Blockchain Security Alliance | BEOSIN Blockchain Security × SUSS NiFT Node for Inclusive Fintech × BLOCKCHAIN ASSOCIATION SINGAPORE

Move – From a security perspective

"SECURITY IN WEB3" WORKSHOP

Contents

01 Move vs Solidity

02 Audit and Common Vulns

03 CTF(capture the flag)

BEOSIN
Blockchain Security

Move VS Solidity

Move VS Solidity

01 What Makes Move safe?

Move's Type System:

Type Safety:

No confusion on aliasing and mutability

Resource Safety:

Resources can only have two capabilities: key and store

Reference Safety:

there are no dangling references to both function locals and global storage

Move VS Solidity

02 Move VS Solidity

Asset Safety

Assets in move cannot be copied, lost and reused. This ensures the security of assets. .

Permission Safety

capabilities && function visibility: The ability to use copy, drop, store, and key to limit types. Use public, entry, friend to limit function visibility

Move VS Solidity

02 Move VS Solidity

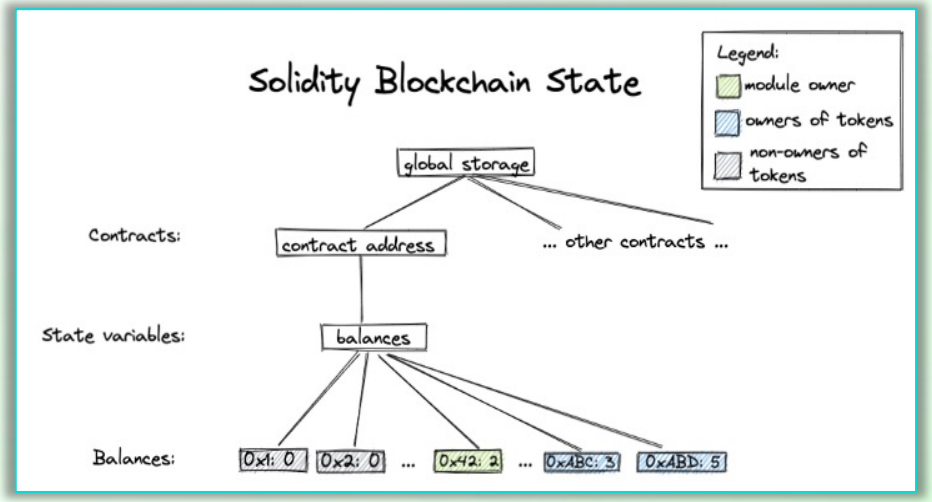
Storage Safety :

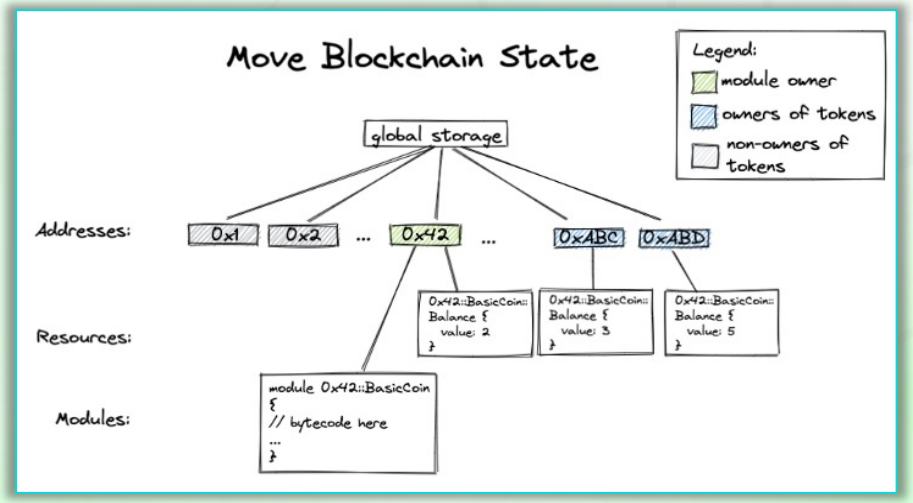
Solidity:

The balance of each address is stored in a state variable of type mapping(address => uint256), which is stored in the global storage of the specific smart contract.

