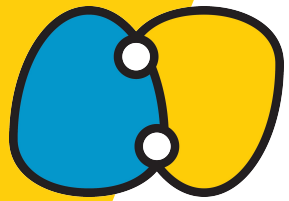
runtime verification



Getting your security under kontrol

Полина Толмач
Runtime Verification

Кто я?

runtime verification

- Tech lead @ **Runtime Verification**
- ex engineer @ **MakerDAO**
- PhD @ **NTU**, Singapore

Кто такие Runtime Verification?

runtime verification

Runtime Verification — это технологический стартап, создающий инструменты и сервисы **формальной верификации** для аэрокосмической, автомобильной и блокчейн индустрий.

runtimeverification.com

Наш срез безопасности

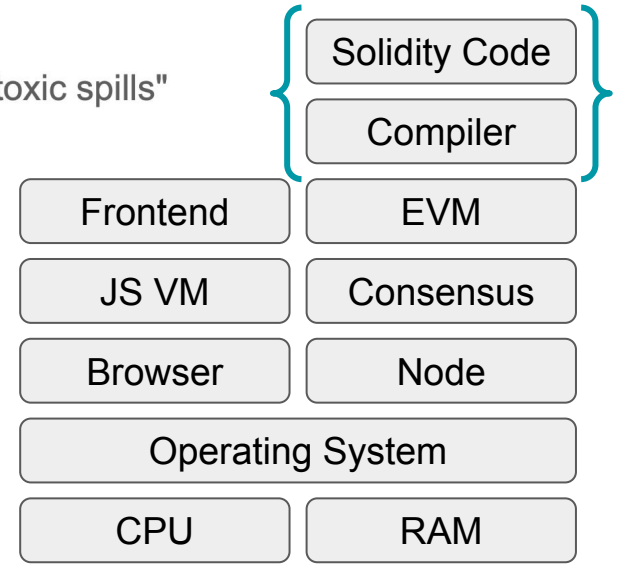
runtime verification

security

/sɪˈkjʊərɪti,sɪˈkjɔːrɪti/

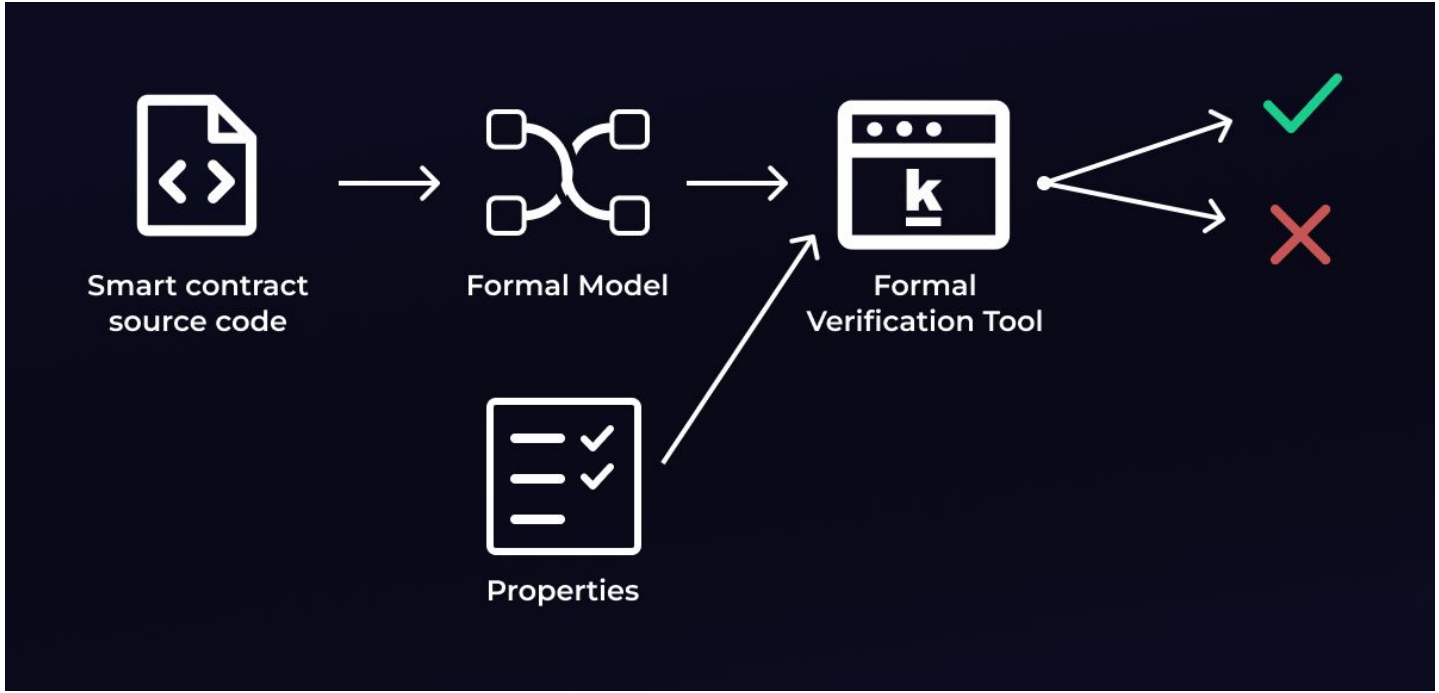
noun

1. the state of being free from danger or threat.
   "the system is designed to provide maximum **security against** toxic spills"



Что такое формальная верификация?

