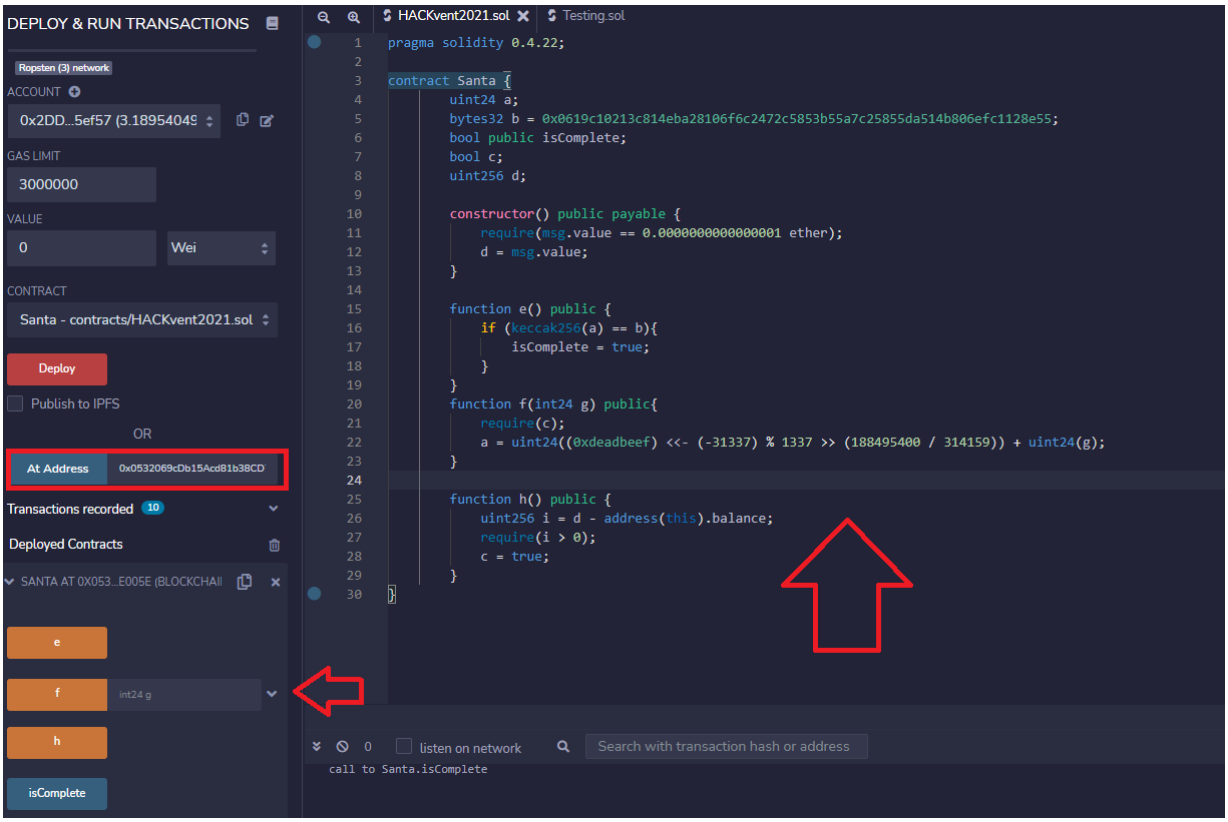
Writeup for HACKvent Challenge – super smart santa

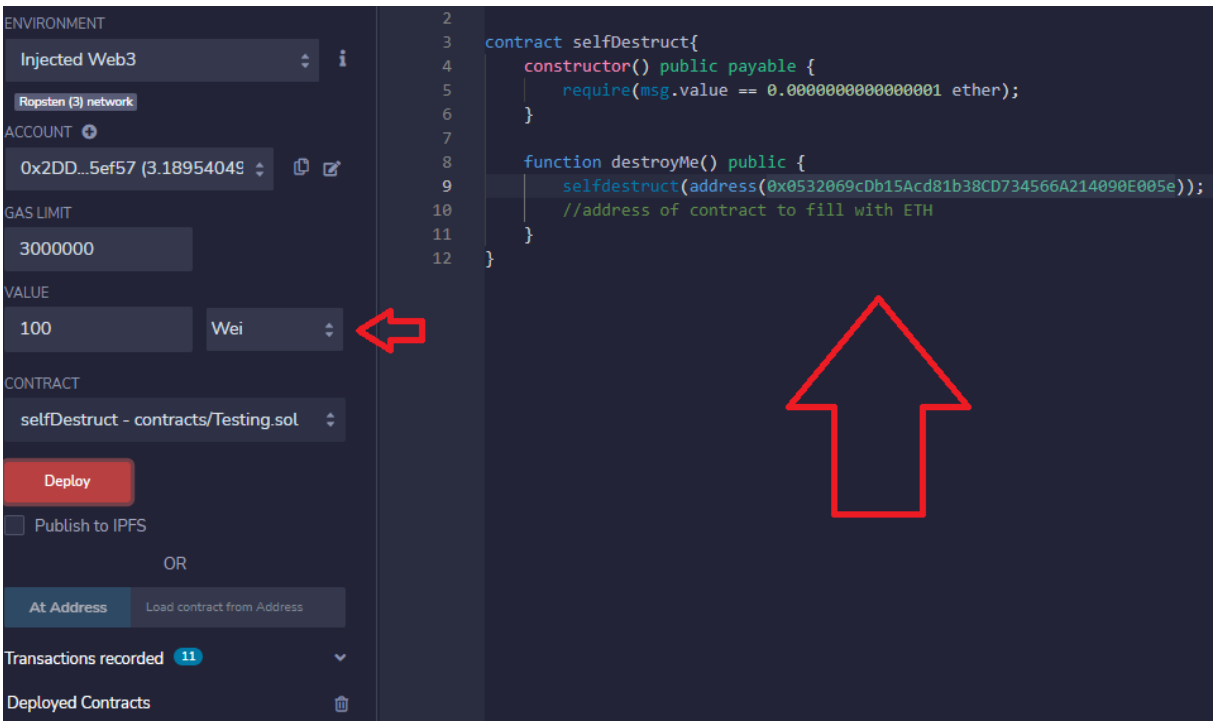
We start off by creating a new contract by clicking on the “Generate new contract” button. After waiting some seconds, we can see the address of our generated contract. Now we head to remix, an online IDE which can compile, deploy, debug and interact with smart contracts.

