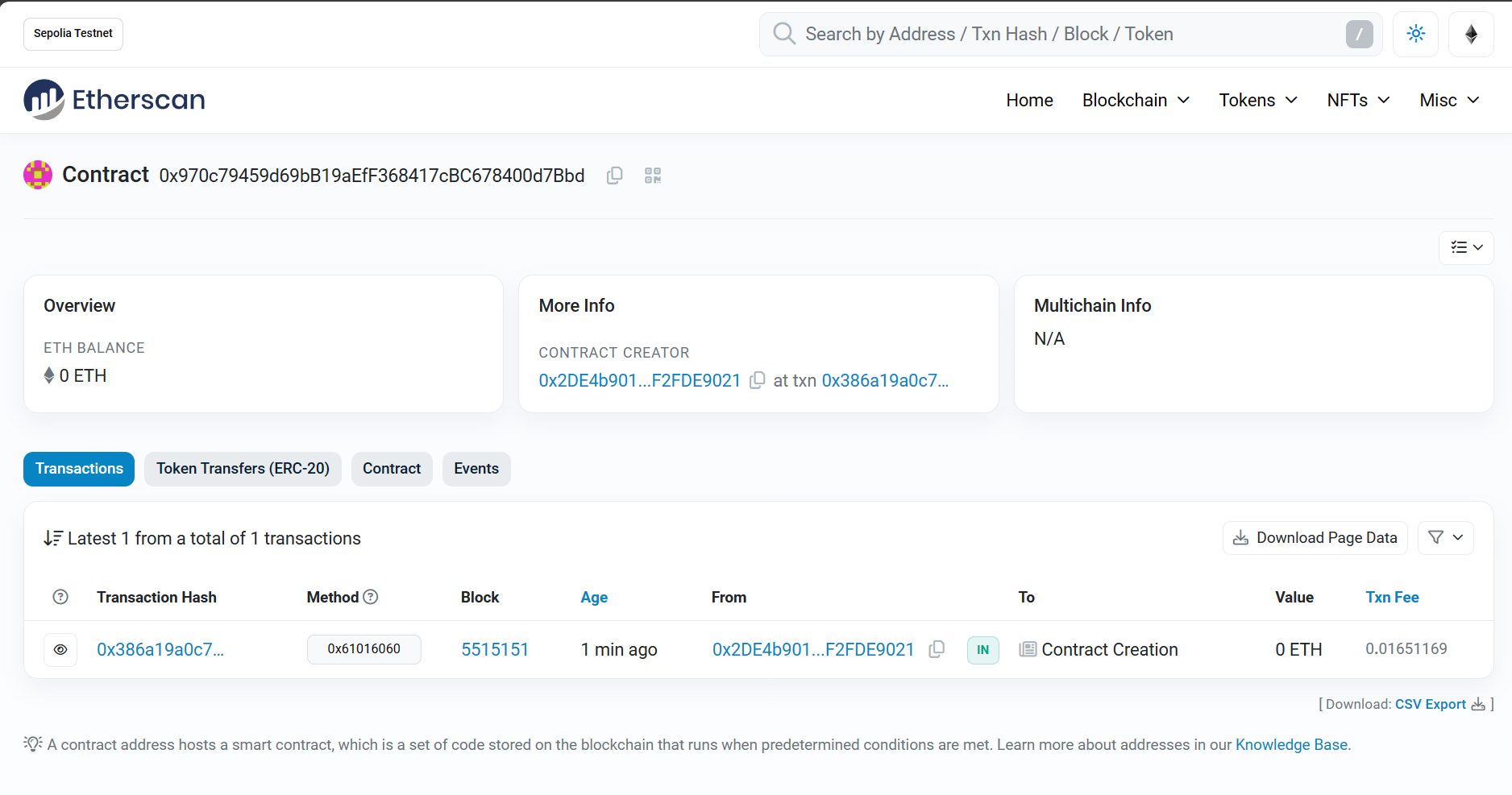
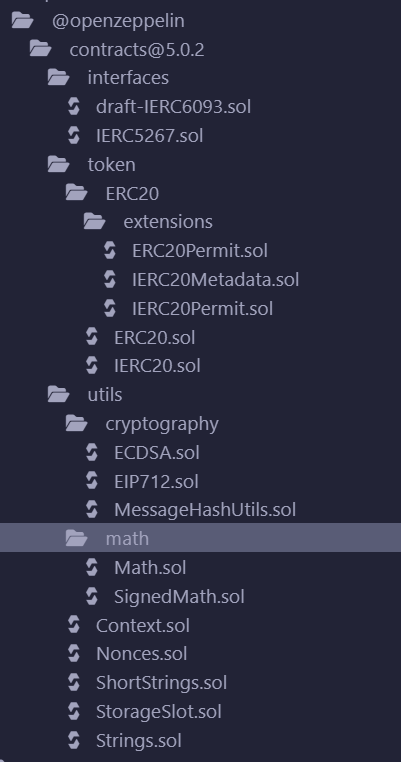
**ERC20 Token: QWWW mint and deployed**

