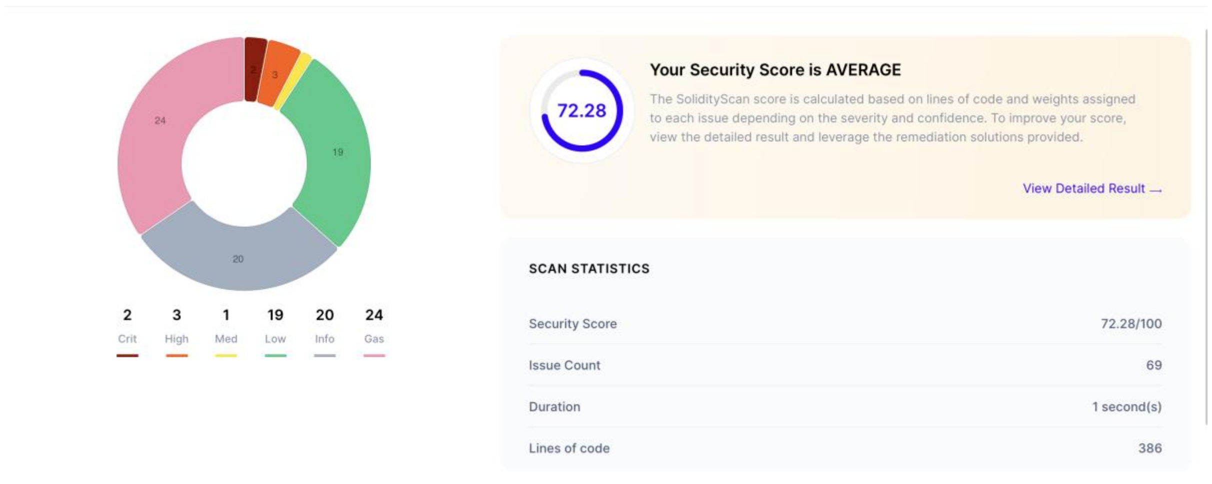


72.28

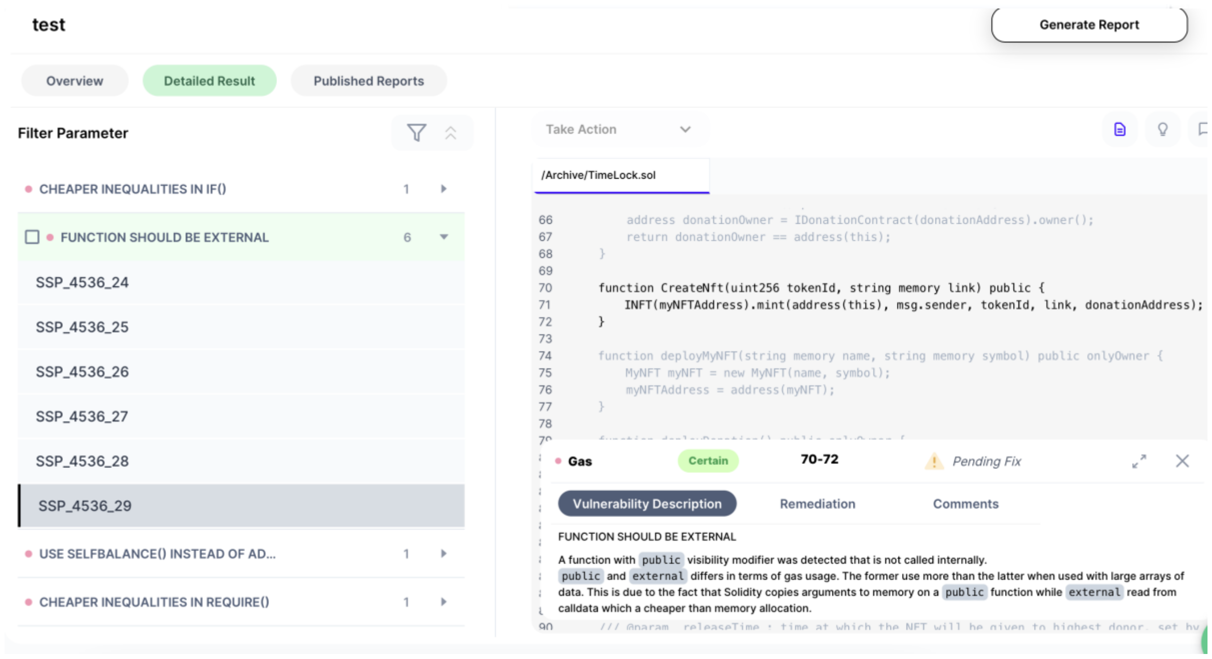
Your Security Score is AVERAGE

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

View Detailed Result →

SCAN STATISTICS

Security Score	72.28/100
Issue Count	69
Duration	1 second(s)
Lines of code	386



test

Filter Parameter

CHEAPER INEQUALITIES IN IF()	1
FUNCTION SHOULD BE EXTERNAL	6
SSP_4536_24	
SSP_4536_25	
SSP_4536_26	
SSP_4536_27	
SSP_4536_28	
SSP_4536_29	
USE SELFBALANCE() INSTEAD OF AD...	1
CHEAPER INEQUALITIES IN REQUIRE()	1

/Archive/TimeLock.sol

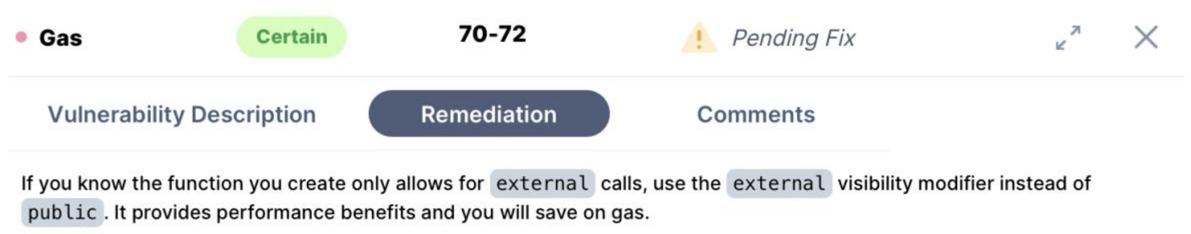
```
        address donationOwner = IDonationContract(donationAddress).owner();
        return donationOwner == address(this);
    }