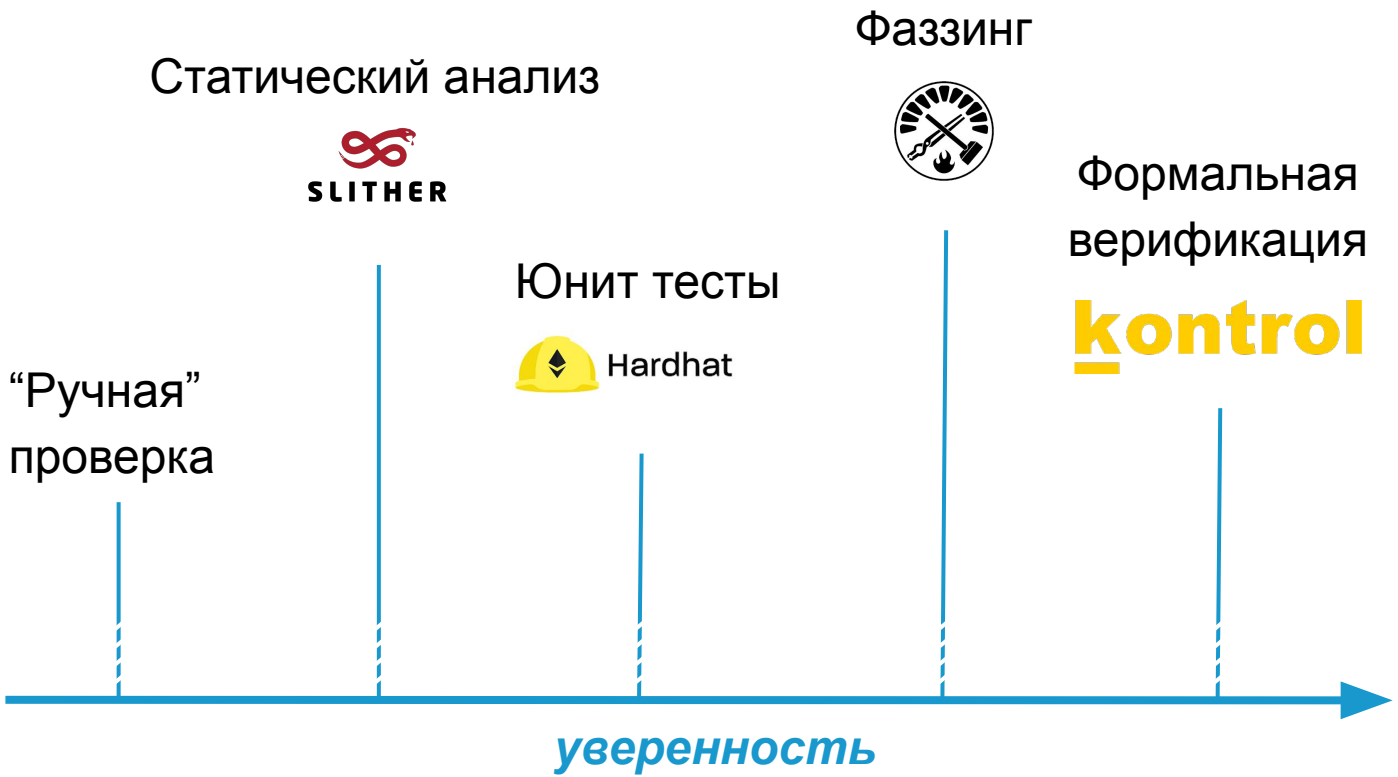
runtime verification



Источник: https://www.auditwizard.io/blog/securing-smart-contracts-with-formal-verification-tools

Существующие техники

runtime verification



runtime verification

```
/// @dev Equivalent to `(x * y) / WAD` rounded down.
function mulWad(uint256 x, uint256 y) internal pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if mul(y, gt(x, div(not(0), y))) {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xffffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`.
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

Источник: https://github.com/Vectorized/solady/blob/main/src/utils/FixedPointMathLib.sol

runtime verification

```
/// @author Modified from Solady
(https://github.com/vectorized/solady/blob/main/src/utils/FixedPointMathLib.sol)
/// @author Modified from Solmate
(https://github.com/transmissions11/solmate/blob/main/src/utils/FixedPointMathLib.sol)


/// @dev The scalar of ETH and most ERC20s.
uint256 internal constant WAD = 1e18;

/// @dev Equivalent to `(x * y) / WAD` rounded down.
function mulWad(uint256 x, uint256 y) public pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xfffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

- *Ограниченная поддержка статистичекими анализаторами*
- *Отсутствие встроенных проверок компилятора*
- *Нетипичная проблема*
- *Плохая читаемость*

Юнит тесты

runtime verification

скрытый loop

```
function test_foo(bytes memory data) public pure {
    assert(foo(data));
}
```