ERC20 Token: 0x970c79459d69bb19aeff368417cbc678400d7bbd

Link: <https://sepolia.etherscan.io/address/0x970c79459d69bb19aeff368417cbc678400d7bbd>



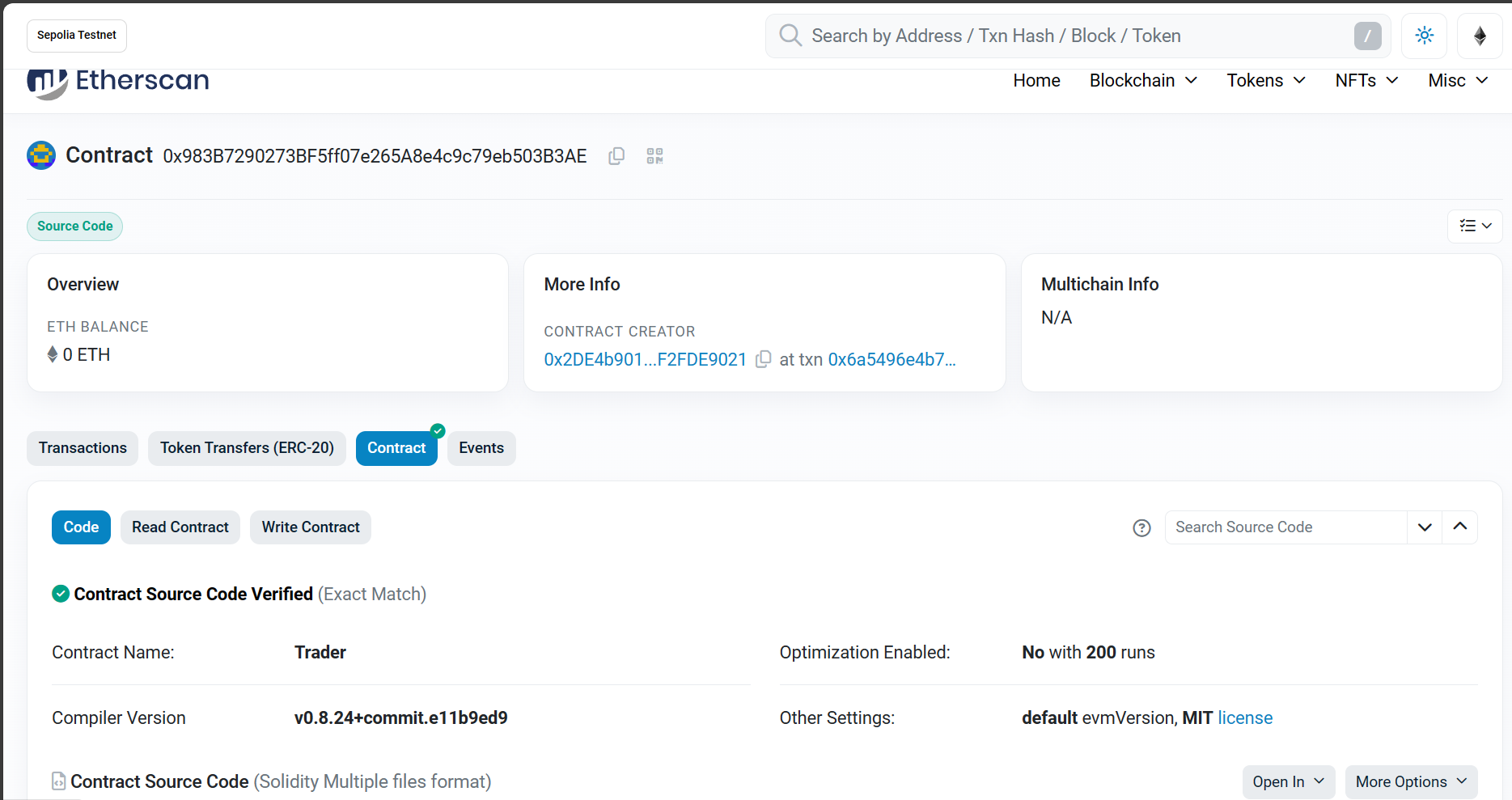
The sol. file of our ERC20 token, QWWW, was generated by OpenZeppelin’s Contract Wizard. See “contract-ERC20Token.sol” for details. The multiple layers of dependencies between files make it difficult to verify it on the Etherscan, whose multiple file verification requires uploading all the sol. files in @openzeppelin folder generated in compilation and change the way to import them in every file on the Etherscan. We are going to verify and publish the following smart contracts, trader and simpleDAO which have fewer layers of dependencies.



