SECURING BLOCKCHAIN ECOSYSTEM

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 What is Move

02 Important Concepts

03 Tutorial

04 Modules & Scripts

05 Move Blockchains

06 Standard Library

BEOSIN
Blockchain Security

What is Move

What is Move

01 What is Move?

Move is a secure, sandboxed and formally verified next-generation programming language whose first use case was the Diem blockchain (then called Libra, a project developed by the Facebook team). Move is a smart contract language made for manipulating digital assets and was originally designed with security in mind.

What is Move

02 Basic Structure of a Move Package

```
$ move new FirstMoveProject
```

```
a_move_package
├── Move.toml        (required)
├── sources          (required)
├── examples         (optional, test & dev mode)
├── scripts          (optional)
├── doc_templates    (optional)
└── tests            (optional, test mode)
```

- **Move.toml:** Configuration file, storing project information, dependencies, naming addresses, etc
- **sources:**Source code directory, used to store modules, scripts and other files
- **script :** Store the script file
- **test:** Store the test files

BEOSIN
Blockchain Security

Important Concepts

Important Concepts

01 Important concepts in Move

Structs : Struct is the only way to create custom type in Move.

* References and tuples cannot be stored as a field value of structs because they are not serializable

```
struct EventStore has key {
    swap_events: EventHandle<SwapEvent>,
}

struct SwapEvent has drop, store {
    trader: address,
    channel: u64,
    x_type_info: TypeInfo,
    y_type_info: TypeInfo,
    input_amount: u64,
    output_amount: u64,
}
```

Important Concepts

01 Important concepts in Move

Capabilities : copy, drop, store and key

```
struct EventStore has key {
    swap_events: EventHandle<SwapEvent>,
}

struct SwapEvent has drop, store {
    trader: address,
    channel: u64,
    x_type_info: TypeInfo,
    y_type_info: TypeInfo,
    input_amount: u64,
    output_amount: u64,
}
```

copy:
Allow to be copied.

drop:
Allow to be discarded (destroyed)

store:
Allow storage to global state

key:
Allow as key value

Important Concepts

01 Important concepts in Move

Resource : Resources are structures that can only have two capabilities: key and store

```
struct R has key { f: u64 }
// will not compile
fun ret_direct_resource_ref_bad(a: address): &R {
    borrow_global<R>(a) // error!
}
// also will not compile
fun ret_resource_field_ref_bad(a: address): &u64 {
    &borrow_global<R>(a).f // error!
}
```

Resources must be stored under the account
One account can only hold one resource at a time
No dangling references to global storage

Important Concepts

02 Challenge: Spot the bug!

$cat workshop.move

```
struct RoundBuffer<T: store + drop> has store {
    buffer: vector<T>,
    last_index: u64,
    entries: u64,
    max_capacity: u64,
}

public fun pop<T: store + drop>( rbuffer: &mut RoundBuffer<T>, value: T ) {
    let current_length = length(&rbuffer);
    let max_length = rbuffer.max_capacity;

    if (current_length == 0) {
        return;
    };

    let element_position = contains(rbuffer , value);

    if (element_position < max_length) {
        vector::remove<T>(rbuffer, element_position);
    }

}
```

BEOSIN
Blockchain Security

Tutorial

Tutorial

01 Tutorial for a publishing a simple coin in Aptos

Source code: https://pastebin.com/f3SAYkiD

Initialize :
Initializing a coin.

Minting:
Creating new coins.

Burning:
Deleting coins.

Registering:
Creating a CoinStore resource on an account for storing coins.

Transferring:
Withdrawing and depositing coins into CoinStore.

BEOSIN
Blockchain Security

Modules & Scripts

Modules & Scripts

Modules and Scripts

Modules

Module	Resource	function
Class	Object	function

Modules are stored in global storage itself.