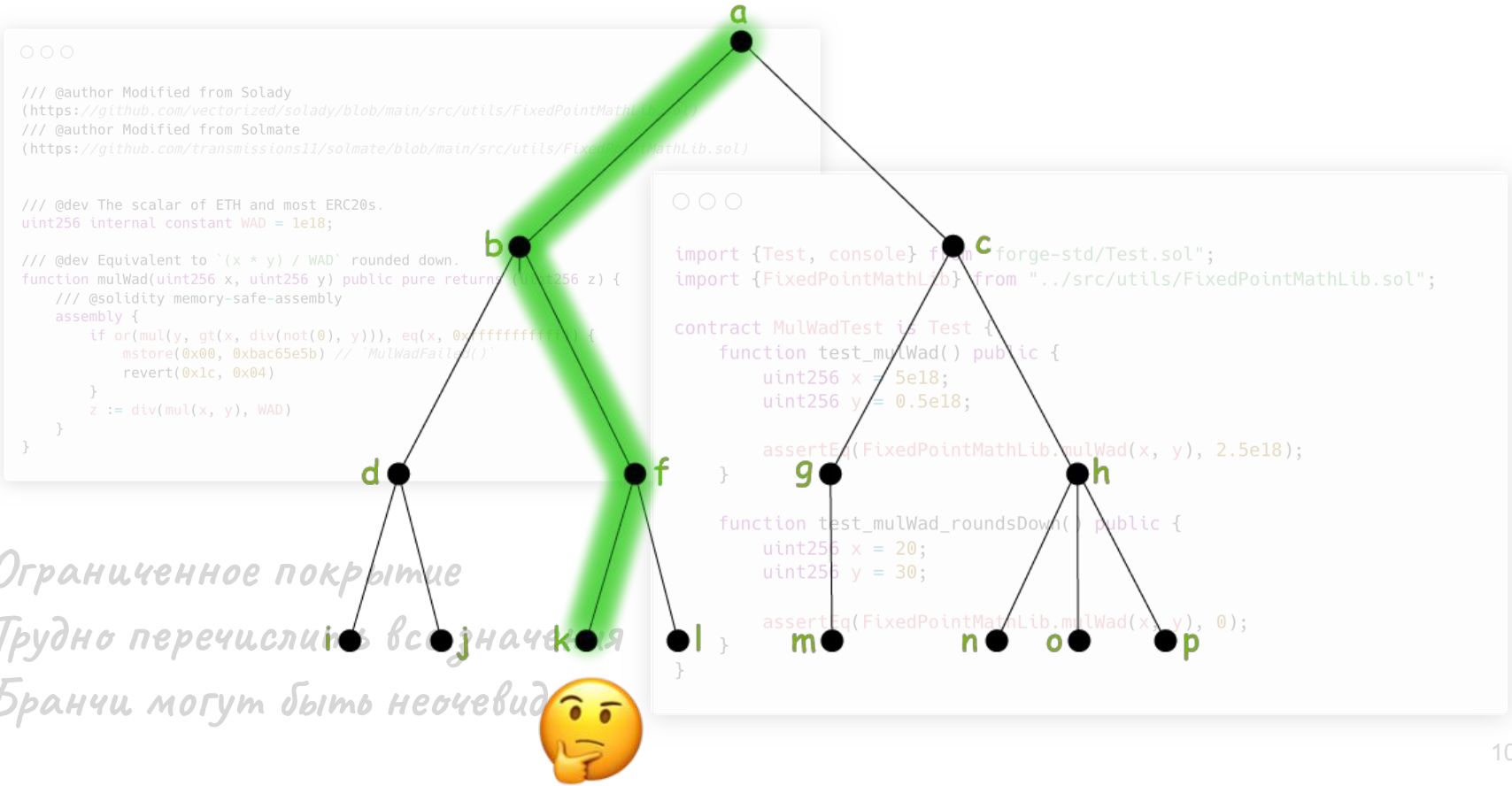
```
/// @author Modified from Solady
(https://github.com/vectorized/solady/blob/main/src/utils/FixedPointMathLib.sol)
/// @author Modified from Solmate
(https://github.com/transmissions11/solmate/blob/main/src/utils/FixedPointMathLib.sol)


/// @dev The scalar of ETH and most ERC20s.
uint256 internal constant WAD = 1e18;

/// @dev Equivalent to `(x * y) / WAD` rounded down.
function mulWad(uint256 x, uint256 y) public pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xfffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

```
import {Test, console} from "forge-std/Test.sol";
import {FixedPointMathLib} from "../src/utils/FixedPointMathLib.sol";

contract MulWadTest is Test {
    function test_mulWad() public {
        uint256 x = 5e18;
        uint256 y = 0.5e18;

        assertEq(FixedPointMathLib.mulWad(x, y), 2.5e18);
    }

    function test_mulWad_roundsDown() public {
        uint256 x = 20;
        uint256 y = 30;

        assertEq(FixedPointMathLib.mulWad(x, y), 0);
    }
}
```

- Ограниченное покрытие
- Трудно перечислить все значения
- Бранчи могут быть неочевидны

Юнит тесты

runtime verification



- Ограниченное покрытие
- Трудно перечислить все значения
- Ветви могут быть неочевидны

Фаззинг тесты

runtime verification

```
function mulWad(uint256 x, uint256 y) public pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xfffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`.
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

