```
Running 1 test for test/ReentrancyAttack.t.sol:ReentrancyAttackTest
[PASS] testAttackFails() (gas: 556163)
Logs:
  EVE balance 1000000000000000000
  etherStore balance 2000000000000000000
  attack SC balance 0

Traces:
  [556163] ReentrancyAttackTest::testAttackFails()
    ├─ [198726] → new EtherStore@0x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f
    │   └─ ← 882 bytes of code
    ├─ [0] VM::deal(0x000000000000000000000000000000000000A11cE, 1000000000000000000 [1e18])
    │   └─ ← ()
    ├─ [0] VM::prank(0x000000000000000000000000000000000000A11cE)
    │   └─ ← ()
    ├─ [22434] EtherStore::deposit{value: 1000000000000000000}()
    │   └─ ← ()
    ├─ [0] VM::deal(0x000000000000000000000000000000000000B0b, 1000000000000000000 [1e18])
    │   └─ ← ()
    ├─ [0] VM::prank(0x000000000000000000000000000000000000B0b)
    │   └─ ← ()
    ├─ [22434] EtherStore::deposit{value: 1000000000000000000}()
    │   └─ ← ()
    ├─ [131908] → new Attack@0x2e234DAe75C793f67A35089C9d99245E1C58470b
    │   └─ ← 547 bytes of code
    ├─ [0] VM::deal(0x000000000000000000000000000000000000eFE, 1000000000000000000 [1e18])
    │   └─ ← ()
    ├─ [0] VM::prank(0x000000000000000000000000000000000000eFE)
    │   └─ ← ()
    ├─ [0] VM::expectRevert()
    │   └─ ← ()
    ├─ [39132] Attack::attack{value: 1000000000000000000}()
    │   ├─ [22434] EtherStore::deposit{value: 1000000000000000000}()
    │   │   └─ ← ()
    │   ├─ [8873] EtherStore::withdraw()
    │   │   ├─ [1168] Attack::fallback{value: 1000000000000000000}()
    │   │   │   ├─ [391] EtherStore::withdraw()
    │   │   │   │   └─ ← "REENTRANCY"
    │   │   │   └─ ← "REENTRANCY"
    │   │   └─ ← "Failed to send Ether"
    │   └─ ← "Failed to send Ether"
    ├─ [0] console::log(EVE balance, 1000000000000000000 [1e18]) [staticcall]
    │   └─ ← ()
    ├─ [0] console::log(etherStore balance, 2000000000000000000 [2e18]) [staticcall]
    │   └─ ← ()
    ├─ [0] console::log(attack SC balance, 0) [staticcall]
    │   └─ ← ()
    └─ ← ()

Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.03ms
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
```

```
    function testAttack() public {
        // 1. Deploy EtherStore
        etherStore = new EtherStore();

        // 2. Deposit 1 Ether each from Account 1 (Alice) and Account 2 (Bob) into
EtherStohoax(ALICE, 1 ether);
        etherStore.deposit{value: 1 ether}();
        assertEq(address(ALICE).balance, 0);

        hoax(BOB, 1 ether);
        etherStore.deposit{value: 1 ether}();
        assertEq(address(BOB).balance, 0);

        // 3. Deploy Attack with address of EtherStore
        attackContract = new Attack(address(etherStore));

        // 4. Call Attack.attack sending 1 ether (using Account 3 (Eve)).
        //     You will get 3 Ethers back (2 Ether stolen from Alice and Bob,
        //     plus 1 Ether sent from this contract).
        hoax(EVE, 1 ether);
        attackContract.attack{value: 1 ether}();
        assertEq(address(EVE).balance, 0);
        assertEq(address(etherStore).balance, 0);
        assertEq(address(attackContract).balance, 3 ether);
    }
```

```
function testAttackFails() public {
    // 1. Deploy EtherStore
    etherStore = new EtherStore();

    // 2. Deposit 1 Ether each from Account 1 (Alice) and Account 2 (Bob) into EtherStore
    hoax(ALICE, 1 ether);
    etherStore.deposit{value: 1 ether}();
    assertEq(address(ALICE).balance, 0);

    hoax(BOB, 1 ether);
    etherStore.deposit{value: 1 ether}();
    assertEq(address(BOB).balance, 0);

    // 3. Deploy Attack with address of EtherStore
    attackContract = new Attack(address(etherStore));

    // 4. Call Attack.attack sending 1 ether (using Account 3 (Eve)).
    //     You will not get 3 Ethers back (2 Ether stolen from Alice and Bob,
    //     plus 1 Ether sent from this contract). Instead the `withdraw` function
    //     would fail on second attempt & hence, the entire `attack` function would roll back although
    //     `deposit` works.
    hoax(EVE, 1 ether);
    vm.expectRevert();
    attackContract.attack{value: 1 ether}();
    console2.log("EVE balance", address(EVE).balance);
    console2.log("etherStore balance", address(etherStore).balance);
    console2.log("attack SC balance", address(attackContract).balance);
    // assertEq(address(EVE).balance, 1 ether);
    // assertEq(address(etherStore).balance, 2 ether);
    // assertEq(address(attackContract).balance, 0);
}
```