**Smart Contracts: Trader and SimpleDAO deployed and verified**

**Trader contract**: 0x983B7290273BF5ff07e265A8e4c9c79eb503B3AE

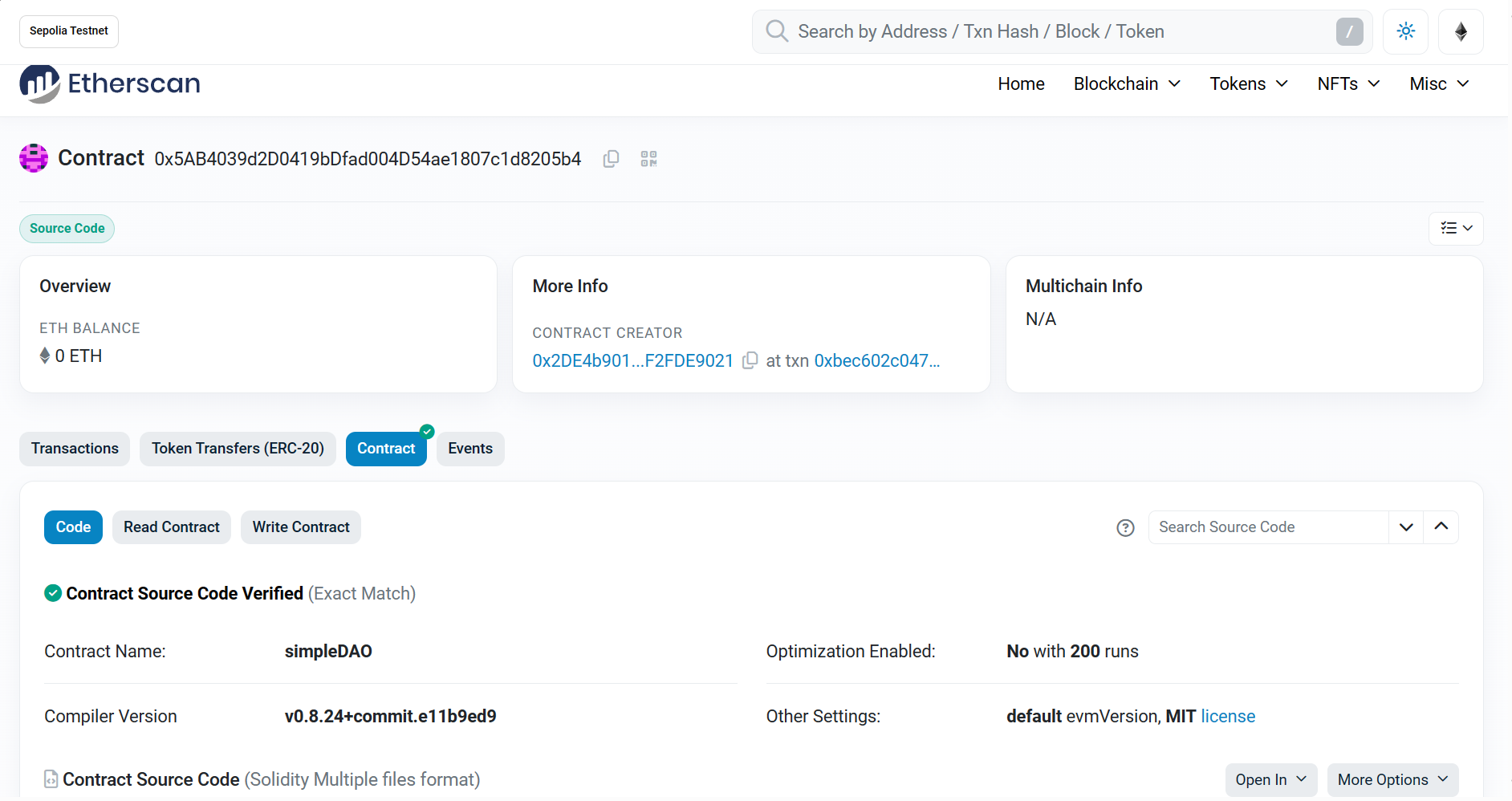
Link: https://sepolia.etherscan.io/address/0x983b7290273bf5ff07e265a8e4c9c79eb503b3ae



The trader contract imports an oracle interface, AggregatorV3Interface.sol, to get the latest BTC/ETH price.

