

## Incorrect Constructor Name

### Description

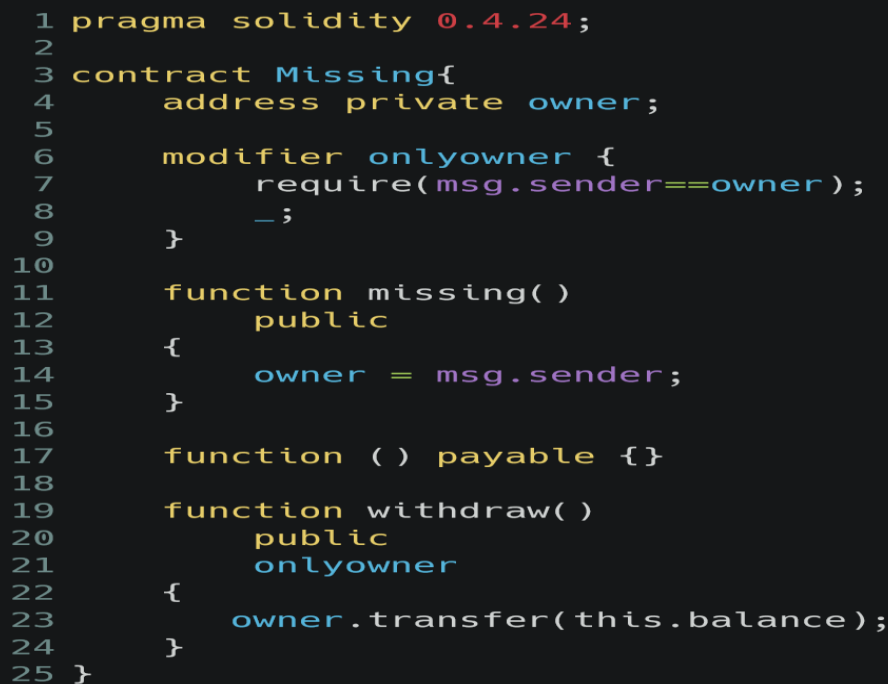
Constructors are special functions that are called only once during the contract creation. They often perform critical, privileged actions such as setting the owner of the contract. Before Solidity version 0.4.22, the only way of defining a constructor was to create a function with the same name as the contract class containing it. A function meant to become a constructor becomes a normal, callable function if its name doesn't exactly match the contract name. This behavior sometimes leads to security issues, in particular when smart contract code is re-used with a different name but the name of the constructor function is not changed accordingly.

### Remediation

Solidity version 0.4.22 introduces a new constructor keyword that make a constructor definition clearer. It is therefore recommended to upgrade the contract to a recent version of the Solidity compiler and change to the new constructor declaration.

### Example:

*Code:*



```
1 pragma solidity 0.4.24;
2
3 contract Missing{
4     address private owner;
5
6     modifier onlyowner {
7         require(msg.sender==owner);
8     }
9
10
11     function missing()
12         public
13     {
14         owner = msg.sender;
15     }
16
17     function () payable {}
18
19     function withdraw()
20         public
21         onlyowner
22     {
23         owner.transfer(this.balance);
24     }
25 }
```

*Explanation:*

Before the solidity 0.4.22 compiler, the name of a constructor and a contract were identical. However, developers often neglect to correctly define the constructor's name, resulting in vulnerability.

If the contract name gets modified, or there is a typo in the constructor's name such that it no longer matches the name of the contract, the constructor will behave like a normal function. This can lead to dire consequences, especially if the constructor is performing privileged operations.