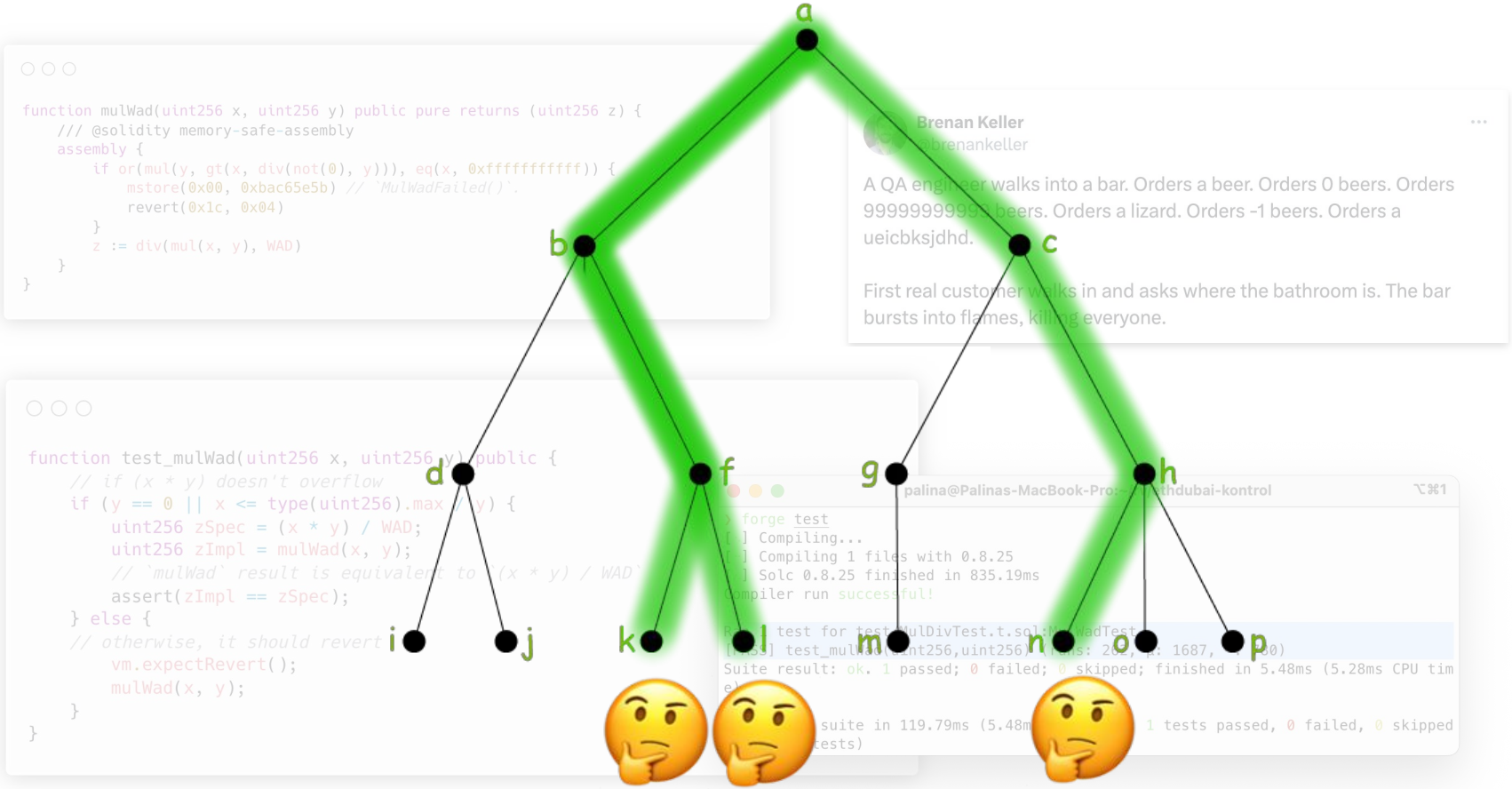
First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

```
function test_mulWad(uint256 x, uint256 y) public {
    // if (x * y) doesn't overflow
    if (y == 0 || x <= type(uint256).max / y) {
        uint256 zSpec = (x * y) / WAD;
        uint256 zImpl = mulWad(x, y);
        // `mulWad` result is equivalent to `(x * y) / WAD`
        assert(zImpl == zSpec);
    } else {
    // otherwise, it should revert
        vm.expectRevert();
        mulWad(x, y);
    }
}
```

```
palina@Palinas-MacBook-Pro:~/rv/ethdubai-kontrol  ⌥⌘1
❯ forge test
[⠊] Compiling...
[⠒] Compiling 1 files with 0.8.25
[⠢] Solc 0.8.25 finished in 835.19ms
Compiler run successful!

Ran 1 test for test/MulDivTest.t.sol:MulWadTest
[PASS] test_mulWad(uint256,uint256) (runs: 262, μ: 1687, ~: 780)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 5.48ms (5.28ms CPU time)

Ran 1 test suite in 119.79ms (5.48ms CPU time): 1 tests passed, 0 failed, 0 skipped (1 total tests)
```

