Secureum Workshop Day 2: Graded Assignment

This graded assignment asks the user to answer specific questions with Simbolik's help. Questions are grouped by the target functions of WETH9 they concern. When we ask for the value of something, we expect KEVM terms. Where an explanation is requested, a short but informative answer would suffice. Each question is worth 1 point. Thus, a maximum of 39 points can be earned. The deadline for submissions is 2024-04-04 12:00 PM UTC (noon). Please submit your solution to raoul.schaffranek@runtimeverification.com.

# WETH9.deposit()

**Q1: How many paths revert?**

1 path which is Branch 3

**Q2: How many paths succeed?**

1 path, Branch 2

**Q3: For each reverting path, explain briefly the reason for the reversal**

For Branch 3, the reason of the reversal **is a panic error with return value of 0x11**, meaning an arithmetic operation resulted in and underflow or overflow

**Q4: What's the value of the Calldata at the beginning of the function?**

**0xd0e30db0** which is the function signature of “deposit()”

**Q5: What's the value of the callvalue at the beginning of the function?**

CALLVALUE\_79cb2bc6:Int

**Q6: What's the value of the ETH balance of the WETH9 contract when the function succeeds?**

DEBUGEE\_BALANCE:Int

**Q7: What's the value of the ETH balance of the msg.sender when the function succeeds?**

MSG\_SENDER\_BALANCE:Int

**Q8: What's the value of balanceOf[msg.sender] when the function succeeds?**

I want to say:

chop ( #lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) +Int CALLVALUE\_79cb2bc6:Int

as I’m able to see that the *1st part is the formula to compute the storage slot of a the value in the mapping based on the parameter value (msg.sender) + position of the mapping itself (0x03)* while the 2nd part of the expression is simply the symbolic representation of msg.value. So the value is the addition of both parts.

# WETH9.withdraw(uint256)

**Q9: How many paths revert?**

2 paths, Branch2 and Branch4

**Q10: How many paths succeed?**

1 path, Branch 3

**Q11: For each reverting path, explain briefly the reason for the reversal:**

* Branch 2: happening on L43; it seems its because C[8]:“DEBUGEE\_BALANCE:Int <Int VV0\_wad\_114b9705:Int”, or in other words when contract balance is less than the supplied wad value to the function.
* Branch 4: It reverts on L41, require statement where `balanceOf[msg.sender]` must be at least equal to wad, also visible on C[20]:”

{ true #Equals #lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) <Int VV0\_wad\_114b9705:Int }”

**Q12: What is the value of Calldata at the beginning of the function?**

0x2e1a7d4d +Bytes #buf ( 32 , VV0\_wad\_114b9705:Int )

Signature + parameter

**Q13: What is the value of ETH balance of the WETH9 contract when the function succeeds?**

DEBUGEE\_BALANCE:Int \_-Int\_ VV0\_wad\_114b9705:Int

**Q14: What is the value of the ETH balance of the msg.sender when the function succeeds?**

MSG\_SENDER\_BALANCE:Int \_+Int\_ VV0\_wad\_114b9705:Int

**Q15: What is the value of balance[msg.sender] when the function succeeds?**

chop ( #lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) )**-**Int VV0\_wad\_114b9705:Int

# WETH9.totalSupply()

**Q16: How many paths revert?**

None

**Q17: How many paths succeed?**

One

**Q18: What's the value of the Calldata at the beginning of the function?**

0x18160ddd

signature

**Q19: At which OPCode does the function pause initially?**

SELFBALANCE

**Q20: What's the value of the Return Data when the function succeeds?**

#buf ( 32 , DEBUGEE\_BALANCE:Int )

# WETH9.approve(address,uint256)

**Q21: How many paths revert?**

None

**Q22: How many paths succeed?**

One

**Q23: What's the value of the Calldata at the beginning of the function?**

0x095ea7b3 +Bytes #buf ( 32 , VV0\_guy\_114b9705:Int ) +Bytes #buf ( 32 , VV1\_wad\_114b9705:Int )

**Q24: What is the value of allowance[msg.sender][guy] at the end of the function?**

Compared to previous function, this mapping get assigned the new value “wad”, so the previous value of the mapping is not important:

#asWord ( #buf ( 32 , VV1\_wad\_114b9705:Int ) )

# WETH9.transfer(address,uint256)

**Q25: How many paths revert?**

2 paths, Branch 3 and 4

**Q26: How many paths succeed?**

1 path, Branch 2

**Q27: What's the value of the Calldata at the beginning of the function?**

0xa9059cbb +Bytes #buf ( 32 , VV0\_dst\_114b9705:Int ) +Bytes #buf ( 32 , VV1\_wad\_114b9705:Int )

Signature + 2 parameters

**Q28: Is there any path that can reach line 65? How can you use the debugger to prove your answer?**

Its not possible as transfer calls transferFrom with src == msg.sender

We can prove it by 2 reasons:

We can prove that as there is no branch reverting on that specific line (should be shown in call stack otherwise). As there is a require statement, and

**Q29: What's the WETH balance of msg.sender when the function succeeds (assuming msg.sender != dst)?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) -Int VV1\_wad\_114b9705:Int

**Q30: What's the WETH balance of dst when the function succeeds (assuming msg.sender != dst)?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) +Int VV1\_wad\_114b9705:Int

**Q31: What's the WETH balance of msg.sender when the function succeeds assuming msg.sender == dst?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) )

# WETH9.transferFrom(address,address,uint256)

**Q32: How many paths revert?**

6 paths, Branch 3, 5, 7, 8, 9, 10

**Q33: How many paths succeed?**

3 paths, Branch 2, 4, 6

**Q34: What's the value of the Calldata at the beginning of the function?**

0x23b872dd +Bytes #buf ( 32 , VV0\_src\_114b9705:Int ) +Bytes #buf ( 32 , VV1\_dst\_114b9705:Int ) +Bytes #buf ( 32 , VV2\_wad\_114b9705:Int )

**Q35: How many paths pass through line 69?**

3 paths, Branch 2,3 and 4 if we only count “first level” paths.   
I’ve noticed that after L69, multiple branches appear, which is expected as then there is 2 branch: the one that revert because of an underflow, and the other one that passes. Same afterward on L70, there’s one branch that will revert because of overflow, and the other passes.

**Q36: What is the value of the WETH balance of src when the function succeeds assuming src != dst?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , VV0\_src\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) -Int VV2\_wad\_114b9705:Int

**Q37: What is the value of the WETH balance of dst when the function succeeds assuming src != dst?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , VV1\_dst\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) ) +Int VV2\_wad\_114b9705:Int

**Q38: What is the value of the WETH balance of src when the function succeeds assuming src == dst?**

chop ( #lookup ( DEBUGEE\_STORAGE:Map [ keccak ( #buf ( 32 , VV0\_src\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 )

<- #lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , VV0\_src\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) )

-Int VV2\_wad\_114b9705:Int ] , keccak ( #buf ( 32 , VV1\_dst\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000003 ) )

+Int VV2\_wad\_114b9705:Int )

We see that it first decrease the storage value by VV2\_wad\_114b9705:Int then increase it by the same symbolic value

**Q39: What is the value of allowance[src][msg.sender] when the function succeeds assuming src != msg.sender?**

#lookup ( DEBUGEE\_STORAGE:Map , keccak ( #buf ( 32 , MSG\_SENDER:Int ) +Bytes #buf ( 32 , keccak ( #buf ( 32 , VV0\_src\_114b9705:Int ) +Bytes 0x0000000000000000000000000000000000000000000000000000000000000004 ) ) ) ) -Int VV2\_wad\_114b9705:Int