Novem NNNToken Remediations Review 04/12/2020

Open Issues Review

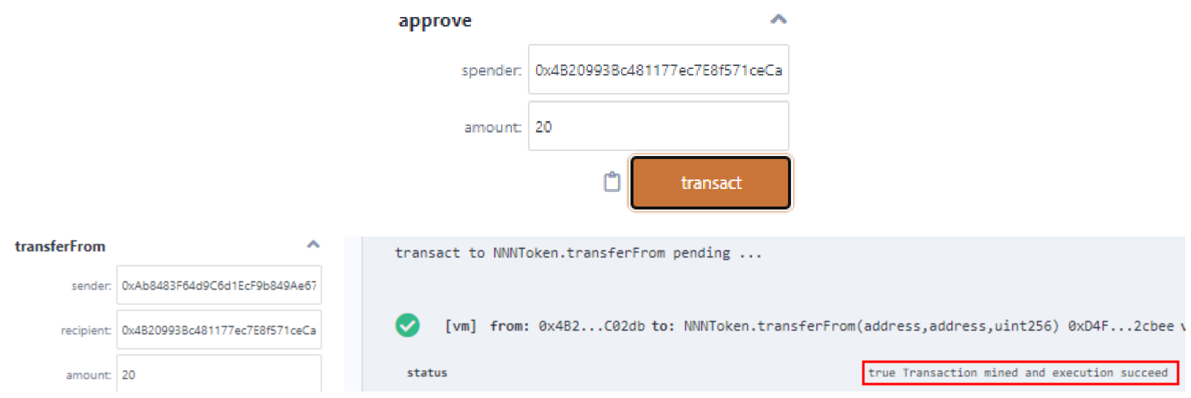
Vulnerability	State
Possible wrong initialization	**Fixed**
Possible front-running in initialize methods	**Assumed**
Wrong logic around Transfer fee	**Not Fixed**
Fixed decimals	**Assumed. 18 decimals**
Provide License for Third-Party Code	**Fixed**
Outdated Compiler Version	**Fixed**

Issues & Recommendations

Fee Bypass new

To include the functionality of fee collection the ***Transfer*** method of *ERC20* has been overwritten, the problem is that not all transfer options are considered which implies that transfers can be made using ***transferFrom***, which uses the _transfer method, avoiding the payment of the fee.

Instead of overwriting Transfer you should overwrite the open zeppelin method ***_transfer.***



approve

spender: 0x4B20993Bc481177ec7E8f571ceCa

amount: 20

transferFrom

sender: 0xAb8483F64d9C6d1EcF9b849Ae67

recipient: 0x4B20993Bc481177ec7E8f571ceCa

amount: 20

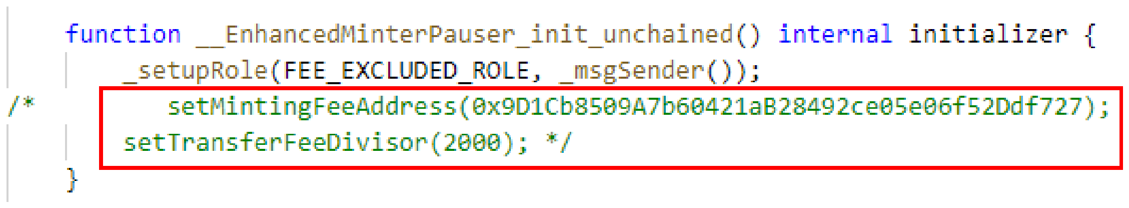
```
transact to NNNToken.transferFrom pending ...

[vm] from: 0x4B2...C02db to: NNNToken.transferFrom(address,address,uint256) 0xD4F...2cbee v

status                                  true Transaction mined and execution succeed
```

Reference:

- https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/5e1f53a0a92f257229dc882b90742a59747c594d/contracts/token/ERC20/ERC20Upgradeable.sol#L214

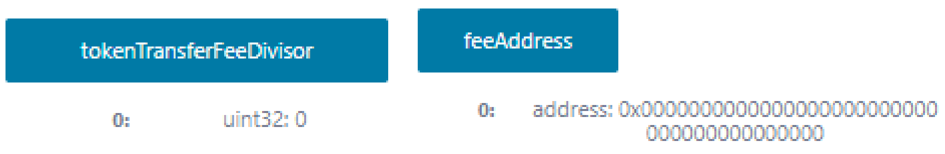
Incomplete initialization

The value of *feeAddress* and of *tokenTransferFeeDivisor* is not set at the time of initializing the contract.