    function CreateNft(uint256 tokenId, string memory link) public {
        INFT(myNFTAddress).mint(address(this), msg.sender, tokenId, link, donationAddress);
    }

    function deployMyNFT(string memory name, string memory symbol) public onlyOwner {
        MyNFT myNFT = new MyNFT(name, symbol);
        myNFTAddress = address(myNFT);
    }

    /// @param  releaseTime : time at which the NFT will be given to highest donor, set by
```

Gas | Certain | 70-72 | Pending Fix

Vulnerability Description | Remediation | Comments

FUNCTION SHOULD BE EXTERNAL

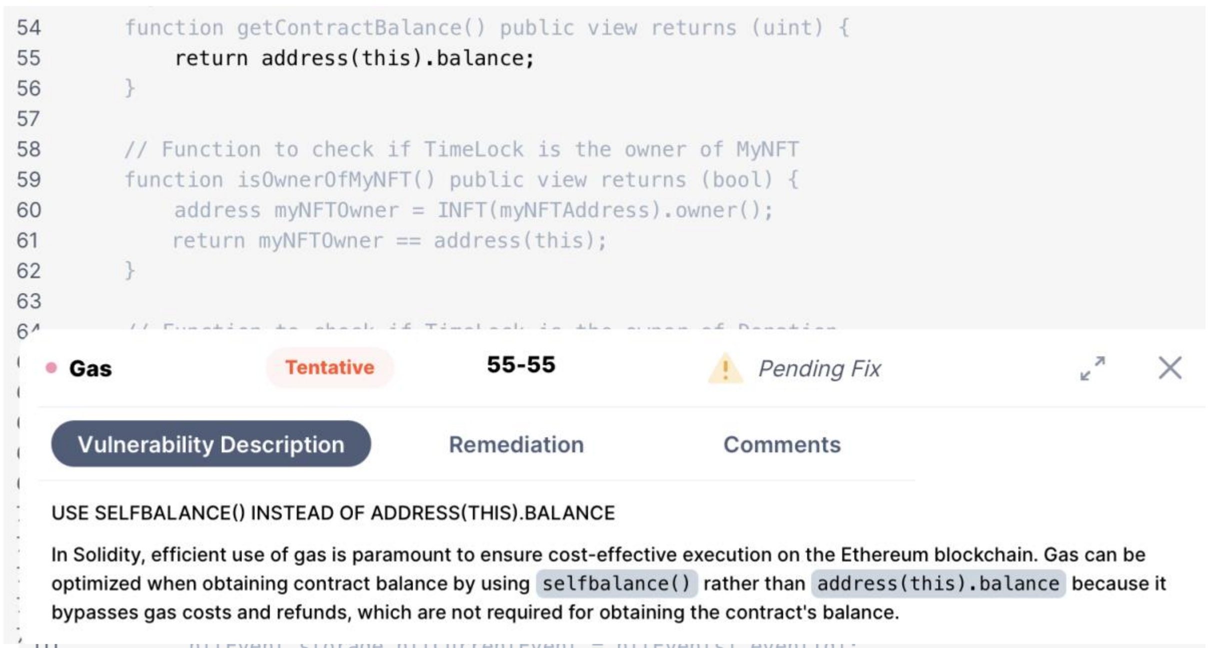
A function with `public` visibility modifier was detected that is not called internally.
`public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.