Scripts

Scripts are executable entry points. They can call functions of published modules to update the global store. Scripts are temporary code snippets that are not published to the global store. (Func main)

Modules & Scripts

Interacting with on-chain program using Module

```
aptos move compile
aptos move publish
aptos move run --function-id 0xdeadbeef::solve::exp
```

```
module simpleswap::solve {
    use aptos_framework::coin;
    use std::signer::address_of;
    use ctfmovement::simple_coin::{Self, SimpleCoin, TestUSDC};
    use ctfmovement::swap;
    public entry fun exp(user: &signer) {
        let addr = address_of(user);
        coin::register<SimpleCoin>(user);
        simple_coin::claim_faucet(user, 100000000000000000);
        while(true) {
            let balance = coin::balance<SimpleCoin>(addr);
            if (balance > 10000000000) {
                break
            };
            let (_s, u) = swap::pool_reserves<SimpleCoin, TestUSDC>();
            let usdc_swap = coin::withdraw<TestUSDC>(user, u);
            let (out, reward) = swap::swap_exact_y_to_x_direct<SimpleCoin, TestUSDC>(usdc_swap);
            coin::deposit(addr, out);
            coin::deposit(addr, reward);
        };
        simple_coin::get_flag(user);
    }
}
```

Modules & Scripts

Interacting with on-chain program using Script

```
aptos move run-script --profile default --compiled-script-path
./build/{}/bytecode_scripts/{}.mv
```

```
script {
    use aptos_framework::coin;
    use std::signer::address_of;
    use ctfmovement::simple_coin::{Self, SimpleCoin, TestUSDC};
    use ctfmovement::swap;
    fun exp(user: &signer) {
        let addr = address_of(user);
        coin::register<SimpleCoin>(user);
        simple_coin::claim_faucet(user, 100000000000000000);
        while(true) {
            let balance = coin::balance<SimpleCoin>(addr);
            if (balance > 10000000000) {
                break
            };
            let (_s, u) = swap::pool_reserves<SimpleCoin, TestUSDC>();
            let usdc_swap = coin::withdraw<TestUSDC>(user, u);
            let (out, reward) = swap::swap_exact_y_to_x_direct<SimpleCoin, TestUSDC>(usdc_swap);
            coin::deposit(addr, out);
            coin::deposit(addr, reward);
        };
        simple_coin::get_flag(user);
    }
}
```