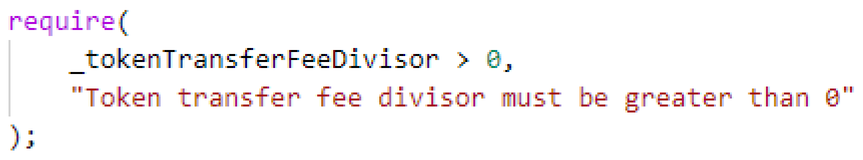


```
    function __EnhancedMinterPauser_init_unchained() internal initializer {
        _setupRole(FEE_EXCLUDED_ROLE, _msgSender());
/*          setMintingFeeAddress(0x9D1Cb8509A7b60421aB28492ce05e06f52Ddf727);
        setTransferFeeDivisor(2000); */
    }
```

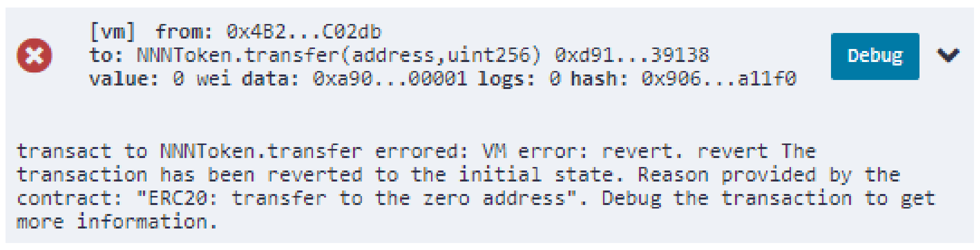
Below you can see how these values are necessary according to the logic of the contract, therefore, at the time of initializing the contract it is not functional.



```
0:    uint32: 0
0:    address: 0x0000000000000000000000000000000000000000
```



```
require(
    _tokenTransferFeeDivisor > 0,
    "Token transfer fee divisor must be greater than 0"
);
```



```
[vm] from: 0x4B2...C02db
to: NNNToken.transfer(address,uint256) 0xd91...39138
value: 0 wei data: 0xa90...00001 logs: 0 hash: 0x906...a11f0

transact to NNNToken.transfer errored: VM error: revert. revert The
transaction has been reverted to the initial state. Reason provided by the
contract: "ERC20: transfer to the zero address". Debug the transaction to get
more information.
```

We recommend setting the corresponding values when initializing the contract so that you do not have to perform more transactions in order to complete this task.

Confusing Transfers Fee

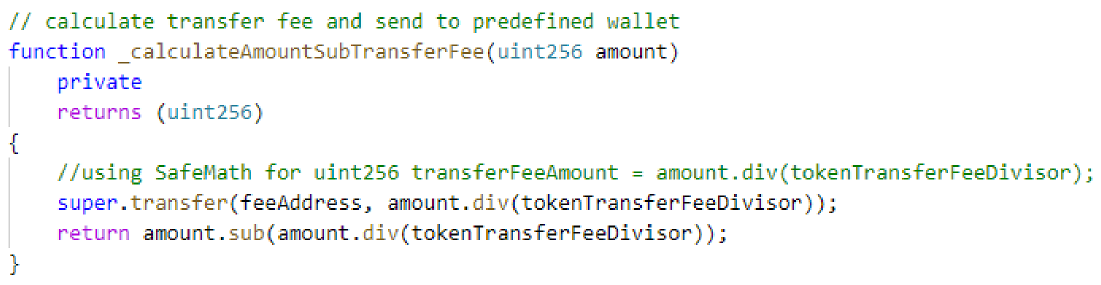
The contract uses the *tokenTransferFeeDivisor* variable to calculate the fee charged by the project during the transfers. In this process there are several failures and/or inconsistencies.

1. Using a dividend becomes complicated when calculating commissions, it becomes easier and clearer if *tokenTransferFee* was a percentage and divide the amount by *100* and divide by this value.

   However, if the current logic fits into your business model it would not be considered an error.

2. A minimum limit of *1* has been set, but no maximums have been set. It would also be advisable to set this minimum to *2*, since a lower value transfers the entire amount as a fee.

3. The current model does not consider establishing a free fee for any special case or promotion.



```
// calculate transfer fee and send to predefined wallet
function _calculateAmountSubTransferFee(uint256 amount)
    private
    returns (uint256)
{
    //using SafeMath for uint256 transferFeeAmount = amount.div(tokenTransferFeeDivisor);
    super.transfer(feeAddress, amount.div(tokenTransferFeeDivisor));
    return amount.sub(amount.div(tokenTransferFeeDivisor));
}
```

RED4SEC

Invest in Security, invest in your future