Фаззинг тесты

runtime verification



Kontrol тесты

runtime verification

```
> kontrol build
> kontrol prove
```

```
function mulWad(uint256 x, uint256 y) public pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xfffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`.
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

```
function test_mulWad(uint256 x, uint256 y) public {
    // if (x * y) doesn't overflow
    if (y == 0 || x <= type(uint256).max / y) {
        uint256 zSpec = (x * y) / WAD;
        uint256 zImpl = mulWad(x, y);
        // `mulWad` result is equivalent to `(x * y) / WAD`
        assert(zImpl == zSpec);
    } else {
    // otherwise, it should revert
        vm.expectRevert();
        mulWad(x, y);
    }
}
```

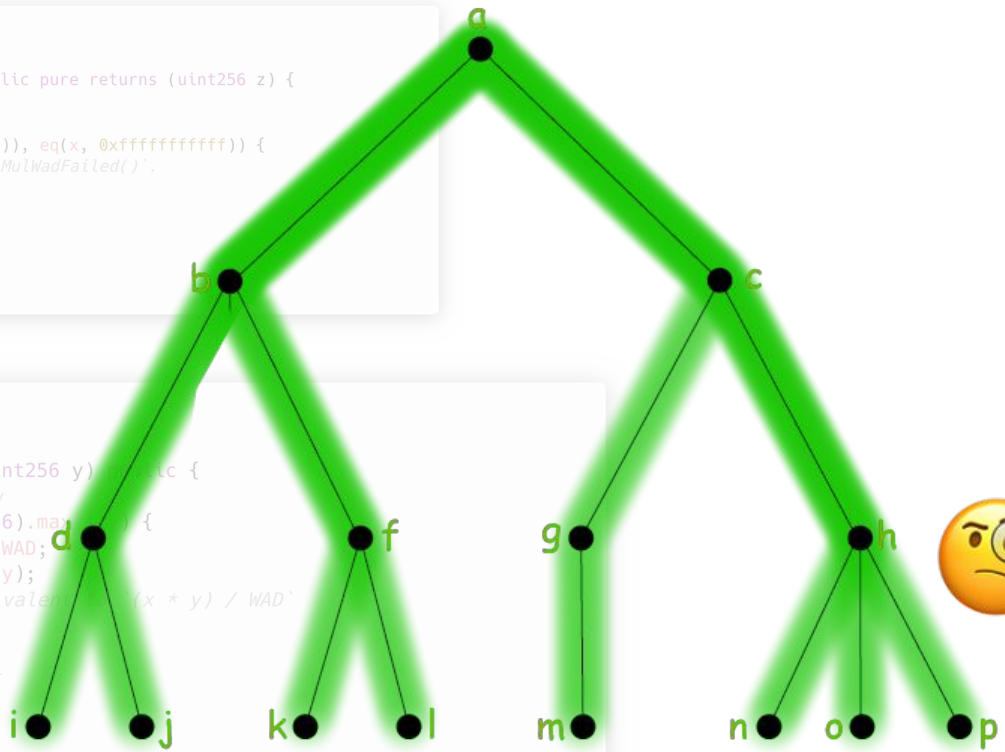
```
palina@Palinas-MacBook-Pro:~/rv/ethdubai-kontrol ⌥⌘1
PROOF FAILED: test%MulWadTest.test_mulWad(uint256,uint256):0
time: 3m 27s
5 Failure nodes. (4 pending and 1 failing)

Pending nodes: [24, 25, 26, 27]

Failing nodes:

