Secureum Workshop Day 3: Graded Assignment

Questions in this graded assignment can be answered by running Kontrol commands and investigating the output. Questions are related to WETH9 functions and tests available in [secureum-kontrol](https://github.com/runtimeverification/secureum-kontrol), or to the code snippets provided in the question. When we ask for the value of something, we expect KEVM terms. Where explanation is requested, a short but informative answer would suffice. Each question is worth 5 points. Thus, a maximum of 40 points can be earned. The deadline for submissions is 2024-04-05 12:00 PM UTC. Please submit your solution to palina.tolmach@runtimeverification.com.

1. **Which command should you use to make `kontrol` generate a proof *only* for `WETH9Test`’s `**test\_approve**` *and* save every 100th node in the KCFG?**

Kontrol prove --match-test test\_approve\_AliceToBob

1. **What counterexample would `kontrol prove` produce for the following test for WETH9? How do you interpret it?**

    function test\_deposit\_quiz(address caller, uint256 amount) public payable {

        \_notBuiltinAddress(caller);

        vm.deal(caller, 5 ether);

        vm.prank(caller);

        asset.deposit{value: amount}();

    }

***Path condition:***

***{ true #Equals 5000000000000000000 <Int VV1\_amount\_114b9705:Int }***

***Model:***

***CONTRACT\_ID = 2***

***VV1\_amount\_114b9705 = 5000000000000000001***

***TIMESTAMP\_CELL = 0***

***NUMBER\_CELL = 0***

***ORIGIN\_ID = 10***

***VV0\_caller\_114b9705 = 0***

***CALLER\_ID = 10***

***CALLVALUE\_79cb2bc6 = 0***

It fails if amount parameter is > 5 ether

We can see this from the path condition, or from the proposed counter-example VV1\_amount\_114b9705 = 5000000000000000001

1. **As discussed in the workshop,** `\_notBuiltinAddress**` inserts additional assumptions as constraints to avoid branches we have no interest in. Please identify the constraints inserted in the nodes of `**test\_deposit\_quiz**` by**

\_notBuiltinAddress(caller);

**#And #Not ( { VV0\_caller\_114b9705:Int #Equals 491460923342184218035706888008750043977755113263 } )**

**#And #Not ( { VV0\_caller\_114b9705:Int #Equals 645326474426547203313410069153905908525362434349 } )**

**#And #Not ( { VV0\_caller\_114b9705:Int #Equals 728815563385977040452943777879061427756277306518 } )**

Can be verified also by console.log in foundry these address casted to uint256

1. **What path condition is printed as part of the failure information for this proof? What does it represent?**

***Path condition:***

***{ true #Equals 5000000000000000000 <Int VV1\_amount\_114b9705:Int }***

This represent the path where the amount parameter supplied in test\_deposit\_quiz is > 5 ether. This is because of the asset.deposit{value: amount}(); call, as the EVM revert if calling address doesn’t have sufficient balance to transfer value: amount

1. **What assumption (a single line) can be added to the above test to make it pass?**

vm.assume(amount < 5 ether);

1. **What status code (`statusCode`) would the execution end with after the above change? What does it mean?**

By adding a constraint on the value amount can take, the previous path condition where amount is >5 ether is skipped, and only **EVMC\_SUCCESS** status code (appart from the symbolic values STATUSCODE:StatusCode) are shown in the proof.   
This means there is no path that would revert in this test.

1. **What status codes (`statusCode`) do you observe during the execution of the following test, and what does it indicate?**

    function test\_approve\_status() public {

        uint256 amount = 10;

        vm.prank(alice);

        (bool success,) = address(asset).staticcall(abi.encodeWithSignature("approve(address,uint256)", bob, amount));

        assert(success);

    }

When executing the test running kontrol prove, we see: statusCode: EVMC\_REVERT

But the path condition isn’t explicit, we only have #top, which based on evm-semantics means that all the claims have been proven.

Running then kontrol show for that proof gives more details, and we can see there’s a path reverting with **statusCode: EVMC\_STATIC\_MODE\_VIOLATION**

**This is because we call a storage modifying function through a static call**

1. **Would the following proof get reported as failing (`PROOF FAILED`) by `kontrol prove –match-test assert\_false`? Why?**

    function assert\_false() public {

        assert(false);

    }

No because the function doesn’t starts with the `test` keyword.

Thus, any violation that are identified will not get the proof get reported as failed.