Gas | Certain | 70-72 | Pending Fix

Vulnerability Description | Remediation | Comments

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.



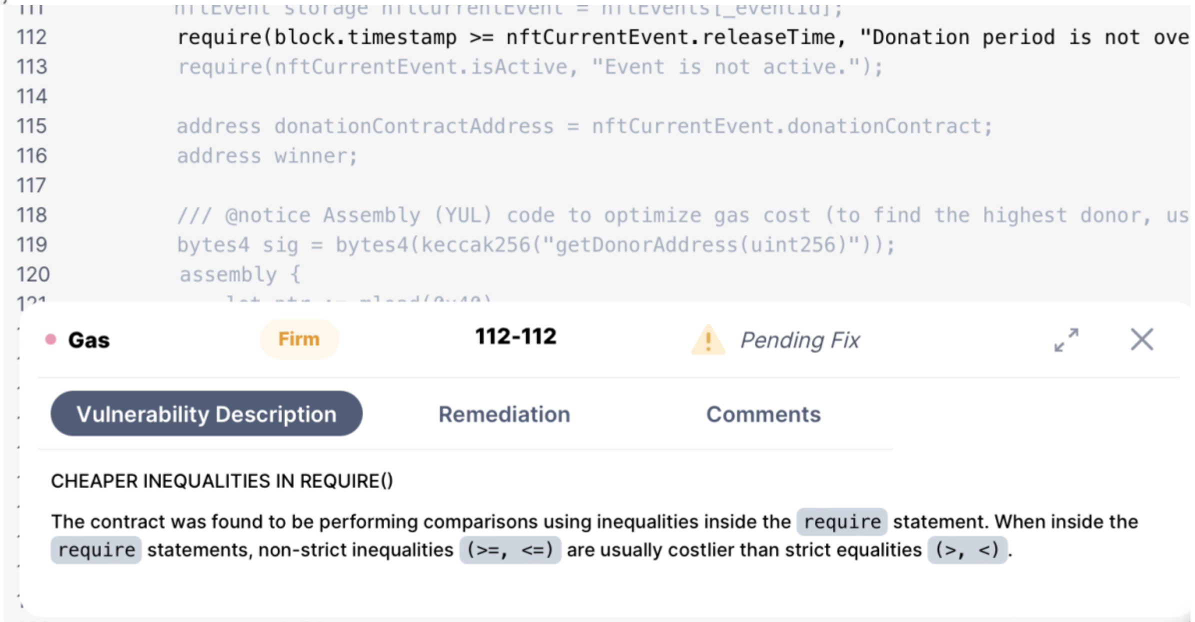
```
54      function getContractBalance() public view returns (uint) {
55          return address(this).balance;
56      }
57
58      // Function to check if TimeLock is the owner of MyNFT
59      function isOwnerOfMyNFT() public view returns (bool) {
60          address myNFTOwner = INFT(myNFTAddress).owner();
61          return myNFTOwner == address(this);
62      }
63
64      // Function to check if TimeLock is the owner of Donation
```

Gas | Tentative | 55-55 | Pending Fix

Vulnerability Description | Remediation | Comments

USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using `selfbalance()` rather than `address(this).balance` because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.