Move:

Move's "global storage" is indexed by addresses, each address stores Move modules and Move resources, and resource storage is a type-to-value mapping.



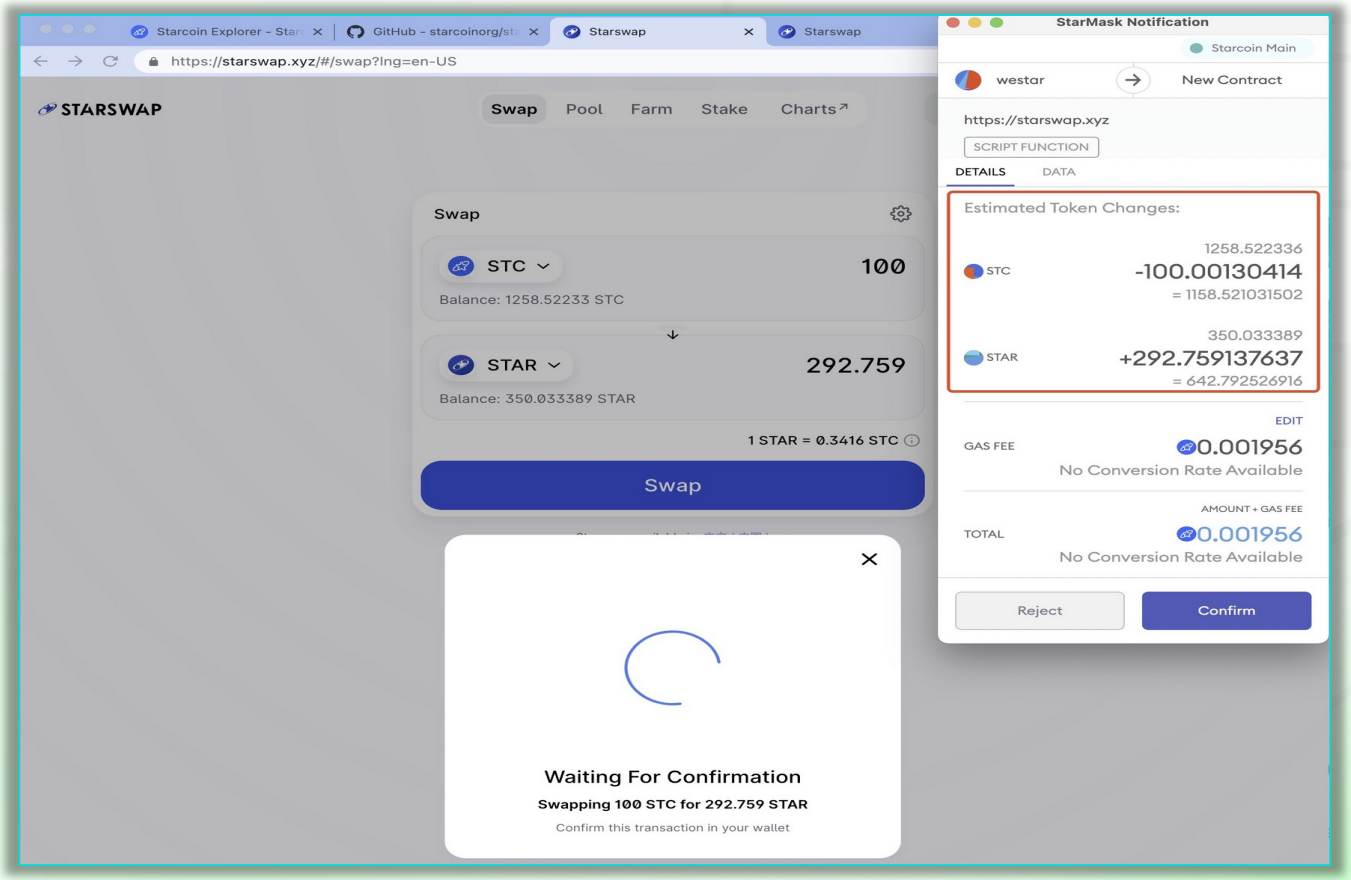


Move VS Solidity

02 Move VS Solidity

Static Safety:

