Preventing Hacks with Security Rules Written in Solidity

```
for oracle price deviations
assertionOraclePrice() external {
orkPreState();
256 prePrice = oracle.price();
orkPostState();
256 postPrice = oracle.price();
256 deviation = (((postPrice > prePric
prePrice : prePrice - postPrice) * 10(

ire(deviation <= 10, "Price deviation
```

Odysseas Lamtzidis, Founder, Phylax Systems

November 18th, 2025, Solidity Summit

Solidity

Ethereum

Ethereum introduced Solidity for development

Foundry, Halmos

Expanded Testing

dapptools

Introduced Solidity for testing

Phylax

Introduced Solidity for invariant enforcement at runtime

```
contract ForkTest is Test {
    // the identifiers of the forks
    uint256 mainnetFork;
    uint256 optimismFork;

    //Access variables from .env file via vm.envString("varname")
    //Replace ALCHEMY_KEY by your alchemy key or Etherscan key, change RPC url if need
    //inside your .env file e.g:
    //MAINNET_RPC_URL = 'https://eth-mainnet.g.alchemy.com/v2/ALCHEMY_KEY'
    //string MAINNET_RPC_URL = vm.envString("MAINNET_RPC_URL");
    //string OPTIMISM_RPC_URL = vm.envString("OPTIMISM_RPC_URL");

    // create two _different_ forks during setup
    function setUp() public {
        mainnetFork = vm.createFork(MAINNET_RPC_URL);
        optimismFork = vm.createFork(OPTIMISM_RPC_URL);
    }

    // demonstrate fork ids are unique
    function testForkIdDiffer() public {
        assert(mainnetFork != optimismFork);
    }
}
```

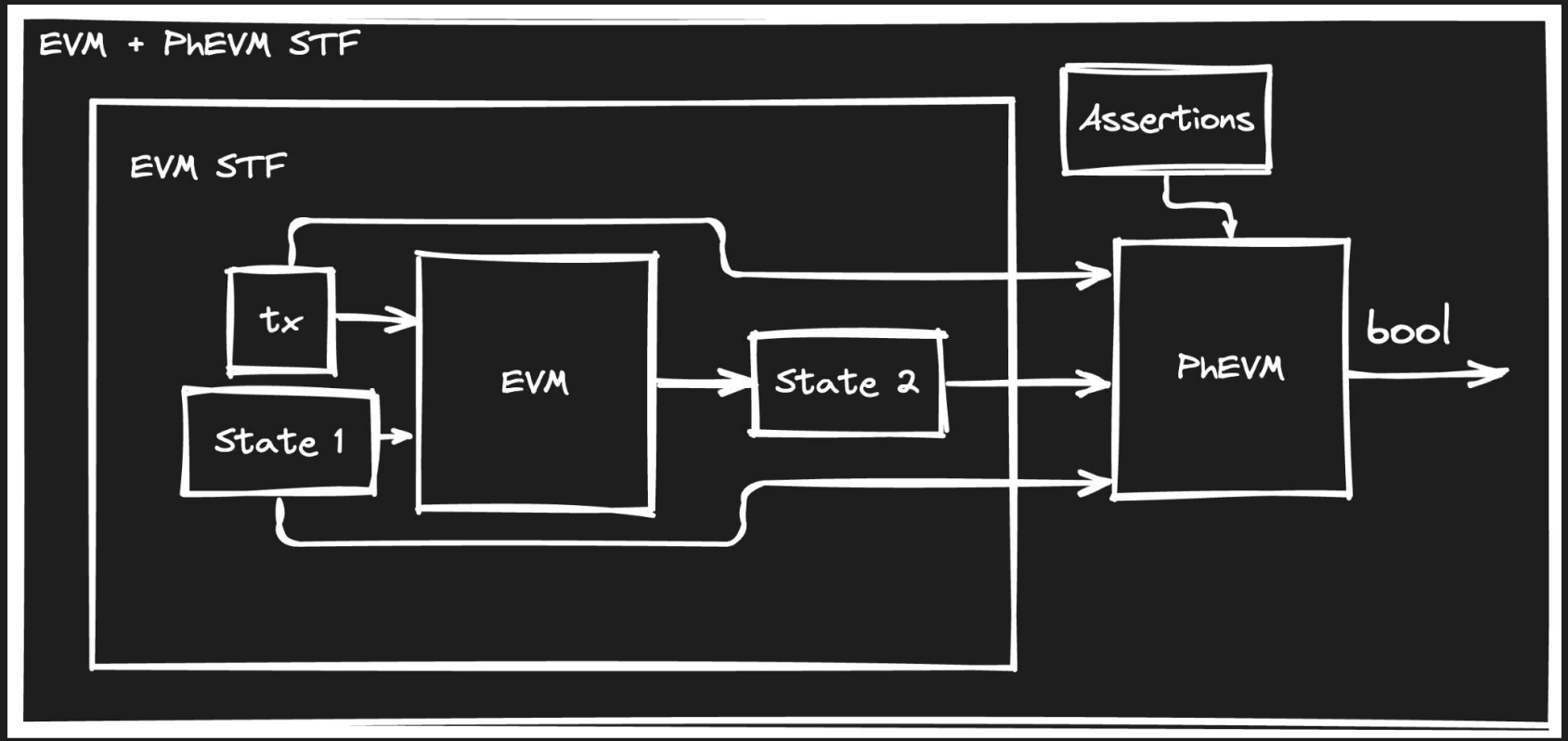
A fork test

```
forge test --fork-url
<your_rpc_url>
--fork-block-number 1
```