BEOSIN
Blockchain Security

Move Blockchains

Move Blockchains

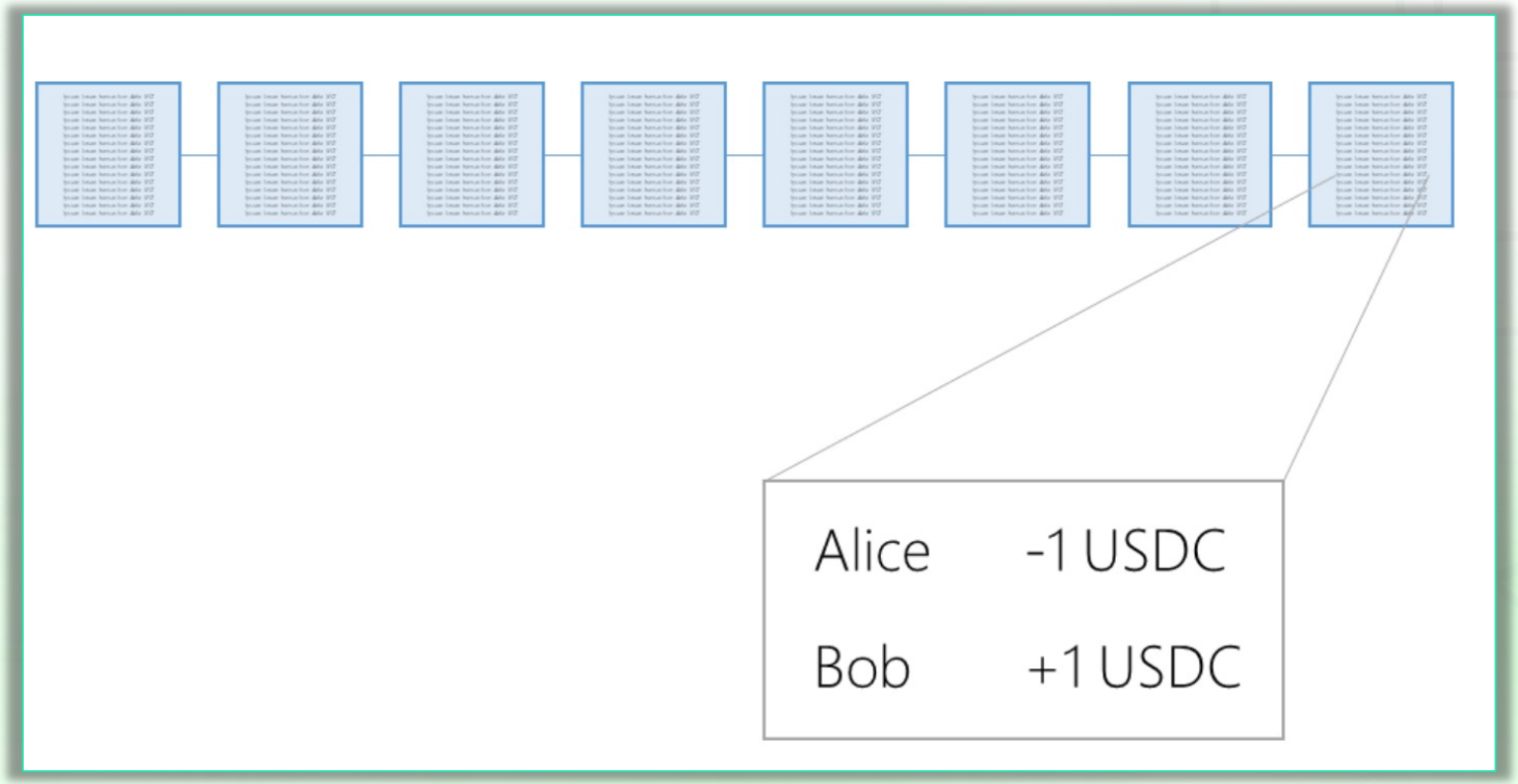
01 Building blockchains on Move

	APTOS	ETHEREUM	AVALANCHE	SOLANA	BINANCE	SUI
MAX TPS	160,000*	30	4,500	120,000	160	INFINTE
TIME TO FINALITY	<1S	78S	1.3–4S	2.34–46S	75S	<1S
CURRENT TPS	1,000 (TESTNET)	12.9	44	2,000	50	N/A
TICKER	APT	ETH	AVAX	SOL	BNB	SUI

Move Blockchains

02 Aptos Transaction visualization

Aptos is address-centric and it records the ledger as a blockchain



Move Blockchains

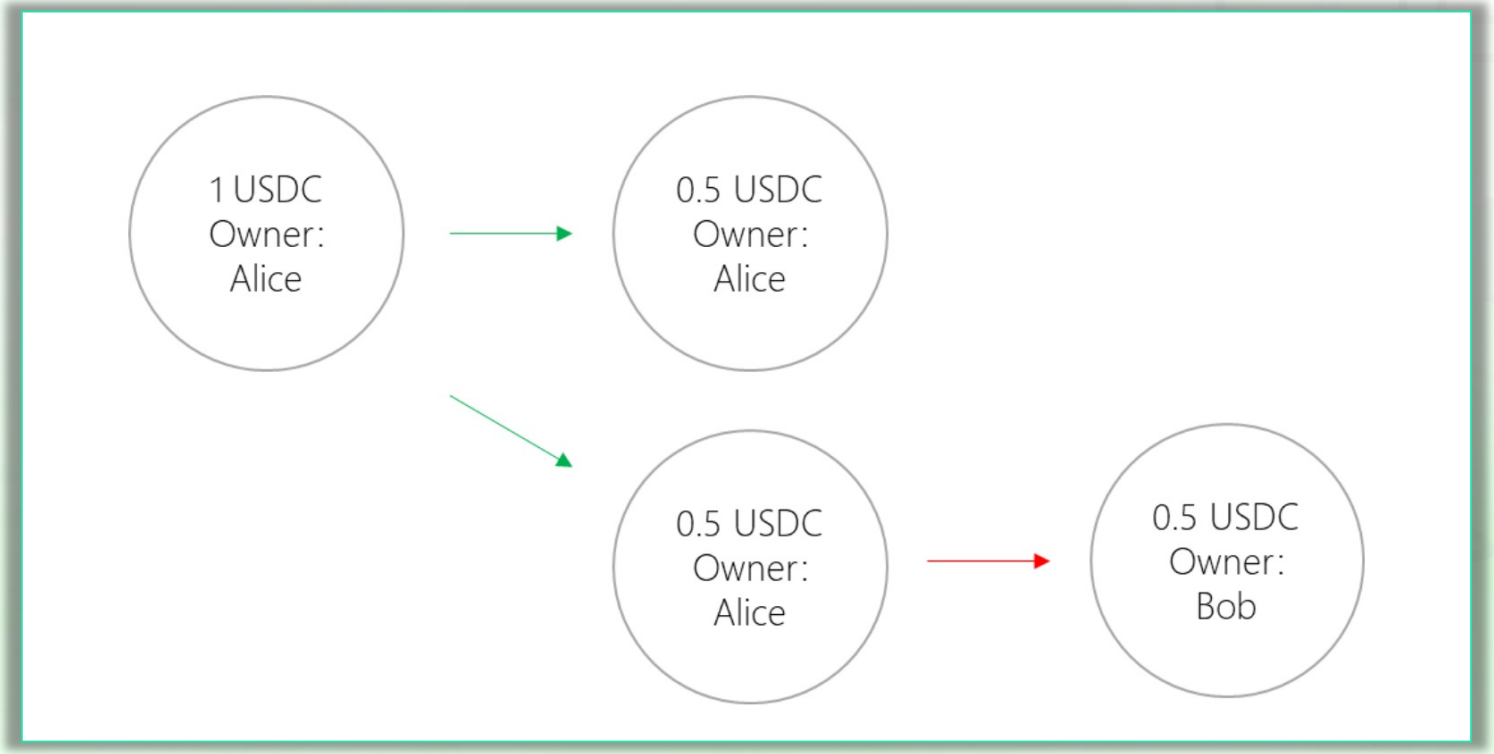
03 Aptos

In Aptos/core Move, any object with the key ability can be stored at an account (does not need UID). An Aptos smart contract can access object of types that it defines under any account at any point in its execution, even if the caller is not the account that is accessed. This is typically done through ***borrow_global***, borrowing a reference to an object type it defines.

Move Blockchains

04 Sui Transaction visualization

Sui is object-centric and it records the ledger as a DAG (Directed Acyclic Graph)



Move Blockchains

05 Sui

No Move global storage

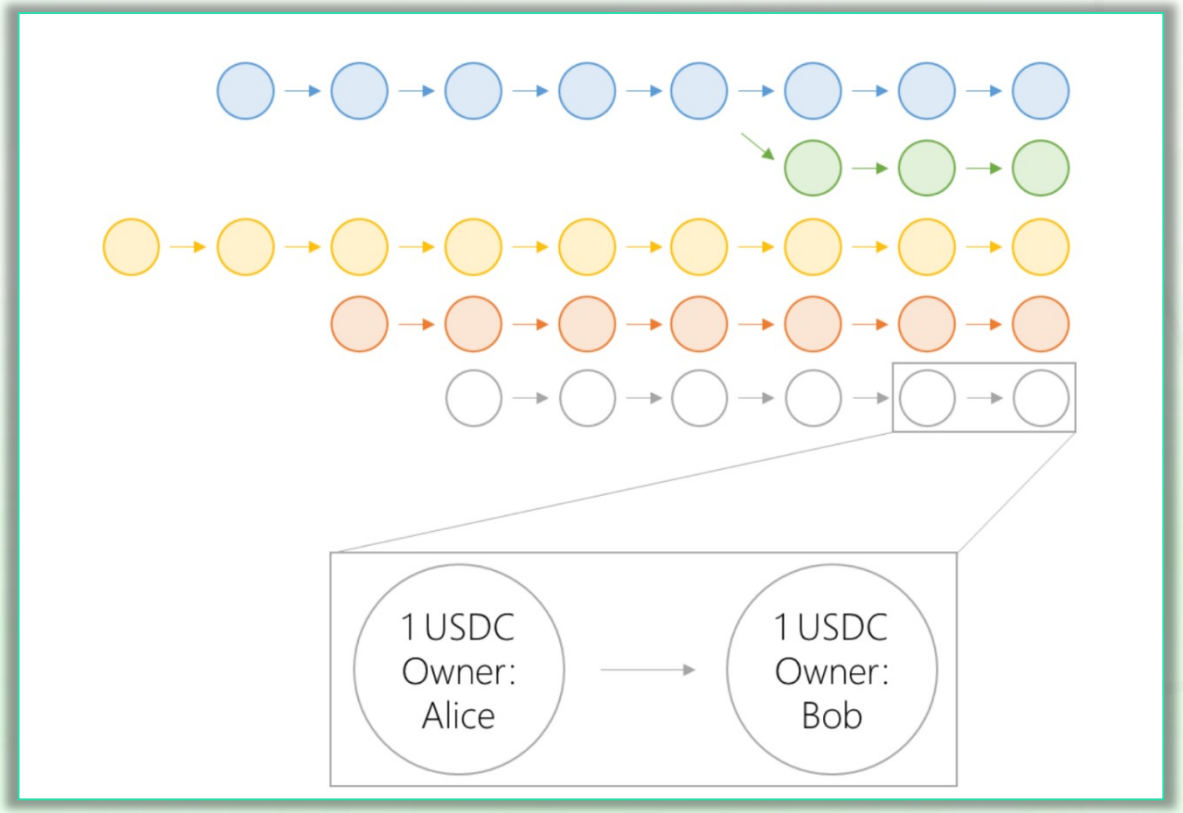
- X move_to
- X move_from
- X acquires

objects are passed to the runtime directly at the top level call and no external references can be borrowed during execution. To store objects, you use the transfer::transfer(object, address) function.

Move Blockchains

06 Innovations of Sui (DAG)

directed acyclic graph： each object has its own ledger and transactions don’t necessarily have to be in sequence

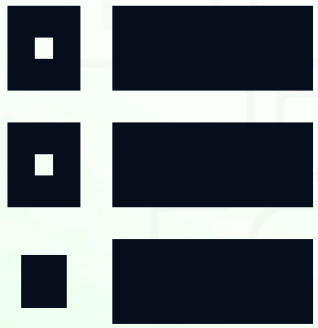


Move Blockchains

07 Innovations of Sui - Objects

Sui objects have the key ability (since they are stored on-chain) and a special first member of the struct named id with type UID. They are created through object::new(&mut TxContext) and destroyed through object::delete(UID)

A bytecode verifier ensures that any struct with the key ability has the special first member.



Move Blockchains

07 Innovations of Sui - Objects

Object States in Sui

- Inside a Move Contract
- Frozen/immutable (global read-only)
- Shared (global read-write)
- Owned by an address
- Owned by another object
- Wrapped

In core Move, the declaring contract can use types it declares from any address regardless of caller, whereas in Sui only the caller’s owned objects may be passed in

07 Innovations of Sui – Objects

- Sui objects with (key + store) capabilities can be freely transferred by account owner or the creating module. Objects with only key capability can only be transferred by the creating module.
- Frozen(immutable)/Shared objects can be passed to the smart contract by **anyone**

Move Blockchains

08 Innovations of Sui - TxContext

Aptos:

```
public entry fun exp(user: &signer) {
    let addr = address_of(user);
    coin::register<SimpleCoin>(user);
    simple_coin::claim_faucet(user, 100000000000000000);
    while(true) {
        let balance = coin::balance<SimpleCoin>(addr);
        if (balance > 10000000000) {
            break
```

Sui: X signer

```
public entry fun solve(lender: &mut FlashLender, ctx: &mut TxContext) {
    let (loan, receipt):(Coin<FLASH>, Receipt) = flash::loan(lender, 1000, ctx);
    flash::get_flag(lender, ctx);
    flash::repay(lender, loan);
    flash::check(lender, receipt);
}
```

Move Blockchains

08 Innovations of Sui - TxContext

Aptos:

```
fun init_module(swap_admin: &signer) {
    let swap_admin_addr = signer::address_of(swap_admin);
    assert!(swap_admin_addr == @ctfmovement, 0);
```

Sui:

```
fun init(witness: FLASH, ctx: &mut TxContext) {
    let cap = coin::create_currency(witness, 2, ctx);
    let owner = tx_context::sender(ctx);

    let flash_coin = coin::mint(&mut cap, 1000, ctx);

    create_lend(flash_coin, ctx);
    transfer::transfer(cap, owner);
}
```

BEOSIN
Blockchain Security

Standard Library

Standard Library

01 Standard Library -> What could go wrong?

Here's How the Multi-Million BNB Chain Hack Went Down: Paradigm Researcher

Author: Chayanika Deka • Last Updated Oct 7, 2022 @ 15:59

The damage could have been far worse, according to a researcher.

The BNB Chain was temporarily paused after an exploit on its cross-chain bridge. The current impact estimate is around $100 million and $110 million equivalent of cryptocurrency.

According to the latest update, the BNB Chain has resumed working as usual, but let's have a look at how the hack went down, according to a popular researcher.

Binance has a special precompile contract that is used to verify IAVL trees. When a user verifies an IAVL tree, they need to specify a list of "operations." The Binance Bridge typically expects two of them: an "iavl:v" operation, and a "multistore" operation, Sun specified. The attacker managed to exploit the bug in the Binance Bridge that verified proofs allowing attackers to forge arbitrary messages.,

Aurora Engine: precompiles

- ExitToNear
- ExitToEthereum

```rust
pub mod exit_to_near {
    use aurora_engine_types::types::Address;

    /// Exit to NEAR precompile address
    ///
    /// Address: `0xe9217bc70b7ed1f598ddd3199e80b093fa71124f`
    /// This address is computed as: `&keccak("exitToNear")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xe9217bc7, 0x0b7ed1f598ddd3199e80b093fa71124f);
}

pub mod exit_to_ethereum {
    use aurora_engine_types::types::Address;

    /// Exit to Ethereum precompile address
    ///
    /// Address: `0xb0bd02f6a392af548bdf1cfaee5dfa0eefcc8eab`
    /// This address is computed as: `&keccak("exitToEthereum")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xb0bd02f6, 0xa392af548bdf1cfaee5dfa0eefcc8eab);
```

```rust
function withdrawToNear(bytes memory recipient, uint256 amount) external override {
    address sender = _msgSender();
    _burn(sender, amount);

    bytes32 amount_b = bytes32(amount);
    bytes memory input = abi.encodePacked("\x01", sender, amount_b, recipient);
    uint input_size = 1 + 20 + 32 + recipient.length;

    assembly {
        let res := call(gas(), 0xe9217bc70b7ed1f598ddd3199e80b093fa71124f, 0, add(input, 32), input_size
    }
}

function withdrawToEthereum(address recipient, uint256 amount) external override {
    _burn(_msgSender(), amount);

    bytes32 amount_b = bytes32(amount);
    bytes20 recipient_b = bytes20(recipient);
    bytes memory input = abi.encodePacked("\x01", amount_b, recipient_b);
    uint input_size = 1 + 32 + 20;

    assembly {
```

Tokens are burnt before exit

Standard Library

02 Standard Library

Index

- 0x1::any
- 0x1::aptos_hash
- 0x1::bls12381
- 0x1::capability
- 0x1::comparator
- 0x1::copyable_any
- 0x1::debug
- 0x1::ed25519
- 0x1::from_bcs
- 0x1::math128
- 0x1::math64
- 0x1::multi_ed25519
- 0x1::pool_u64
- 0x1::ristretto255
- 0x1::secp256k1
- 0x1::simple_map
- 0x1::table
- 0x1::table_with_length
- 0x1::type_info

Aptos (0x1::)

- address.move
- bag.move
- balance.move
- bcs.move
- coin.move
- devnet_nft.move
- digest.move
- dynamic_field.move
- dynamic_object_field.move
- epoch_time_lock.move
- erc721_metadata.move
- event.move
- hex.move
- immutable_external_resource.move
- locked_coin.move
- math.move
- object.move

Sui (sui::)

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

SECURING BLOCKCHAIN ECOSYSTEM

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

copyright © 2023 by Beosin. All Rights Reserved.