SMART
state

SMART CONTRACT AUDIT

Neko

Experimental Network of Maze Protocol

PROJECT:

NEKO

METHODOLOGY

Main tests list:

- Best code practices
- ERC20/BEP20 compliance (if applicable)
- Logical bugs
- General Denial Of Service(DOS)
- Locked ether
- Private data leaks
- Using components with known vulns
- Weak PRNG
- Unsed vars
- Uncheked call return method
- Code with no effects
- Function visibility
- Use of deprecated functions
- Authorization issues
- Re-entrancy
- Arithmetic Over/Under Flows
- Hidden Malicious Code
- External Contract Referencing
- Short Address/ Parameter Attack
- Race Conditions / Front Running
- Uninitialized Storage Pointers
- Floating Points and Precision
- Signatures Replay
- Pool Asset Security (backdoors in the underlying ERC-20)

STRUCTURE OF CONTRACT

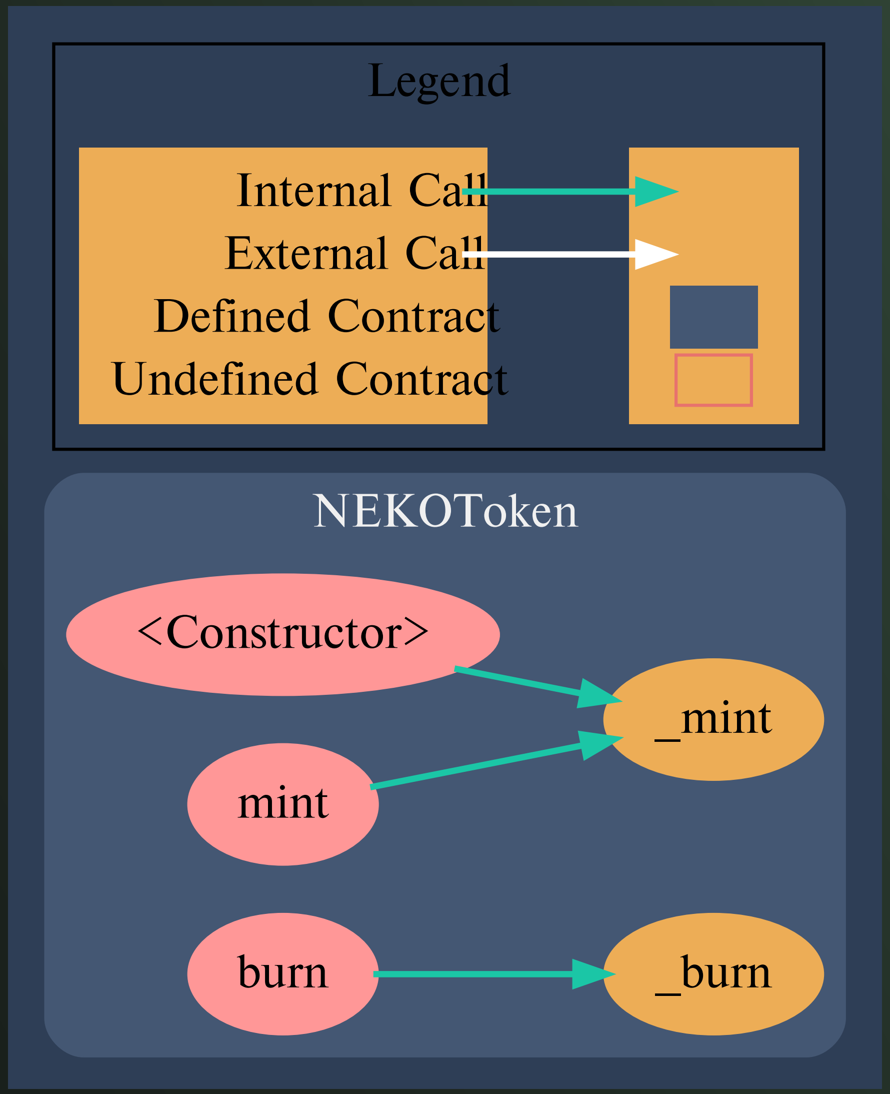
NEKOTOKEN.SOL

Constrcutor lacks check for 0 address.