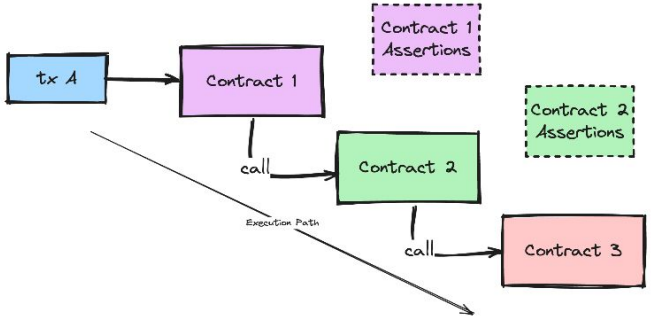
```
n=1
while [ $count -le 5 ]; do
  forge test --fork-url <your_rpc_url>
--fork-block-number n
  ((n++))
done
```

It’s a lame alert, but with great devex

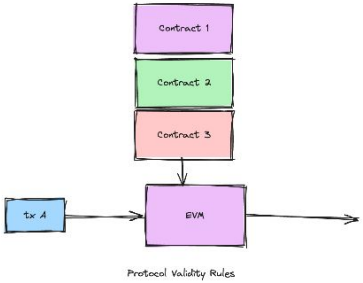
Alerts are lame, they are not preventative


EVM + PhEVM STF
EVM STF
tx
State 1
EVM
State 2
Assertions
PhEVM
bool

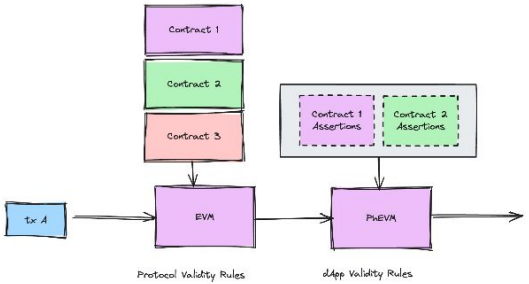
Different Transactions now have Different Validity Rules



BEFORE



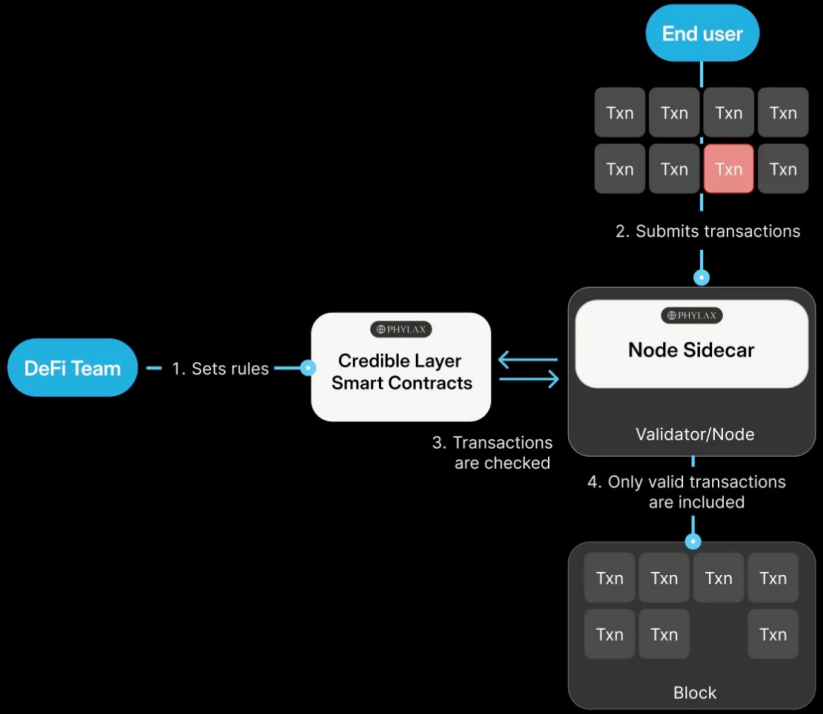
AFTER



Assertions are atomic, read-only EVM computations

They express the “what”
not the “how”

Lazily-executed fuzz test,
where the inputs are
generated by real users


End user
Txn
Txn
Txn
Txn
Txn
Txn
Txn
Txn
2. Submits transactions
PHYLAX
Node Sidecar
Validator/Node
DeFi Team
1. Sets rules
PHYLAX
Credible Layer
Smart Contracts
3. Transactions
are checked
4. Only valid transactions
are included
Txn
Txn
Txn
Txn
Txn
Txn
Txn
Block

If a user express an invariant as an Assertion, then we prevent hacks!

Balancer Hack

```
for (uint256 j = 0; j < uniquePoolIds.length; j++) {
    (address poolAddress, ) = IVault(vault).getPool(uniquePoolIds[j]);

    // Check rate before and after the batchSwap
    ph.forkPreCall(callInput.id);
    uint256 preRate = IRateProvider(poolAddress).getRate();

    ph.forkPostCall(callInput.id);
    uint256 postRate = IRateProvider(poolAddress).getRate();

    uint256 rateChangeMultiplier = (postRate * 1e18) / preRate;

    // Revert if rate changed by more than 3x
    require(
        rateChangeMultiplier ≤ MAX_RATE_CHANGE_MULTIPLIER &&
        rateChangeMultiplier ≥ MIN_RATE_CHANGE_MULTIPLIER,
        "BatchSwap: Extreme pool rate manipulation detected"
    );
}
```

Hereustic or Invariant?

Either works!

Let’s solve security so crypto can win.

Twitter: @odysseas_eth, Telegram: odyslam

to learn more check: **docs.phylax.systems**