Contract Asset.sol

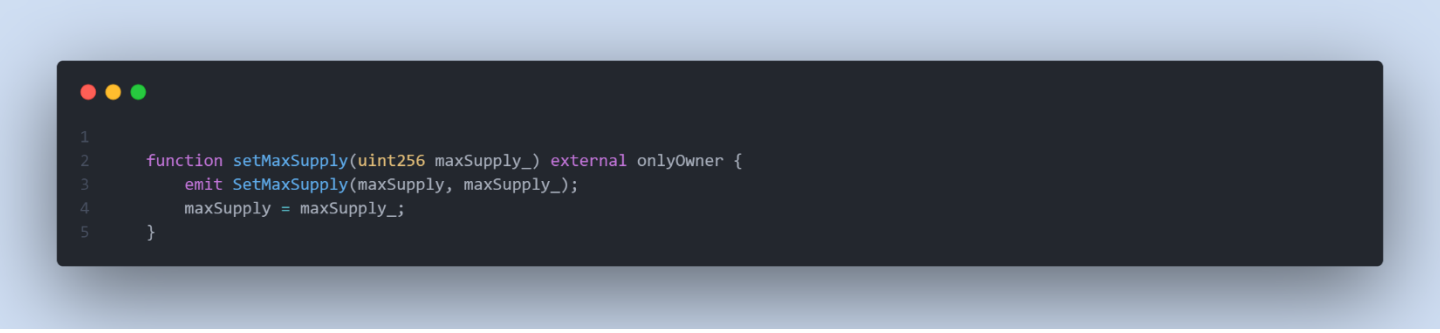
Medium severity issues

- ### CENTRALIZATION RISKS

In the contract, the owner has authority over the functions shown below.

- function setPool(address pool_);
- function setMaxSupply(uint256 maxSupply_)

```
function setPool(address pool_) external override onlyOwner {
    require(pool_ != address(0), "Wombat: Pool address cannot be zero");
    emit SetPool(pool, pool_);
    pool = pool_;
}
```



```
function setMaxSupply(uint256 maxSupply_) external onlyOwner {
    emit SetMaxSupply(maxSupply, maxSupply_);
    maxSupply = maxSupply_;
}
```

Any compromise to the onlyOwner may allow the hacker to take advantage of this authority and drastically change the stored addresses and state.

Remediation

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

Low severity issues

- ## VALUE VALIDATION CHECK

The input maxSupply_ is not checked to see if it is greater than 0 in function setMaxSupply.

```
function setMaxSupply(uint256 maxSupply_) external onlyOwner {
    emit SetMaxSupply(maxSupply, maxSupply_);
    maxSupply = maxSupply_;
}
```

Remediation

We advise adding a require statement for setMaxSupply function.

- ## ZERO ADDRESS VALIDATION CHECK

The input address to is not checked in function transferUnderlyingToken, mint,

```
function mint(address to, uint256 amount) external override onlyPool {
    if (maxSupply != 0) {
        // if maxSupply == 0, asset is uncapped.
        require(
            amount + this.totalSupply() <= maxSupply,
            "Wombat: MAX_SUPPLY_REACHED"
        );
    }
    return _mint(to, amount);
}
```

```
function transferUnderlyingToken(address to, uint256 amount)
    external
    override
    onlyPool
{
    IERC20(underlyingToken).safeTransfer(to, amount);
}
```

Remediation

We advise adding a require statement for above functions about to input, to not be zero address.

- ## 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.

Contract Pool.sol

Medium severity issues

- ### CENTRALIZATION RISKS

In the contract, the owner has authority over the functions shown below.

- function pause();
- function unpause();
- function pauseAsset();
- function unpauseAsset();
- Function setDev();
- Function setAmpFactor();
- Function setHaircutRate();
- Function addAsset();
- Function removeAsset()

Any compromise to the onlyOwner and _onlyDev account may allow the hacker to take advantage of this authority and drastically change the stored addresses and state.

Remediation

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