Getting started with ZeppelinOS

Building upgradeable smart contract applications

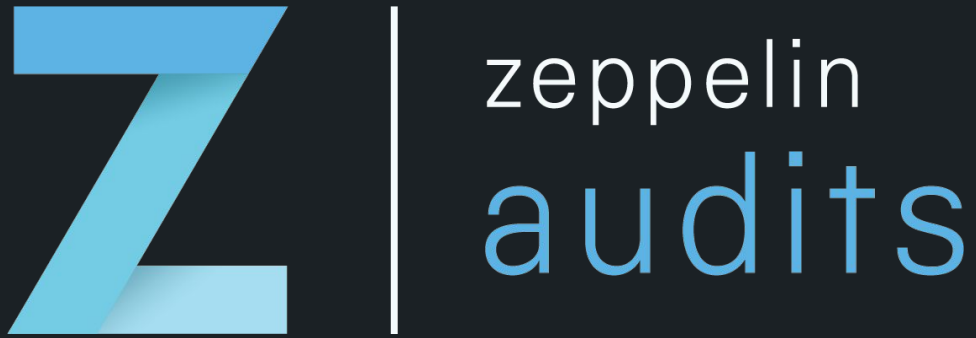

zeppelin

Facu Spagnuolo

Core dev and security researcher

@facuspagnuolo


openzeppelin


zeppelin
audits

ZeppelinOS

Immutability

ZeppelinOS

Immutability

Downsides

ZeppelinOS

No way to change our source code

That's why we all:

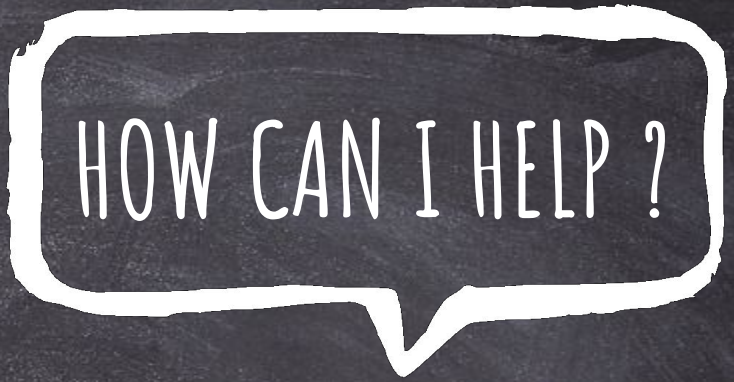
- ➔ perform security audits
- ➔ reuse existing audited code

Immutability

Downsides

ZeppelinOS

There is no way to upgrade our code once deployed



ZeppelinOS

ZeppelinOS

npm install -g zos

ZeppelinOS

Upgradeable smart contracts

On-chain standard libraries

Upgradeable smart contracts

On-chain standard libraries

Upgradeability

Initializing your project

ZeppelinOS

```
$ npm install -g zos
$ zos init my-project
```

Upgradeability

Write your smart contract

ZeppelinOS

```
contract MyWallet {
   address public owner;
   ...
   function initWallet(address _owner) public {
     owner = _owner
   }

  function withdraw(uint256) public onlyOwner { … }
  ...
}
```

Upgradeability

Adding your contracts

ZeppelinOS

```
$ zos add MyWallet
$ zos push -n ropsten
$ zos create MyWallet -n ropsten
> 0x2c2b9c9a4a25e24b174f26114…
```

Upgradeability

Buggy contract

ZeppelinOS

```
contract MyWallet {
  address public owner;
  ...
  function initWallet(address _owner) public {
    owner = _owner
  }

  function withdraw(uint256) public onlyOwner { … }
  ...
}
```

Upgradeability

Buggy contract

ZeppelinOS

```
> wallet = MyWallet.at('0x2c2b9c9a4a25e24b174f26114…')
> wallet.initWallet(attacker, { from: attacker })
# success


> wallet.withdraw(100, { from: attacker })
# success
```

