBlock;
737                 }
738 
739                 return incurCycleBlock;
740             }
741 
742             return 0;
743         }
744     }
745 
```

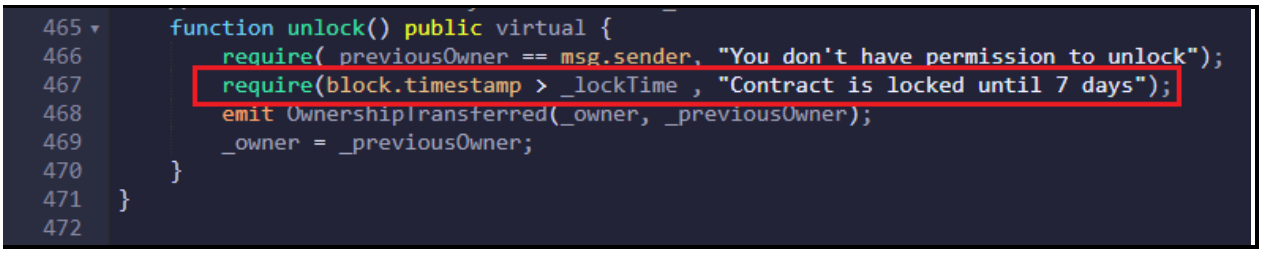
2. Integer Arithmetic Bugs

- SWC ID: 101
- Severity: High
- Estimated Gas Usage: 21347 - 83534
- Relationships: CWE-682: Incorrect Calculation
- Description: The arithmetic operator can overflow. It is possible to cause an integer overflow or underflow in the arithmetic operation.



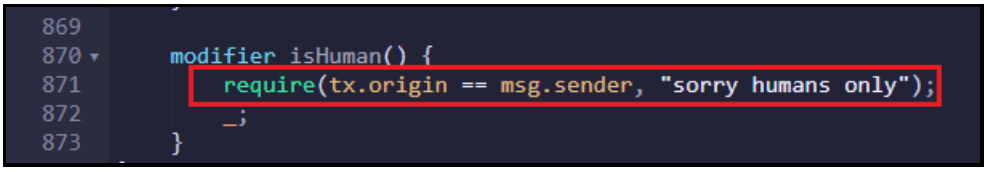
3. Dependence on predictable environment variable

- SWC ID: 116
- Severity: Low
- Relationships: CWE-829: Inclusion of Functionality from Untrusted Control Sphere
- Description: A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.



4. Authorization through tx.origin

- SWC ID: 115
- Severity: Low
- Relationships: CWE-477: Use of Obsolete Function
- Description: Use of "tx.origin" as a part of authorization control. The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.



5. Weak Sources of Randomness from Chain Attributes

- SWC ID: 120
- Severity: Low
- Relationships: CWE-330: Use of Insufficiently Random Values
- Description: Potential use of "block.number" as source of randonmness. The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

```
683     function random(uint256 from, uint256 to, uint256 salty) private view returns (uint256) {
684         uint256 seed = uint256(
685             keccak256(
686                 abi.encodePacked(
687                     block.timestamp + block.difficulty +
688                     ((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (block.timestamp)) +
689                     block.gaslimit +
690                     ((uint256(keccak256(abi.encodePacked(msg.sender)))) / (block.timestamp)) +
691                     block.number +
692                     salty
693                 )
694             )
695         );
696         return seed.mod(to - from) + from;
697     }
```