  Node id: 23
  Failure reason:
    Matching failed.
    The remaining implication is:
    { VV1_y_114b9705:Int #Equals 0 }
    #And { VV0_x_114b9705:Int #Equals 17592186044415 }
    #And { true #Equals 0 <=Int CALLER_ID:Int }
    #And { true #Equals 0 <=Int ORIGIN_ID:Int }
    #And { true #Equals 0 <=Int NUMBER_CELL:Int }
    #And { true #Equals 0 <=Int TIMESTAMP_CELL:Int }
    #And #Not ( { CALLER_ID:Int #Equals 645326474426547203313410069153905908525362434349 } )
    #And #Not ( { ORIGIN_ID:Int #Equals 645326474426547203313410069153905908525362434349 } )
    #And #Not ( { CONTRACT_ID:Int #Equals 645326474426547203313410069153905908525362434349 } )
    #And { true #Equals CALLER_ID:Int <Int pow160 }
    #And { true #Equals ORIGIN_ID:Int <Int pow160 }
    #And { true #Equals NUMBER_CELL:Int <=Int maxSInt256 }
    #And { true #Equals TIMESTAMP_CELL:Int <Int pow256 }
    #And { false #Equals 0 <Int CALLER_ID:Int andBool CALLER_ID:Int <=Int 9 }
    #And { false #Equals 0 <Int ORIGIN_ID:Int andBool ORIGIN_ID:Int <=Int 9 } #Implies { true #Equals foundry_success (
... statusCode: EVMC_REVERT , failed: #lookup ( .Map , 463080223264950070279727286779179148927297929992997458304755966871808015073 28 ) , revertExpected: false , opcodeExpected: false , recordEventExpected: false , eventExpected: false ) }
  Path condition:
    { true #Equals VV1_y_114b9705:Int ==Int 0 }
#And { true #Equals notBool notBool VV0_x_114b9705:Int ==Int 17592186044415 }
  Model:
    ORIGIN_ID = 10
    VV1_y_114b9705 = 0
    CALLER_ID = 10
    NUMBER_CELL = 0
    VV0_x_114b9705 = 17592186044415
    TIMESTAMP_CELL = 0
    CONTRACT_ID = 2
```

Kontrol тесты

runtime verification

```
function mulWad(uint256 x, uint256 y) public pure returns (uint256 z) {
    /// @solidity memory-safe-assembly
    assembly {
        if or(mul(y, gt(x, div(not(0), y))), eq(x, 0xfffffffffff)) {
            mstore(0x00, 0xbac65e5b) // `MulWadFailed()`.
            revert(0x1c, 0x04)
        }
        z := div(mul(x, y), WAD)
    }
}
```