No Dynamic call -> no more re-entrancy exploits



Move VS Solidity

02 Move VS Solidity

Verification Safety :

- Install Move Prover
- Write specifications

```
git clone https://github.com/move-language/move.git
cd move && ./scripts/dev_setup.sh -ypt
move prove -t TEST_NAME
```

```
struct Account has key {
        balance: u64
}

const MAX_BALANCE: u64 = 1000;

fun deposit(target: address, amount: u64) acquires Account {
    let account = borrow_global_mut<Account>(target);
    let new_balance = account.balance + amount;
    assert!(new_balance <= MAX_BALANCE, 0);
    account.balance = new_balance;
}
```

```
invariant forall a: address where exists<Account>(a):
    balance(a) <= MAX_BALANCE;
```

Move VS Solidity

02 Move VS Solidity

* Overflow/Underflow

```
#[test]
fun test_overflow() {
    let num: u8 = 255;
    num = num + 1;
    debug::print(&num);
}
```

```
VMError (if there is one): VMError {
|     major_status: ARITHMETIC_ERROR,
|     sub_status: None,
|     message: None,
|     exec_state: None
```

⚠ *Be careful about bitwise operation*

```
#[test]
fun test_overflow() {
    let num: u8 = 255;
    num = num << 2;
    debug::print(&num);
}
```


BEOSIN
Blockchain Security
Audit and
common vulns

Audit and common vulns

$whereis Audit

Static Analysis

- Code review
- Code query
- Formal verification

Dynamic Analysis

- Debugging
- Fuzzing
- Formal verification

Audit and common vulns

Steps to audit a Move Smart Contract

- Research the architecture, purpose and use of the platform
- Manual review of the smart contract to identify any logic issues

Audit and common vulns

Steps to audit a Move Smart Contract

- Test coverage review (aptos move test)

```
#[test]
public fun unit_test_add() {
  let a = 1;
  let b = 2;
  assert!((a + b) == 3, 100);
}

#[test_only]
public fun add_test(a: u64, b: u64):u64 {
  return a + b
}

#[test]
#[expected_failure(abort_code = 100)]
public fun unit_test_fail() {
  abort 100
}
```

Audit and common vulns

Steps to audit a Move Smart Contract

- Write Move Prover representations in Move Specification Language (MSL) for formal verification of important variables

- Smart contract deployment (Aptos Devnet) and on-chain testing of core functions (aptos-cli)

Audit and common vulns

Vulnerability Example (Aptos Liquid Swap)

What is missing? (hint: Oracle)

```
public fun get_reserves_size<X, Y, LP>(pool_addr: address): (u64, u64) acquires LiquidityPool
{
    ...
}
```

Audit and common vulns

Vulnerability Example (Aptos Liquid Swap)

What is missing?

```
public fun get_reserves_size<X, Y, LP>(pool_addr: address): (u64, u64) acquires LiquidityPool
{
  assert!(pool.locked == false, ERR_POOL_IS_LOCKED);
    ...
}
```

Exploit Example:
https://chainsecurity.com/curve-lp-oracle-manipulation-post-mortem/

