Let’s Move2EVM

Secure Compilation of Move to EVM

Lorenzo Benetollo, Andreas Lackner, Markus Scherer, Matteo Maffei

USENIX Security '25

Background: Move vs EVM

Move: Statically typed, structured, ensures security at compile time.

EVM: Dynamic dispatch, dynamically computed jump destinations

Move's safety properties (e.g., resource linearity) are not enforced in EVM.

Challenges in Move-to-EVM Compilation

C1: Dropping resources

Resources should not be lost.

C2: Manipulating resources

Untrusted EVM contracts should not be able to modify resources.

C3: Forging references

Attackers should not create invalid references.

C4: Forging resources

Resources should not be maliciously created.

Challenges example

```
module 0x2::UntypedAttacker {

    use 0x1::Coin;

    fun oops(acc:&signer) {
      let coin = Coin::getCoin(acc);       // coin dropped
    }

    fun atm(acc:&signer) {
      let coin = Coin::get(acc);
      coin.value += 100;                   // coin manipulated
      putBack(acc, coin);
    }

    fun forgeRef() {
      let ref:&Coin = ... ;                // reference forged
      Coin::setZero(ref);
    }

    fun forgeCoin(acc:&signer) {
      let coin:Coin = { value: 1000 };     // coin forged;
      Coin::putBack(acc, coin);
    }
}
```

Proposed **Solution**: Inlined Reference Monitoring (**IRM**)

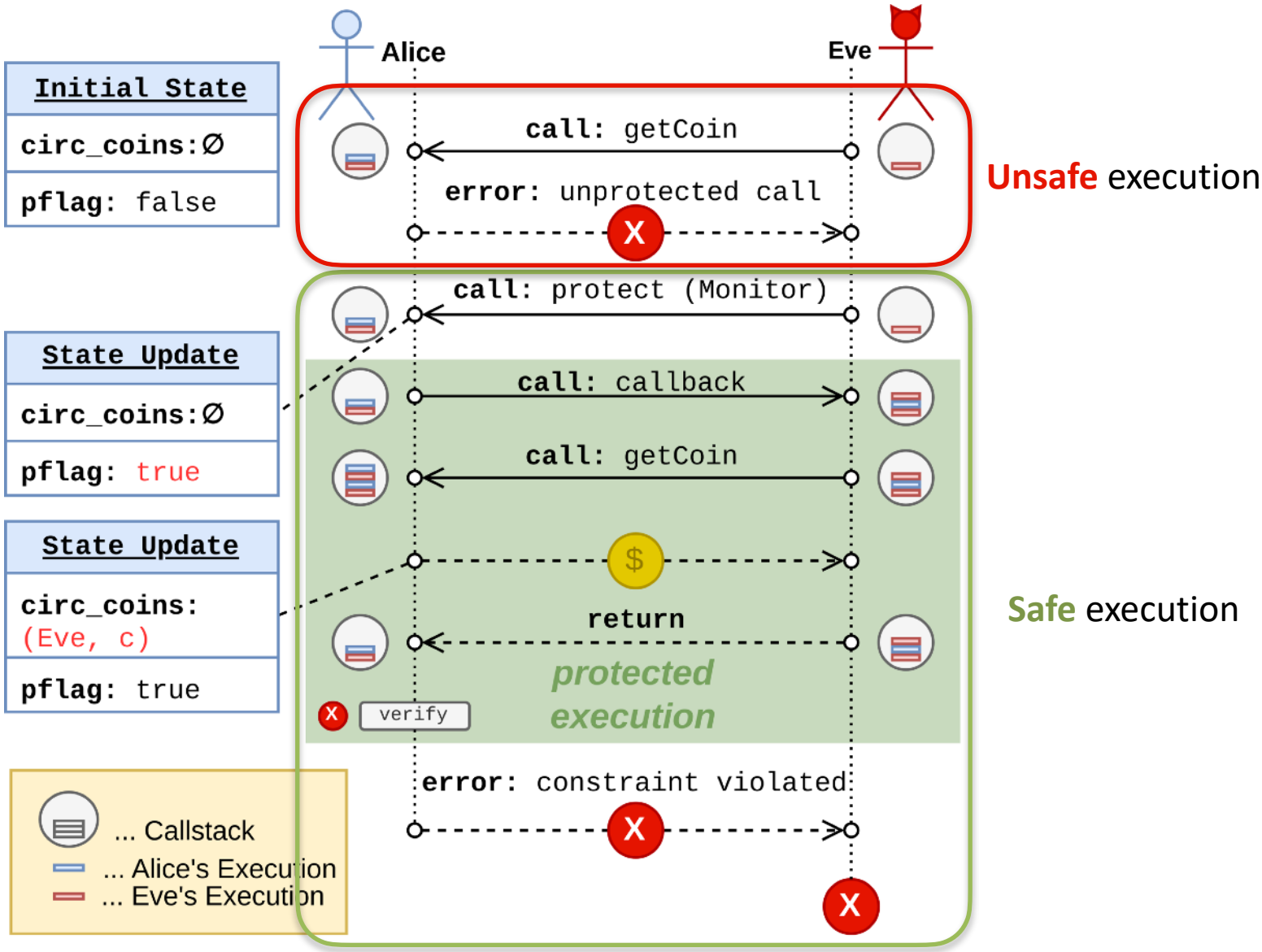
Insert runtime checks to **dynamically enforce** Move's safety guarantees.

Ensures security even in adversarial EVM environments.

Introduces a reasonable gas cost overhead

(approximately a 50% increase compared to the original Move To EVM compiler).

Protecting coins from getting lost



Implementation: IRM-based Protection Layer

- Integrated into the compiler at the **Yul** translation stage.
- Inserts dynamic checks into the **function dispatcher**.
- Uses **transient storage** (Solidity 0.8.26) for cost optimization.
- **Monitors** execution at function entry and exit points.

Experiments & Results

- Compared gas costs for ERC-20 implementation in Solidity vs Move.
- ERC-20 in Solidity: 47K gas, Move-to-EVM compiler: 89K gas.
- ERC-20 with IRM-based compiler: 64K gas (~33% increase).
- Most overhead comes from Move-to-EVM compilation, not security checks.