Upgradeability

Fixing the bug

ZeppelinOS

```
contract MyWallet {
   address public owner;
   ...
   function initWallet(address _owner) public onlyOwner {
     owner = _owner
   }

  function withdraw(uint256) public onlyOwner { … }
  ...
}
```

Upgradeability

Upgrading your contracts with a bugfix

ZeppelinOS

```
$ zos add MyWallet
$ zos push -n ropsten
$ zos update MyWallet -n ropsten
```

Upgradeability

Contract fixed

ZeppelinOS

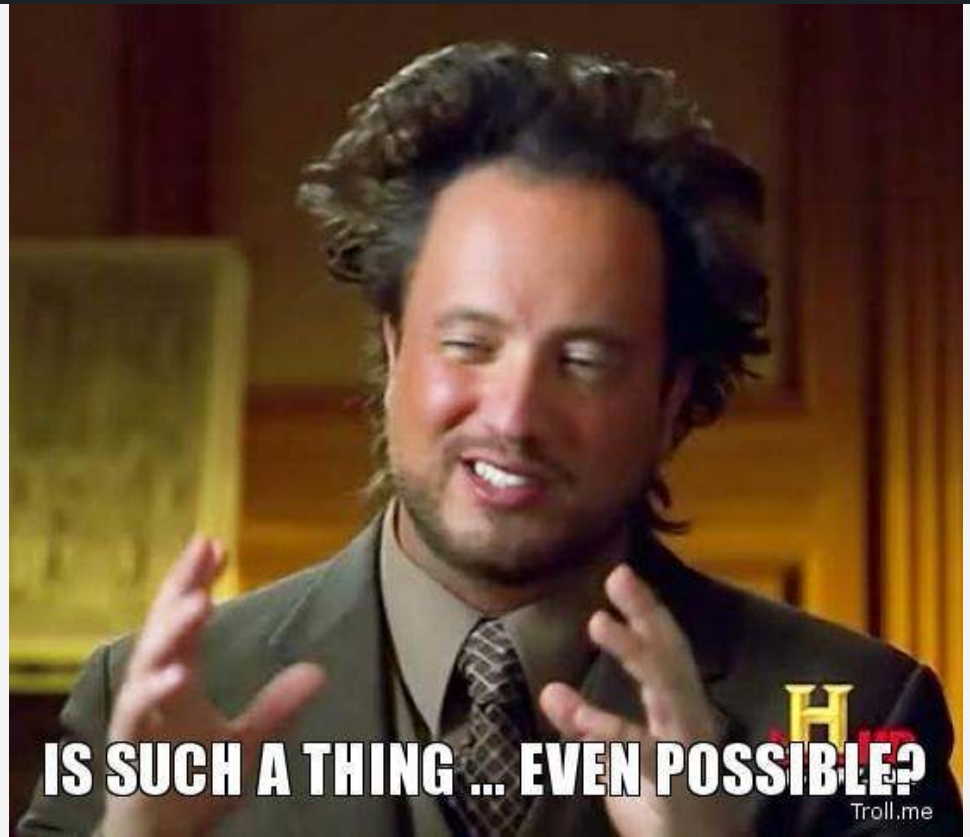
```
> wallet = MyWallet.at('0x2c2b9c9a4a25e24b174f26114…')
> wallet.initWallet(attacker, { from: attacker })

# Revert, initWallet can now only be called from owner
```

Upgradeability

How was that possible?