Audit and common vulns

Vulnerability Example (Sui-AMM-swap)

```
    /// Swap Coin<X> for Coin<Y>
    /// Returns Coin<Y>
    public(friend) fun swap_out<X, Y>(
        global: &mut Global,
        coin_in: Coin<X>,
        coin_out_min: u64,
        ctx: &mut TxContext
    ): vector<u64> {
        assert!(coin::value<X>(&coin_in) > 0, ERR_ZERO_AMOUNT);

        if (is_order<X, Y>()) {
            let pool = get_mut_pool<X, Y>(global);
            let (coin_x_reserve, coin_y_reserve, _lp) = get_reserves_size(pool);
            assert!(coin_x_reserve > 0 && coin_y_reserve > 0, ERR_RESERVES_EMPTY);
            let coin_x_in = coin::value(&coin_in);

            let coin_x_fee = get_fee_to_fundation(coin_x_in);
            let coin_y_out = get_amount_out(
                coin_x_in,
                coin_x_reserve,
                coin_y_reserve,
            );
            assert!(
                coin_y_out >= coin_out_min,
                ERR_COIN_OUT_NUM_LESS_THAN_EXPECTED_MINIMUM
            );
```

Audit and common vulns

Vulnerability Example (Sui-AMM-swap)

Lack K value verification after swap:

The attacker can potentially exchange a small number of tokens for almost all tokens inside the Pool.

```
public(friend) fun swap_out<X, Y>(
    global: &mut Global,
    coin_in: Coin<X>,
    coin_out_min: u64,
    ctx: &mut TxContext
): vector<u64> {
    assert!(coin::value<X>(&coin_in) > 0, ERR_ZERO_AMOUNT);

    if (is_order<X, Y>()) {
        ......
        let (new_reserve_x, new_reserve_y, _lp) = get_reserves_size(pool);
        assert!(
            (coin_x_reserve as u128) * (coin_y_reserve as u128)
                < (new_reserve_x as u128) * (new_reserve_y as u128),
            14
        )
```


BEOSIN
Blockchain Security
CTF
(capture the flag)

Road to become an auditor?

```
/IntroCTF/Examples/IntroAudits/Examples/Challenge
```

cts
@gf_256

the ctf team to blockchain audit shop pipeline

* perfect blue --> zellic
* dice gang --> ottersec
* alles --> neodyme
* ppp / theori --> theori

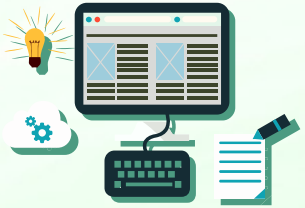
```
❯ cat example.txt
Paradigm CTF
https://www.paradigm.xyz

Move CTF
https://movectf.movebit.xyz
```

Move CTF

$cat sui.txt

Sui CTF 2022

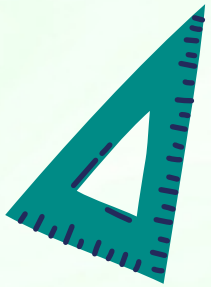
Hero Game



https://github.com/movebit/movectf-6

Move CTF

$cat hint



Exploit pseudo randomness on blockchain

About Beosin

Beosin is a leading global blockchain security company co-founded by several professors from world-renowned universities and there are 40+ PhDs in the team. It has offices in Singapore, UAE, Korea, Japan and other 10+ countries. With the mission of "Securing Blockchain Ecosystem", Beosin provides "All-in-One" blockchain security solution covering Smart Contract Audit, Risk Monitoring & Alert, KYT/AML, and Crypto Tracing. Beosin has already audited more than 3000 smart contracts including famous Web3 projects PancakeSwap, Uniswap, DAI, OKSwap and all of them are monitored by Beosin EagleEye. The KYT AML are serving 100+ institutions including Binance.

3,000+

Smart Contracts Audited

$500 Billion

Protected Assets

85,000+

Identified Code Vulnerabilities

1 Million+

Users

1 Billion+

Crypto Addresses Labeled

20+ Years

Cybersecurity Experience

150+ Members

40+ PhD Security Experts

Formal Verification
AI Data Analysis

Over 97% Accuracy

CONTACT US

- Website: www.beosin.com
- Email: contact@beosin.com
- Official Twitter: twitter.com/Beosin_com
- Alert Twitter: twitter.com/BeosinAlert
- Telegram: t.me/beosin
- Discord: discord.com/invite/B4QJxhStV4
- Linkedin: linkedin.com/company/beosin