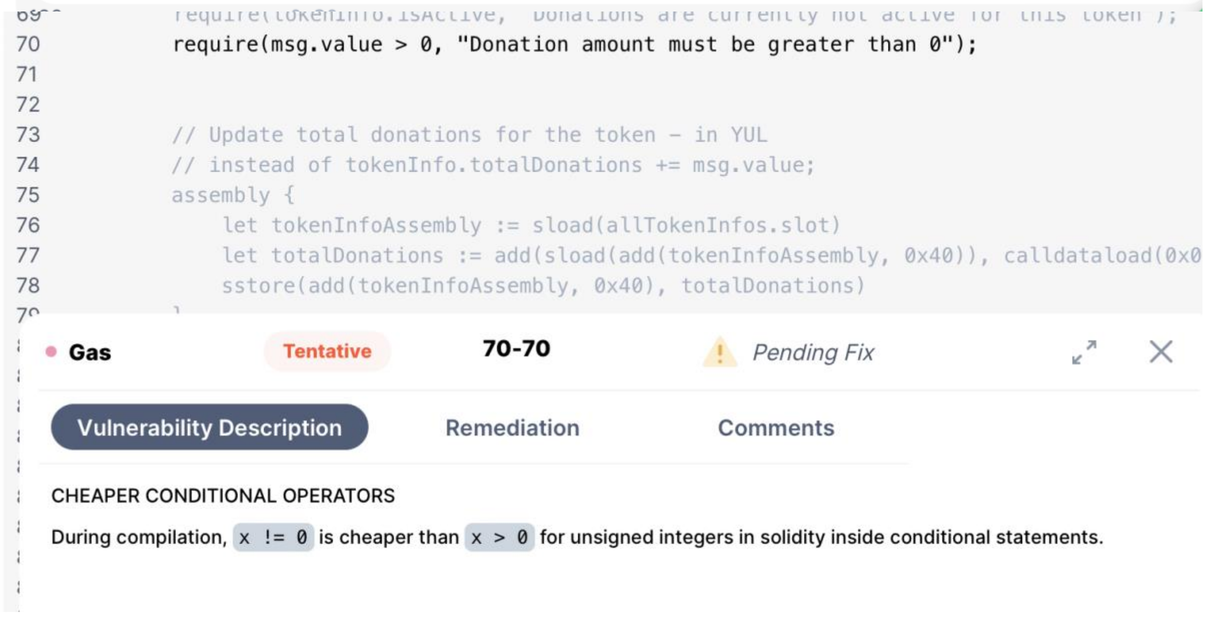
```
111          nftEvent storage nftCurrentEvent = nftEvents[_eventId];
112          require(block.timestamp >= nftCurrentEvent.releaseTime, "Donation period is not ove
113          require(nftCurrentEvent.isActive, "Event is not active.");
114
115          address donationContractAddress = nftCurrentEvent.donationContract;
116          address winner;
117
118          /// @notice Assembly (YUL) code to optimize gas cost (to find the highest donor, us
119          bytes4 sig = bytes4(keccak256("getDonorAddress(uint256)"));
120          assembly {
121              let ptr := mload(0x40)
```

Gas | Firm | 112-112 | Pending Fix

Vulnerability Description | Remediation | Comments

CHEAPER INEQUALITIES IN REQUIRE()

The contract was found to be performing comparisons using inequalities inside the `require` statement. When inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.



```
69          require(tokenInfo.isActive, "Donations are currently not active for this token");
70          require(msg.value > 0, "Donation amount must be greater than 0");
71
72
73          // Update total donations for the token - in YUL
74          // instead of tokenInfo.totalDonations += msg.value;
75          assembly {
76              let tokenInfoAssembly := sload(allTokenInfos.slot)
77              let totalDonations := add(sload(add(tokenInfoAssembly, 0x40)), calldataload(0x0
78              sstore(add(tokenInfoAssembly, 0x40), totalDonations)
79          }
```