**DAO contract**: 0x5AB4039d2D0419bDfad004D54ae1807c1d8205b4

Link: https://sepolia.etherscan.io/address/0x5AB4039d2D0419bDfad004D54ae1807c1d8205b4



The SimpleDAO contract imports the trader contract and interact with its purchase function in EndVote function.

**Test of the contracts**

**Trader contract unit tests**: See trader\_test.js.

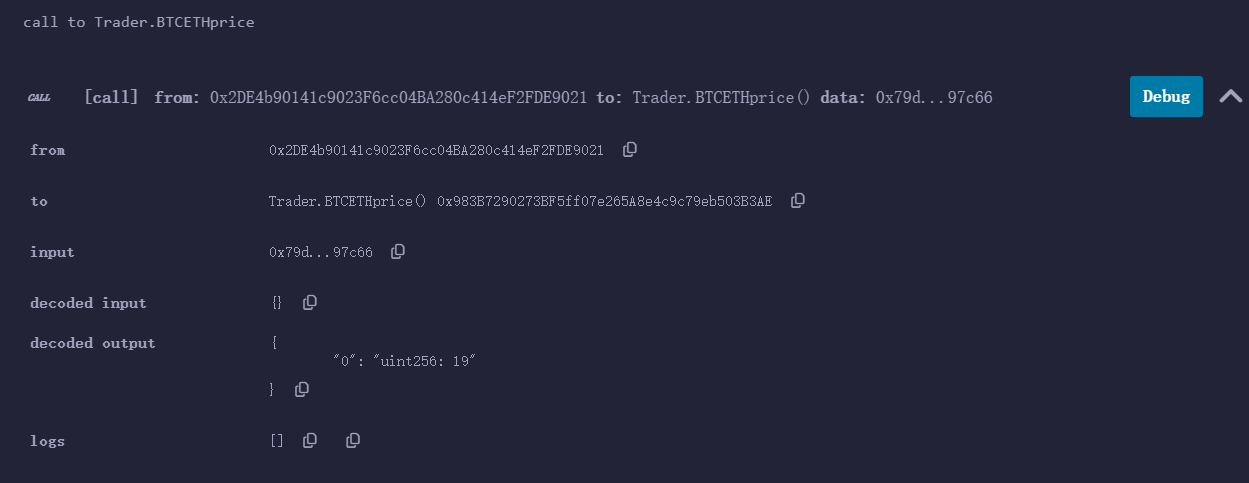
Get owner: the contract should be able to return its owner (passed)

Show ETH reserve: the contract should be able to return the ETH reserve balance of a given address (passed)

Should be able to refill: the contract should be able to add the ETH reserve of itself (passed)

Get BTC/ETH price by oracle: the contract should be able to get BTC/ETH price from oracle. (manual test, see below)

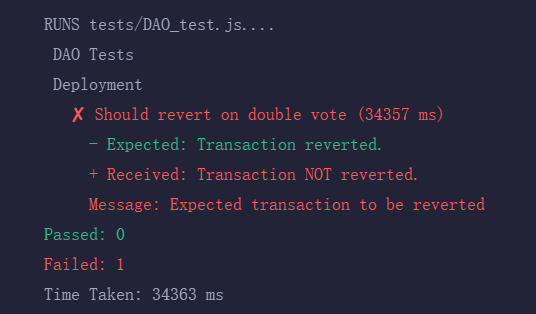




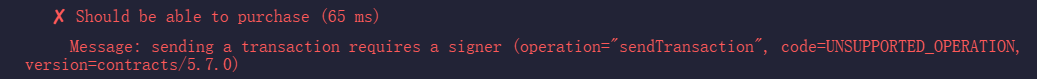
**DAO unit test**: see DAO\_test.js.

Should check trader address: the contract should be able to check the address of trader contract passed to constructor as parameter (passed)

Should revert on double vote: the contract should revert if a user votes twice (the transaction was rejected, but the judgement occurred before Metamask get the failure notice)



Should vote: the contract should be able to collect users’ votes, take actions according to the voting result (successful until EndVote() in which purchase() is called and the dao is not a signer in this context)



**Conclusion**

a summary of the steps you took to accomplish your assignment and unrealized profit and loss strategy using pre-defined assumptions. What have you learned?

**References**

[1] SimpleDAO example: <https://github.com/partylikeits1983/simpleDAO.git>

[2] Contract Wizard of OpenZeppelin: <https://docs.openzeppelin.com/contracts/5.x/wizard>

[3] CryptoZombies solidity courses: <https://cryptozombies.io/en/course>

[4] Chainlink’s datafeed blocks for oracle:

https://docs.chain.link/data-feeds/price-feeds/addresses/?network=ethereum&page=1