```
function test_mulWad(uint256 x, uint256 y) public pure {
    // if (x * y) doesn't overflow
    if (y == 0 || x <= type(uint256).max / y) {
        uint256 zSpec = (x * y) / WAD;
        uint256 zImpl = mulWad(x, y);
        // `mulWad` result is equivalent to `(x * y) / WAD`
        assert(zImpl == zSpec);
    } else {
    // otherwise, it should revert
        vm.expectRevert();
        mulWad(x, y);
    }
}
```

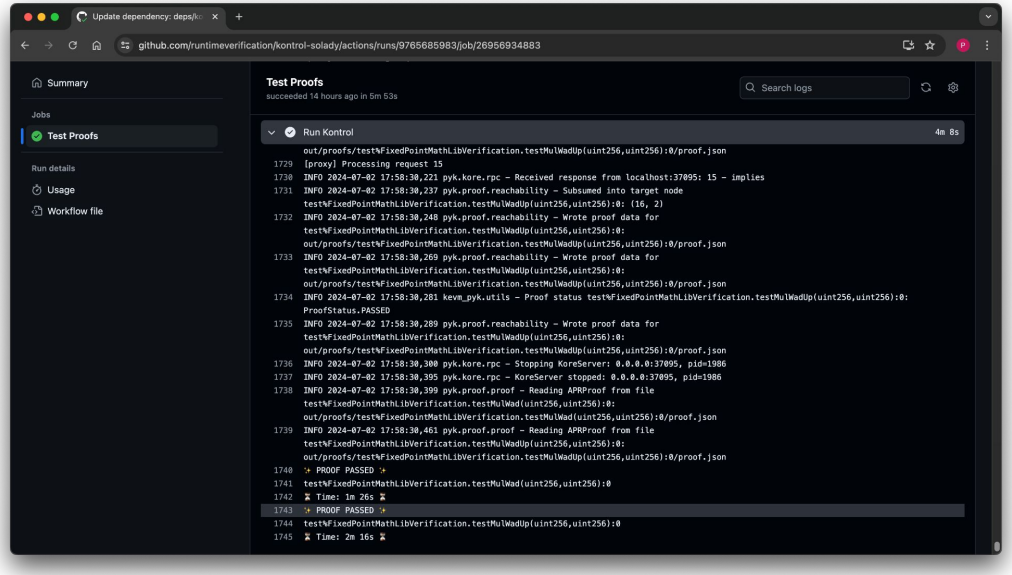


kontrol

Kontrol your pipeline

runtime verification