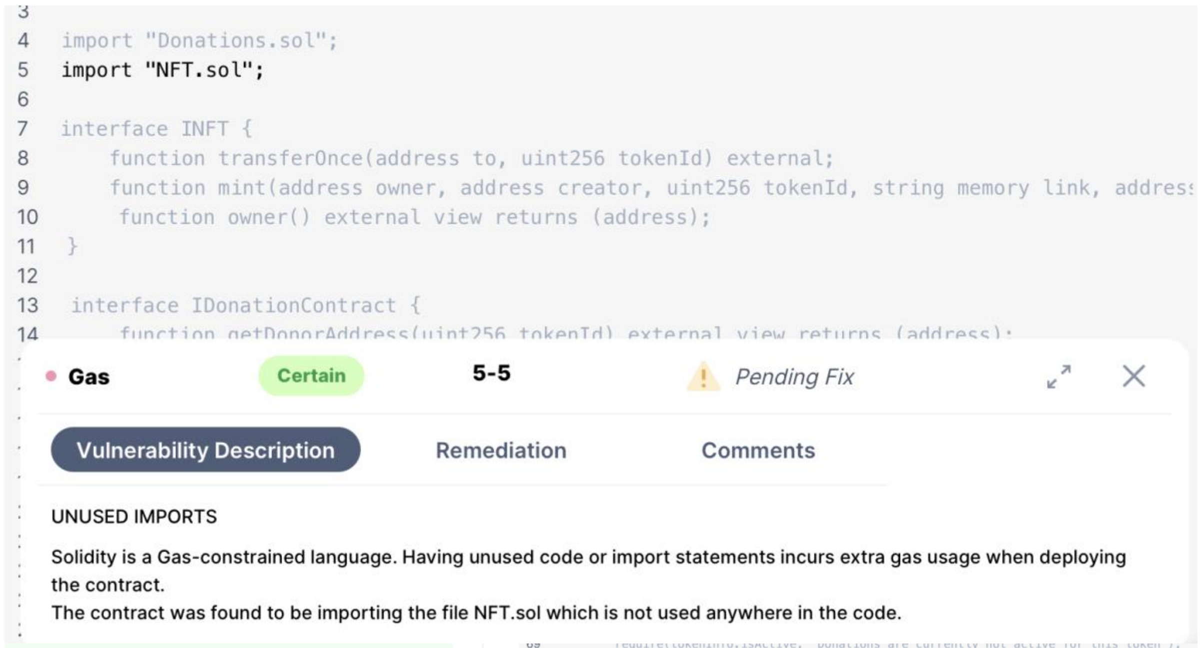
Gas | Tentative | 70-70 | Pending Fix

Vulnerability Description | Remediation | Comments

CHEAPER CONDITIONAL OPERATORS

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

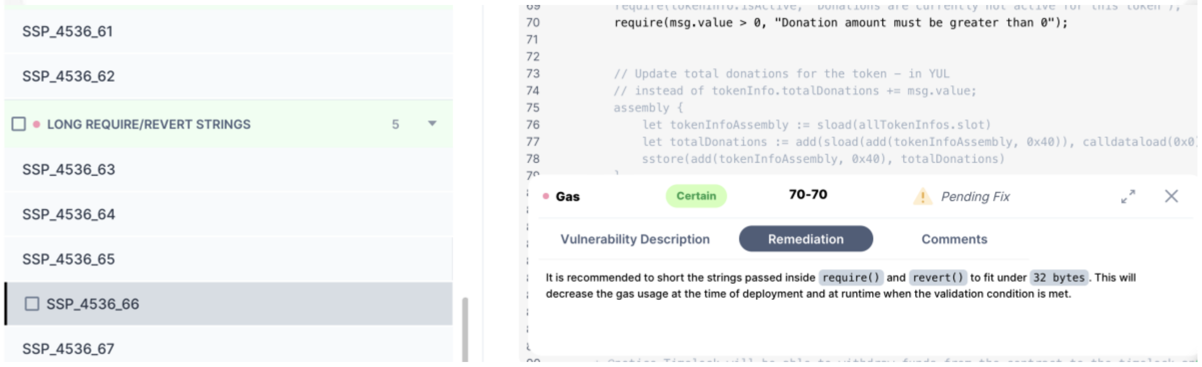
Since we used the interface instead



```
import "Donations.sol";
import "NFT.sol";

interface INFT {
    function transferOnce(address to, uint256 tokenId) external;
    function mint(address owner, address creator, uint256 tokenId, string memory link, addres
     function owner() external view returns (address);
 }

 interface IDonationContract {
     function getDonorAddress(uint256 tokenId) external view returns (address);
```