Contract methods analysis

mint(address user, uint256 amount)
Vulnerabilities not detected

burn(address user, uint256 amount)
Vulnerabilities not detected



Pic. 1.1.
NEKOToken.sol

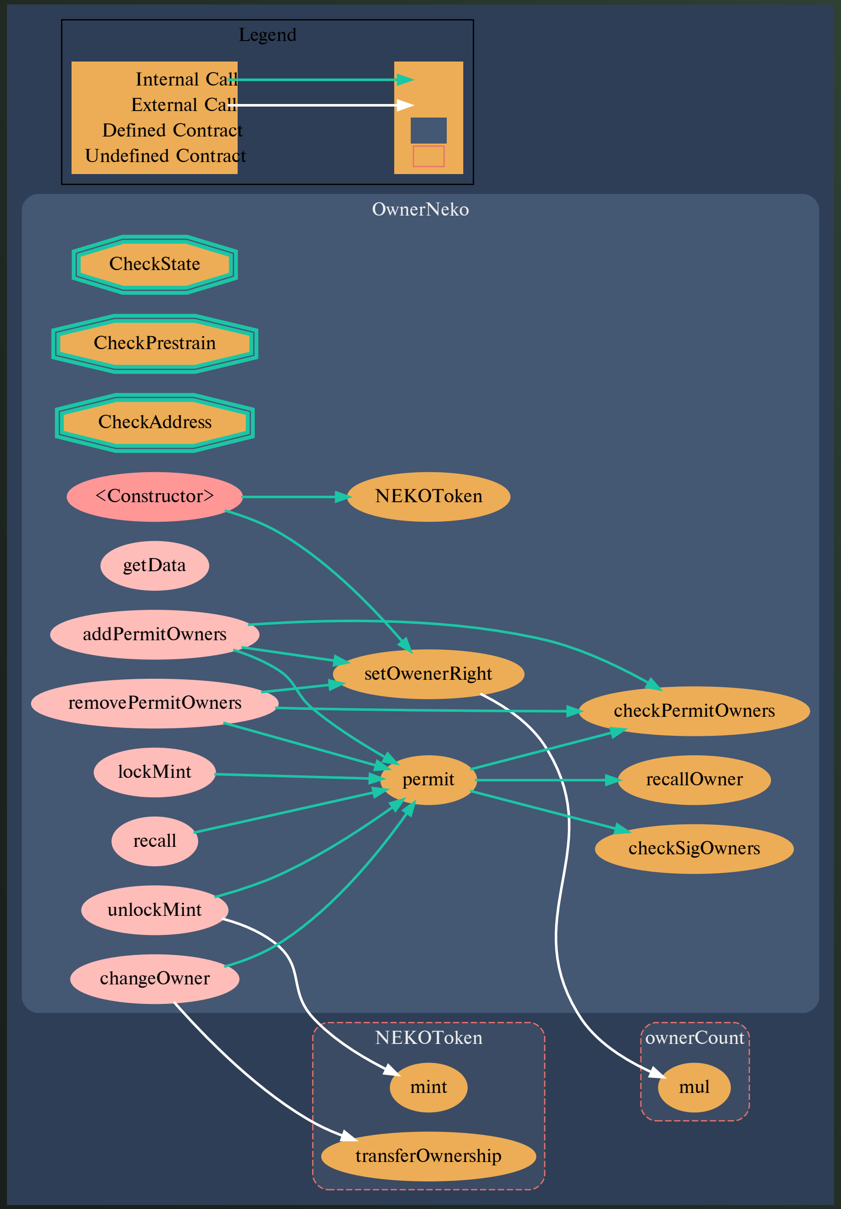
STRUCTURE OF CONTRACT OWNERNEKO.SOL

Vulnerabilities not detected

Contract methods analysis

getData()
Vulnerabilities not detected

setOwenerRight()
Vulnerabilities not detected



Pic. 1.2.
OwnerNeko.sol

```
addPermitOwners
  (
      address owner,
      address spender,
      uint value,
      uint deadline,
      uint8 v,
      bytes32 r,
      bytes32 s
  )
```
Instead of assigning to result and then checking result variable you can do require(result);

```
removePermitOwners
  (
      address owner,
      address spender,
      uint value,
      uint deadline,
      uint8 v,
      bytes32 r,
      bytes32 s
  )
```
You should cache permitOwners array here for gas optimization. E.g, initialize new local variable like _tmpPermitOwners = permitOwners. Refering to memory in solidity in much cheaper than refering to storage.

checkSigOwners (address[] memory owners, address owner, uint state)
Vulnerabilities not detected

checkPermitOwners (address owner)
You should cache permitOwners array here for gas optimization. E.g, initialize new local variable like _tmpPermitOwners = permitOwners. Refering to memory in solidity in much cheaper than refering to storage.

recallOwner (address owner)
You chould cache data.sigOwners here for gas optimization. E.g., initialize new variable _tmpDataSigOwners. Refering to memory in solidity in much cheaper than refering to storage.

```
permit
  (
     uint256 state,
     address owner,
     address spender,
     uint value,
     uint deadline,
     uint8 v,
     bytes32 r,
     bytes32 s
  )
```
Vulnerabilities not detected

```
lockMint
  (
     address owner,
     address spender,
     uint value,
     uint deadline,
     uint8 v,
     bytes32 r,
     bytes32 s
  )
```
Instead of assigning to result and then checking result variable you can do require(result);

```
unlockMint
  (
     address owner,
     address spender,
     uint value,
     uint deadline,
     uint8 v,
     bytes32 r,
     bytes32 s
  )
```
Instead of assigning to result and then checking result variable you can do require(result);

```
function changeOwner
  (
     address owner,
     address spender,
     uint value,
     uint deadline,
     uint8 v,
     bytes32 r,
     bytes32 s
  )
```
Instead of assigning to result and then checking result variable you can do require(result);

```
function recall
  (
     uint state,
     address owner,
     address spender,
     uint value,
     uint deadline,
     uint8 v,
     bytes32 r,
     bytes32 s
  )
```
Instead of assigning to result and then checking result variable you can do require(result);

GET IN TOUCH

info@smartstate.tech
smartstate.tech