- Kontrol тесты можно интегрировать в CI



https://github.com/runtimeverification/kontrol-solady/tree/master/.github

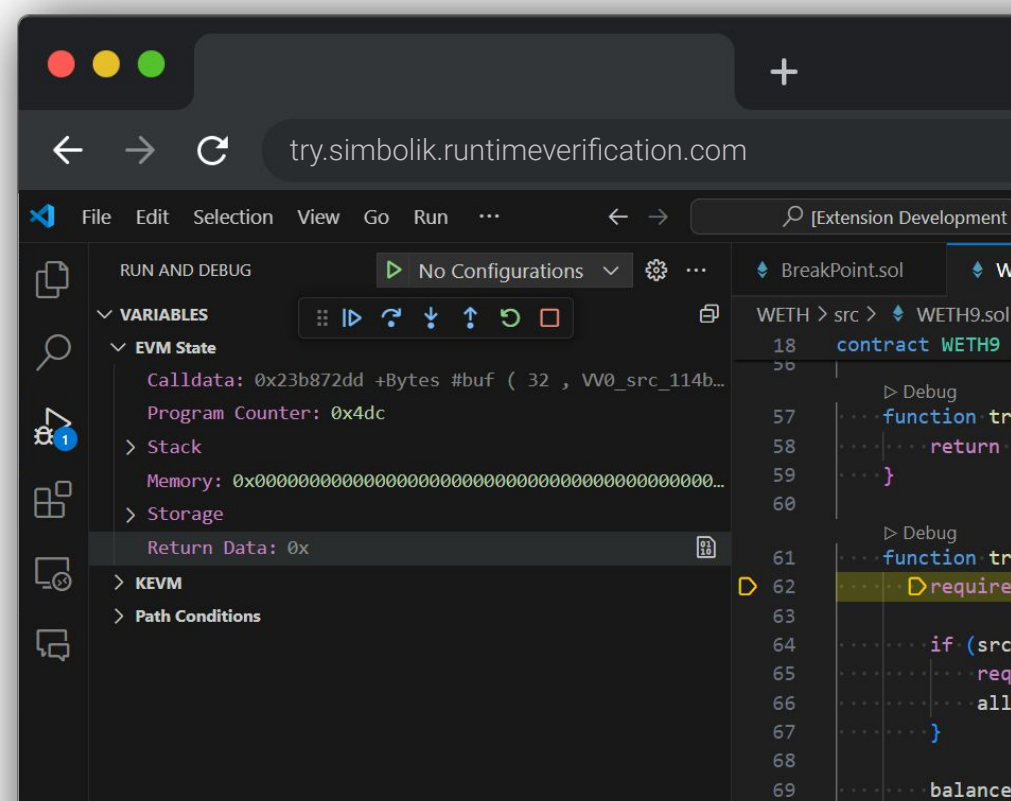
github.com/runtimeverification/install-kontrol

github.com/runtimeverification/kontrol

Что еще?

runtime verification

simbolik.runtimeverification.com



runtime verification

Stay up to date

 @rv_inc

 @palinatolmach

 https://discord.com/invite/CurfmXNtbN

 https://docs.runtimeverification.com/kontrol

 https://github.com/runtimeverification/kontrol

 **@rv_kontrol**