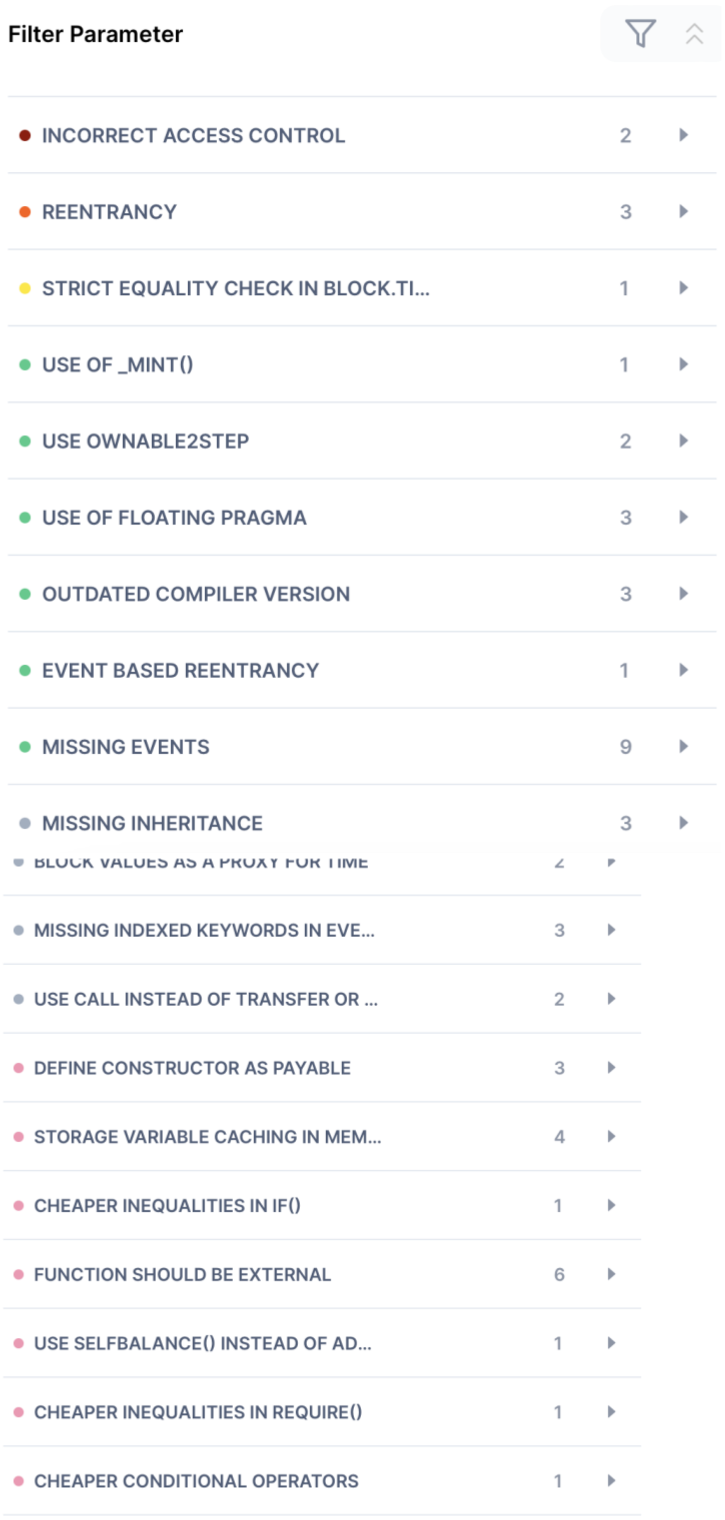
UNUSED IMPORTS

Solidity is a Gas-constrained language. Having unused code or import statements incurs extra gas usage when deploying the contract.
The contract was found to be importing the file NFT.sol which is not used anywhere in the code.



```
        require(tokenInfo.isActive, "Donations are currently not active for this token");
        require(msg.value > 0, "Donation amount must be greater than 0");


        // Update total donations for the token - in YUL
        // instead of tokenInfo.totalDonations += msg.value;
        assembly {
            let tokenInfoAssembly := sload(allTokenInfos.slot)
            let totalDonations := add(sload(add(tokenInfoAssembly, 0x40)), calldataload(0x0
            sstore(add(tokenInfoAssembly, 0x40), totalDonations)
```

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.



Filter Parameter

●	INCORRECT ACCESS CONTROL	2
●	REENTRANCY	3
●	STRICT EQUALITY CHECK IN BLOCK.TI...	1
●	USE OF _MINT()	1
●	USE OWNABLE2STEP	2
●	USE OF FLOATING PRAGMA	3
●	OUTDATED COMPILER VERSION	3
●	EVENT BASED REENTRANCY	1
●	MISSING EVENTS	9
●	MISSING INHERITANCE	3
●	BLOCK VALUES AS A PROXY FOR TIME	2
●	MISSING INDEXED KEYWORDS IN EVE...	3
●	USE CALL INSTEAD OF TRANSFER OR ...	2
●	DEFINE CONSTRUCTOR AS PAYABLE	3
●	STORAGE VARIABLE CACHING IN MEM...	4
●	CHEAPER INEQUALITIES IN IF()	1
●	FUNCTION SHOULD BE EXTERNAL	6
●	USE SELFBALANCE() INSTEAD OF AD...	1
●	CHEAPER INEQUALITIES IN REQUIRE()	1
●	CHEAPER CONDITIONAL OPERATORS	1