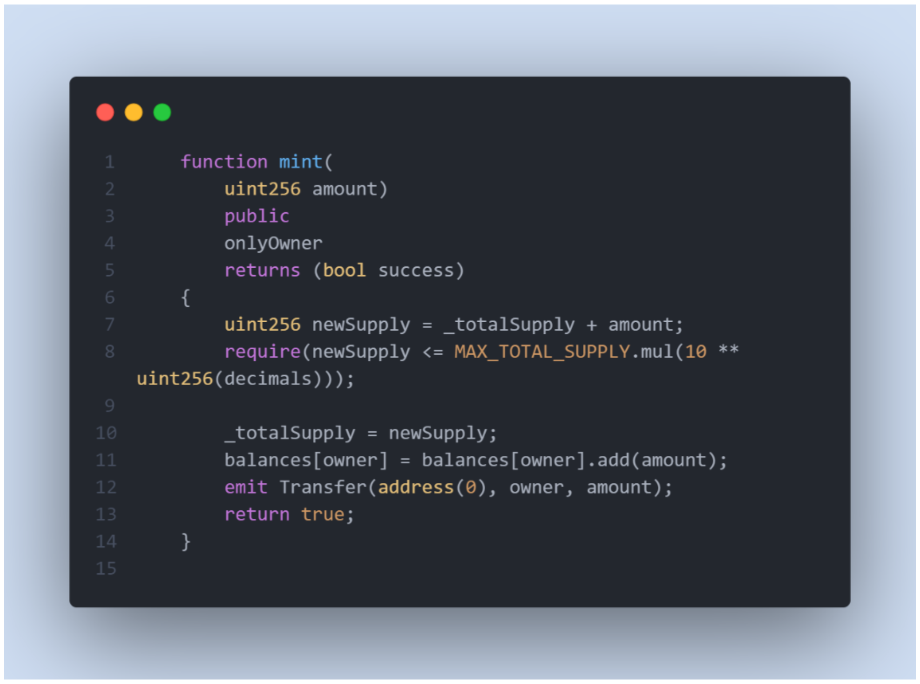
https://www.certik.com/projects/buddydaotoken

High severity issues

- **CENTRALIZATION RISKS**

The owner has the authority to function mint and can mint tokens to any wallet address.



```
    function mint(
        uint256 amount)
        public
        onlyOwner
        returns (bool success)
    {
        uint256 newSupply = _totalSupply + amount;
        require(newSupply <= MAX_TOTAL_SUPPLY.mul(10 **
uint256(decimals)));

        _totalSupply = newSupply;
        balances[owner] = balances[owner].add(amount);
        emit Transfer(address(0), owner, amount);
        return true;
    }
```

Remediation

The risk describes the current project design and potentially makes iterations to improve in the securityoperation and level of decentralization, which in most cases cannot be resolved entirely at the presentstage. We advise the client to carefully manage the privileged account's private key to avoid any potentialrisks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol beimproved via a decentralized mechanism or smart-contract-based accounts with enhanced securitypractices, e.g., multisignature wallets.

Low severity issues

• Unchecked returned value

In the functions below, the contract is using the transfer function without checking returned value.

function lock at line 441;

function transferWithLock at line 464;

function unlock at line 564;

Remediation

We recommend checking the returned value or using safeERC20library.

• Unchecked returned value

In the functions below, the contract is using the transfer function without checking returned value.

function lock at line 441;

function transferWithLock at line 464;

function unlock at line 564;

Remediation

We recommend checking the returned value or using safeERC20library.

• Missing Emit Events

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Remediation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

The functions shown below could be declared -immutable.

```
string public symbol;

string public name;

uint8 public decimals;
```

Remediation

We advise using the constructor function and adding these variables as immutable in the contract. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

TIMESTAMP DEPENDENCY

in the functions below there is timestamp dependency. Be aware that the timestamp of the block can be manipulated by the miner, and all direct and indirect uses of the timestamp should be considered.

Function tokensLockedAtTime;

Function tokensUnlockable;

Remediation

You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't destroy the logic of the staking contract.

IMPROPER USAGE OF PUBLIC AND EXTERNAL TYPE

public functions that are never called by the contract could be declared as external. external functions are more efficient than public functions.

Remediation

Consider using the external attribute for public functions that are never called within the contract.