To interact with the generated contract, we need to paste the source code into remix, compile it using the “compile” button on the left. After that, we can paste the address into the field “At Address” to interact with the contract. We can now also see the functions we can use.



First of all, we need to set the bool “c” to true. For that, the contract’s balance needs to change. However, we don’t have any payable function to pay to. The constructor can only be called at creating, so that’s a dead end.
However, there are some methods we can use force ether into a contract, such as selfdestruct() (https://ethereum.stackexchange.com/questions/63987/can-a-contract-with-no-payable-function-have-ether/63988).

We now create a new contract with this content:



Be careful to also submit 100 Wei to the contract, as this is the wanted value in the original contract.

After the contract is created, we can call its function destroyMe() and try to call the function h() on the original contract again. It should work now and “c” should be set to true.

Great, now we are now able to set “a”. keccak256(a) should result in
0x0619c10213c814eba28106f6c2472c5853b55a7c25855da514b806efc1128e55
so let’s try to crack this hash with hashcat.
We know from that “a” is a uint24, so this limits the bruteforce to 3 bytes. A string and an int actually result in a different hash in keccak256, so you have to pay attention with the right command in hashcat. In the contract, the keccak256() function is used on an integer, so you must use the “?b” mask to be able to bruteforce it. This would look like this:

./hashcat.exe -a 3 -m 17800
'0619c10213c814eba28106f6c2472c5853b55a7c25855da514b806efc1128e55' '?b?b?b'

After not even a second, the hash is cracked: $HEX[dec0de]

dec0de in hex is equal to 14598366 in decimal, so we have to set “a” to this value: 14598366

We can set an input, however we can only set an int24, which means we can only reach 8388607, which is way below the desired value.

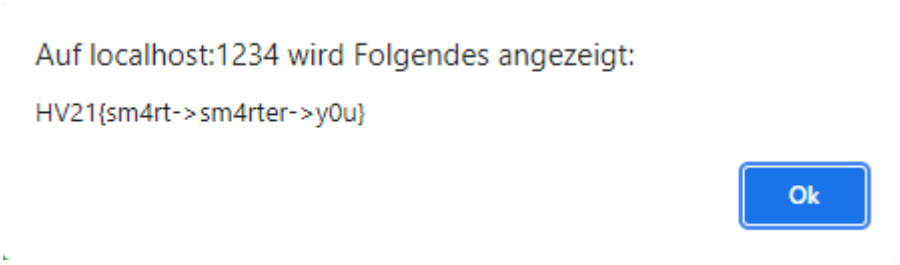
uint24((0xdeadbeef) <<- (-31337) % 1337 >> (188495400 / 314159))

^ This is actually just for confusion, this basically adds 228022 to your input. Nothing more and nothing less.

The trick to set the value to the right one is by utilizing the fact that we can input negative numbers, and our input is used as an uint24. This means that if we enter -1 as the value, it will set the value to 2**24 – 1.

Now we can calculate 2**24 – 14598366, which results in 2178850. We must use the negative value of that, obviously. Now, taking the confusion value 228022 into consideration, we must use -2178850 – 228022, which results in -2406872. We can input this now to set our “a”.

Now that we set “a” using the f() function, we can set the bool isComplete to true by executing the e() function. We can now see that isComplete is true and click on the button “Check contract” on the original page. The flag gets alerted:



HV21{sm4rt->sm4rter->y0u}

A little tip: If there’s an error that looks like this, there’s most likely a requirement unfilled: