aelf.

DECENTRALISED WORLD FOR TOMORROW

Smart Contract & Blockchain Security Practical Blockchain Meetup at Google Developer Space

29 Feb 2024

What is aelf.

The scalable, high-performance blockchain ecosystem empowering developers to build diverse Web 3 applications in C#

Our Vision

Mass Adoption of Web3 by Real Users

What is DECENTRALISATION

Distribution of authority, shifting decision-making away from a central entity

What's the Potential?

Accountability

Resilience

How safe is BLOCKCHAIN

It is resistant to tampering and provides transparency in transactions.

Foundations of blockchain

- Immutable Ledger
- Decentralization
- Cryptography
- Consensus Mechanisms

What are SMART CONTRACTS

Piece of code which enables reading and writing data from/to the blockchain

- Self-executing contracts with the terms of the agreement directly written into code
- Eliminate the need for trust between parties involved in a transaction

Properties

Immutable

Transparent

Automated

Is blockchain

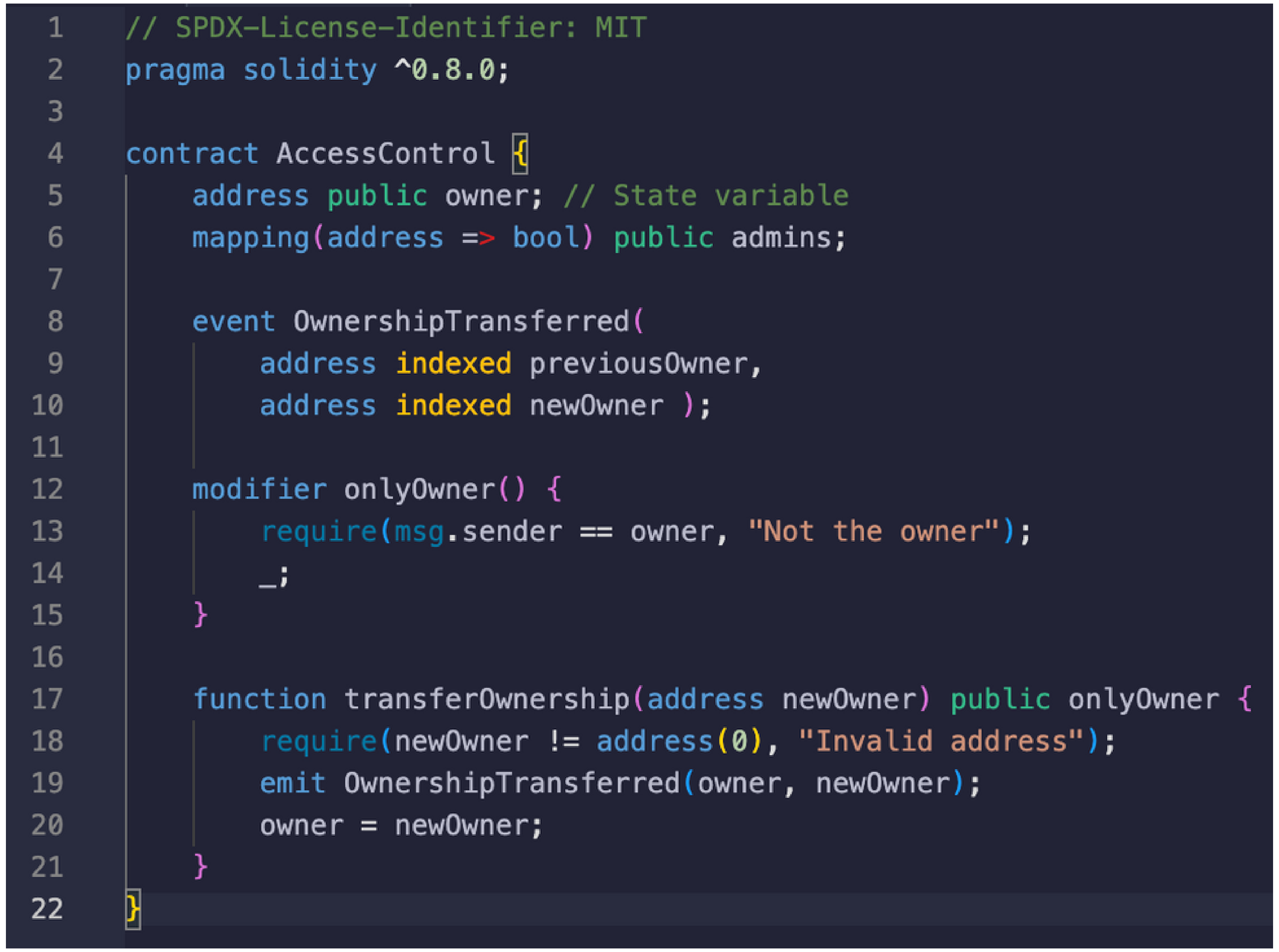
REALLY SAFE?

Blockchain's security improves with the help of decentralisation, cryptography & secure smart contracts

"Nothing is safe if it is not kept safe.."

Smart Contract Concepts

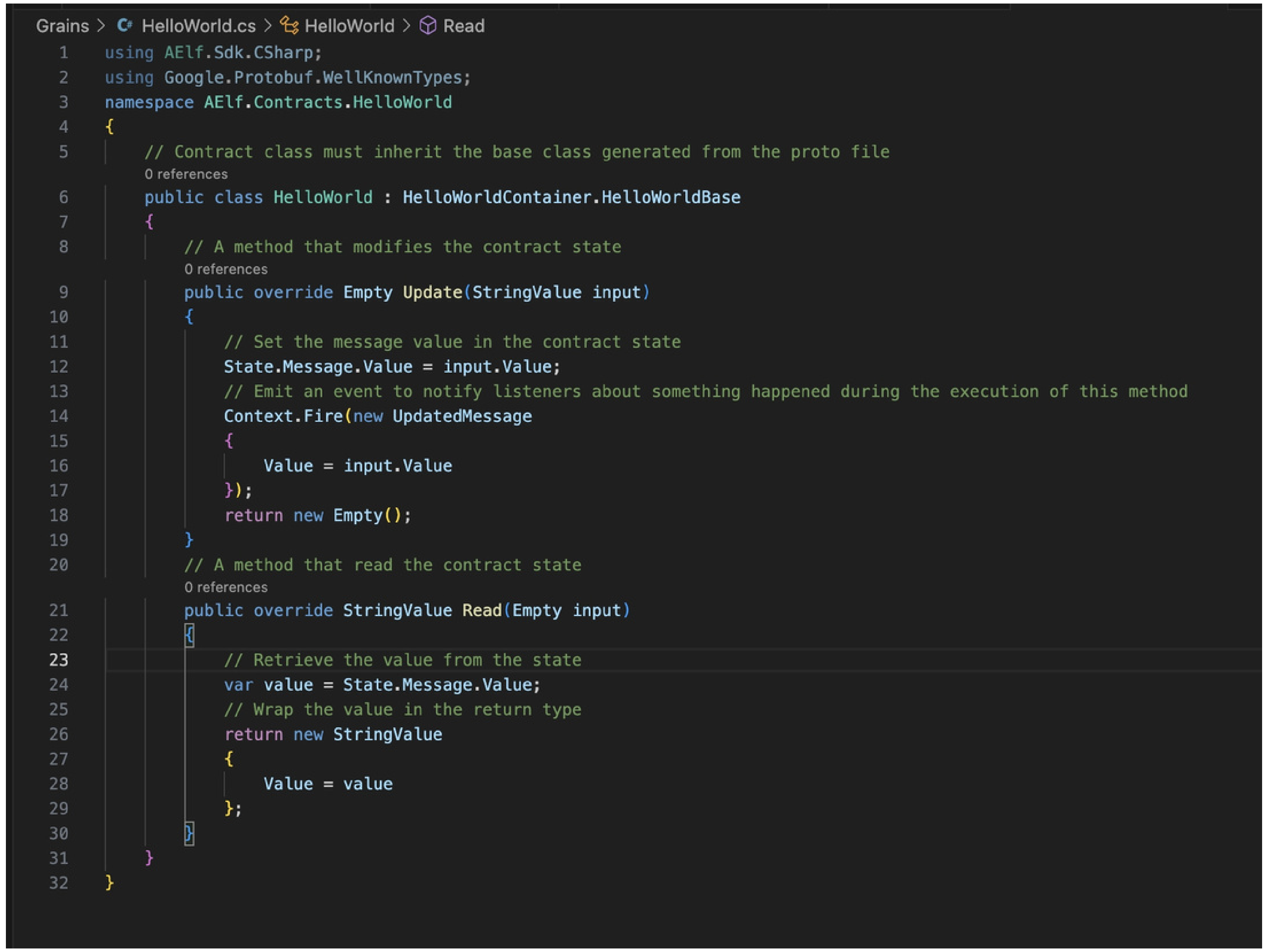
1. State Variables
2. Functions
3. Modifiers
4. Access Control
5. Events



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract AccessControl {
    address public owner; // State variable
    mapping(address => bool) public admins;

    event OwnershipTransferred(
        address indexed previousOwner,
        address indexed newOwner );

    modifier onlyOwner() {
        require(msg.sender == owner, "Not the owner");
        _;
    }

    function transferOwnership(address newOwner) public onlyOwner {
        require(newOwner != address(0), "Invalid address");
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
    }
}
```

aelf’s Smart Contract Concepts

1. State Variables
2. Functions (Action & View)
3. Proto Files
4. Context property
5. Access Control



Grains > HelloWorld.cs > HelloWorld > Read

```csharp
using AElf.Sdk.CSharp;
using Google.Protobuf.WellKnownTypes;
namespace AElf.Contracts.HelloWorld
{
    // Contract class must inherit the base class generated from the proto file
    0 references
    public class HelloWorld : HelloWorldContainer.HelloWorldBase
    {
        // A method that modifies the contract state
        0 references
        public override Empty Update(StringValue input)
        {
            // Set the message value in the contract state
            State.Message.Value = input.Value;
            // Emit an event to notify listeners about something happened during the execution of this method
            Context.Fire(new UpdatedMessage
            {
                Value = input.Value
            });
            return new Empty();
        }
        // A method that read the contract state
        0 references
        public override StringValue Read(Empty input)
        {
            // Retrieve the value from the state
            var value = State.Message.Value;
            // Wrap the value in the return type
            return new StringValue
            {
                Value = value
            };
        }
    }
}
```

Recent Smart Contract Hacks

- 2016 – The DAO Attack
- 2017 – Parity Wallet Vulnerability
- 2018 – Batch Overflow, Proxy Overflow
- 2018 – Gas Token Re-entrancy Attack
- 2020 – Uniswap ERC20 attack
- 2021 – Alpha Finance Flash Loan Attack
- 2022 – Ronin Bridge Hack

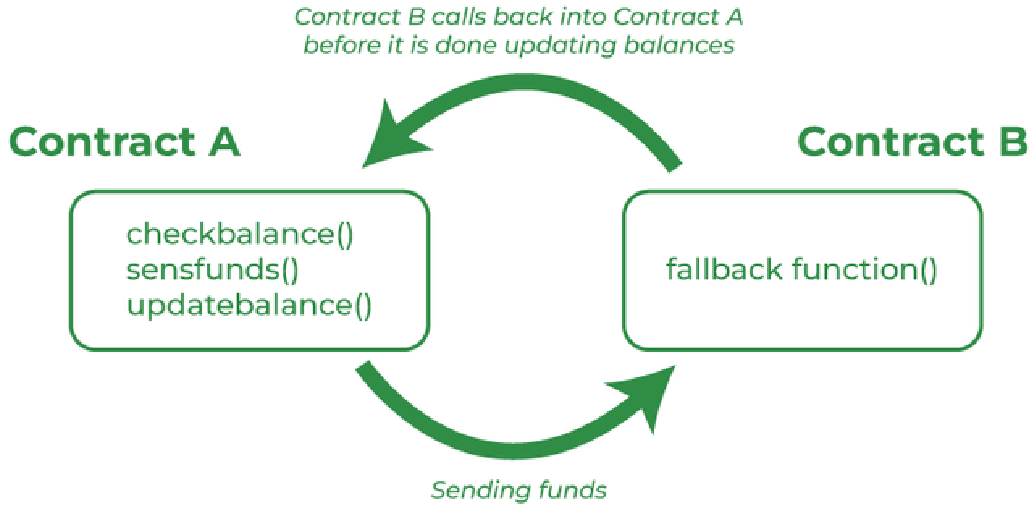
How to mitigate SC Attacks?

- **Re-entrancy Attack**
  - Use withdrawal patterns
  - Checks-Effects-Interactions pattern

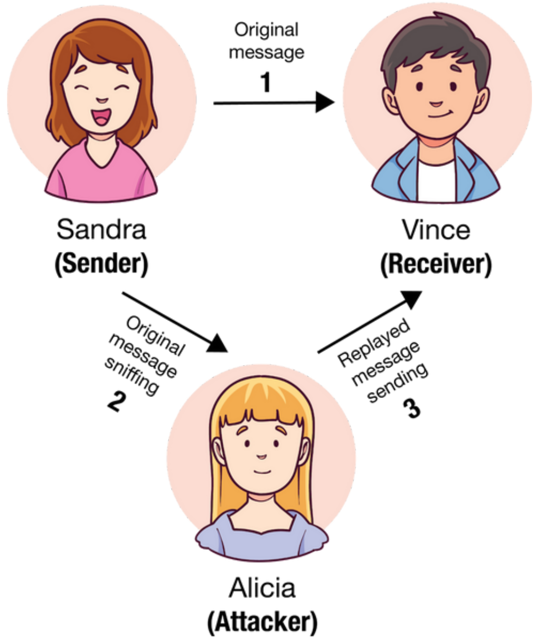
- **Signature Replay**
  - Use nonce or timestamp along with signatures to ensure that each signed message can only be executed once

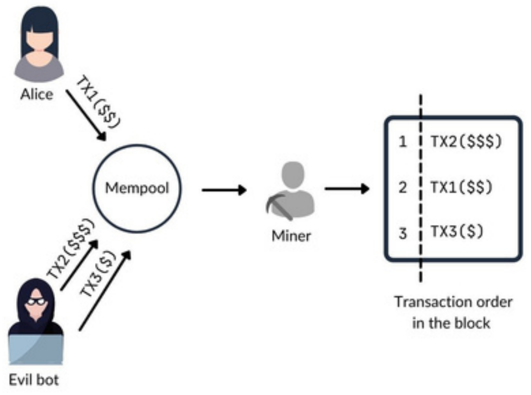
- **Insecure Source of Randomness**
  - Use external randomness sources like Chainlink VRF or Oraclize

- **Front Running Attack**
  - Use timestamp-based randomness
  - Implement strategies like commit-reveal schemes



HOW A REPLAY ATTACK WORKS





How aelf mitigate SC attacks

- RE ENTRANCY
- SIGNATURE REPLAY
- UNSAFE DELEGATED CALL
- DENIAL OF SERVICE

- HONEYPOT
- FRONT RUNNING
- INSECURE SOURCE OF RANDOMNESS
- ACCESS PRIVATE DATA

Best Practices for SC Security

- Sanitise all user input to prevent code injection and other vulnerabilities
- Implement access control mechanisms to restrict access to sensitive functions and data
- Use secure/whitelisted libraries and avoid implementing custom cryptography
- Keep your smart contracts simple, modular, and testable
- Test smart contracts thoroughly and use formal verification tools to ensure accuracy

Best practices for SC Security

- Implement emergency stop mechanisms to pause or disable the contract in case of emergency
- Use secure coding practices such as not using global variables, function overloading, and proper use of visibility modifiers
- Use events to inform users about important contract state changes
- Use multi-factor authentication and cold wallets to secure sensitive keys

Tools and Resources for SC Security

- Security Auditing Firms
- Static Analysis Tools (e.g., MythX, Securify)
- Dynamic Analysis Tools (e.g., Truffle Debugger)
- Formal Verification Tools (e.g., Solidity Prover)
- Bug Bounty Programs (With Certik)

ENABLING SC SECURITY AT aelf.

Secure Experience

1 Authenticity

- Immutable and distinct tokens: unique identifier for every token or NFT collection created

2 Layered Security

- Multi-party approval is required to increase transaction limits
- Prevents wallet drains from hackers

Unique Features of aelf

Multi-chain Structure

Flexible Fees

Anti Congestion

Customisable Sidechains

Building a Decentralised World

1 Enabling Enterprise

- Customised side chains
- International data standard

2 Empowering Users

- Seamless onboarding through Portkey
- Smooth user experience
- Scam prevention mechanisms

3 Equipping Developers

- Familiar programming languages
- Workshops
- Engagement through TMRW Dao Platform

aelf.

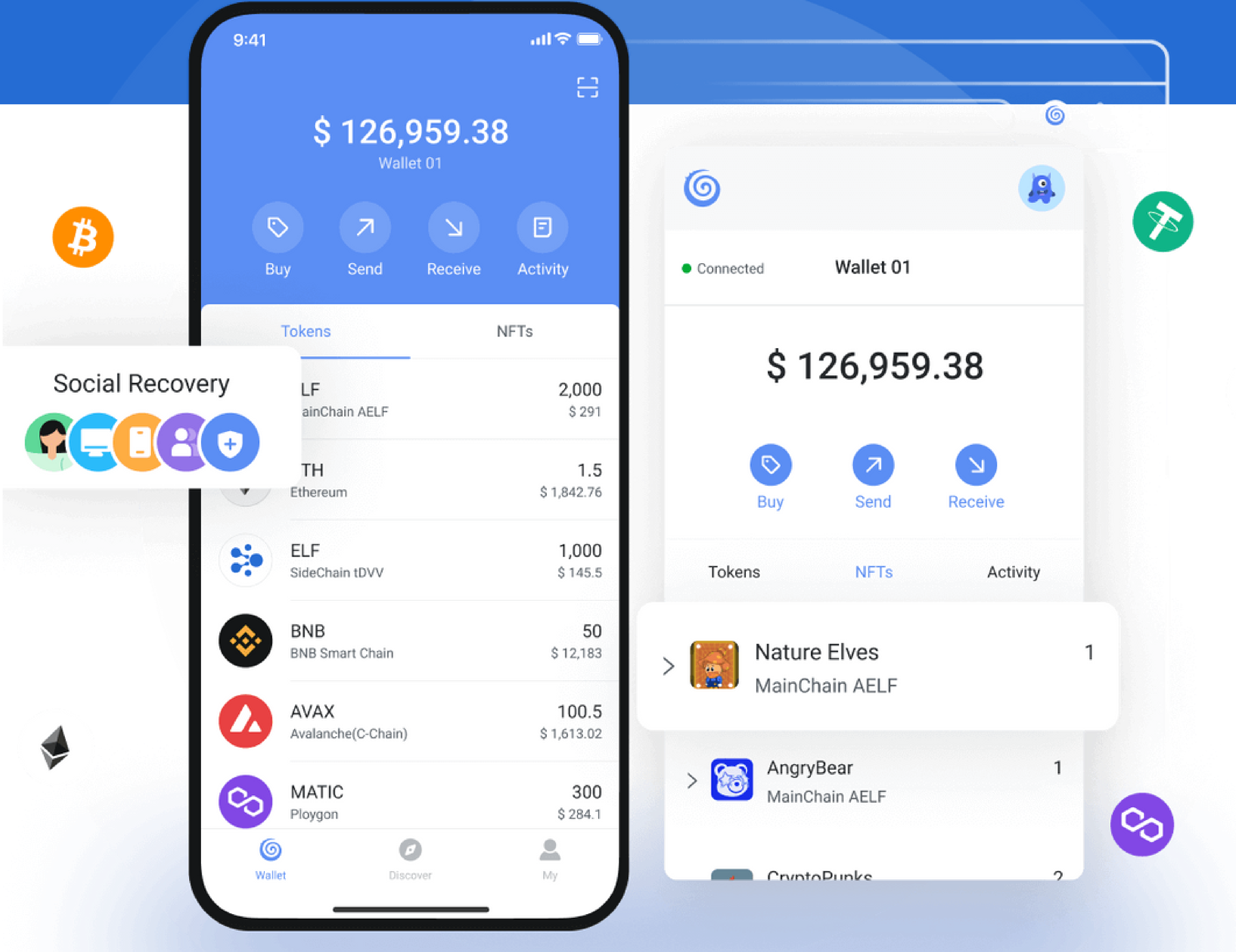
Explore aelf's Ecosystem

PORTKEY	AA social recovery wallet	ETransfer	Asset deposit & withdrawal tool	TMRW DAO	DAO tooling platform
eWell	Initial Decentralised Offering platform	eBridge	Cross-chain bridge	ÆLFVENTURES	Venture capital fund
FOREST	NFT marketplace	AWAKEN SWAP	Decentralised exchange	ÆLEVATE	Gaming grant
Symbol Market	Token creation platform	BeanGo TOWN	Fully onchain game	ævolvelabs	Incubator

Get rewarded to integrate aelf & Portkey

Portkey is a social recovery Web2-to-3 wallet enabling your migration to blockchain!

- Familiar Web2 logins with social recovery enabled through Account Abstraction;
- Fully decentralised, enabled by our verifiers & guardians technology;
- Perfect for applications with non-blockchain native use cases such as gaming;
- Preferred gateway to the AELF ecosystem including the 150k USD Aelevate grant.



Code Comfortably

Built on .NET

Enterprise devs can customise their side chains easily

C# for smart contracts

Devs can build DApps without learning a new programming language

Multi-language SDK

Seamless interaction between smart contracts and enterprise systems

TOMORROW IS RUNNING ON ÆLF

Decentralisation is the journey.

Every step brings us closer to a decentralised world for all.

Be a part of the journey to a decentralised world.

TOMORROW IS RUNNING ON ÆLF

Let's Connect

Unlock unlimited possibilities in blockchain!

MOHIT VASHISTHA
Blockchain Reseacher
@ ælf pte. ltd
@mohit6b

GAN KOK KOON
Developer Relation Specialist
@ ælf pte. ltd
@KK9AN