ZeppelinOS



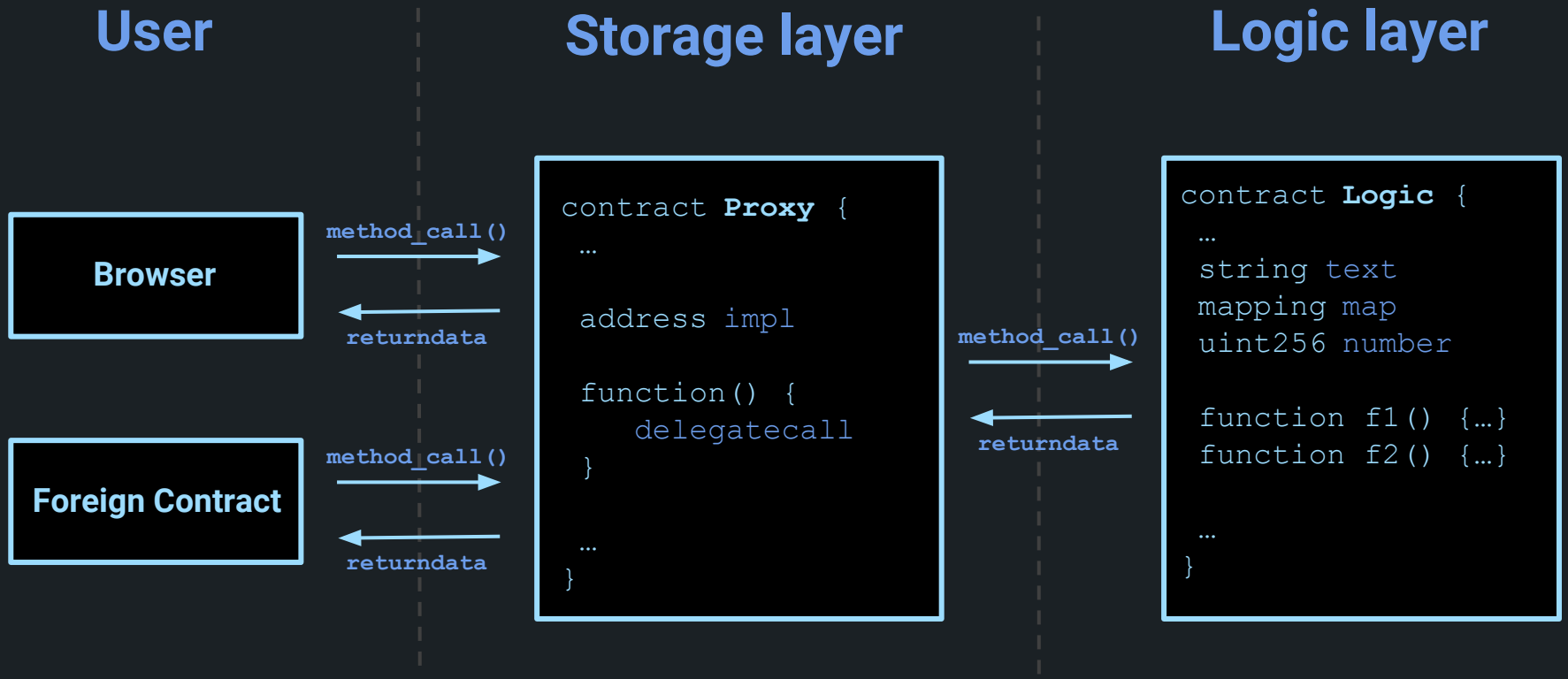
PROXIES

EVERYWHERE

Upgradeability

Proxies

ZeppelinOS



Upgradeability

Extending functionality

ZeppelinOS

```
contract ETHBerlinToken is StandardToken { … }

contract StandardToken {
  …
  function totalSupply() public view returns (uint256) { … }
  function balanceOf(address owner) public view returns (uint256) { … }
  function approve(address spender, uint256 value) public returns (bool) { … }
  function transfer(address to, uint256 value) public returns (bool) { … }
  function transferFrom(address from, address to, uint256 value) public returns (bool) { … }
  …
}
```

Upgradeability

Extending functionality

ZeppelinOS

```
$ zos add ETHBerlinToken
$ zos push -n ropsten
$ zos create ETHBerlinToken -n ropsten
> 0x2c2b9c9a4a25e24b174f26114…
```

Upgradeability

Extending functionality

ZeppelinOS

```
contract StandardToken {
  ...
  function totalSupply() public view returns (uint256) { ... }
  function balanceOf(address owner) public view returns (uint256) { ... }
  ...
}

contract ETHBerlinToken is StandardToken {
  ...
  function burn(uint256 value) public returns (bool) { ... }
  ...
}
```

Upgradeability

Extending functionality

ZeppelinOS

```
$ zos add ETHBerlinToken
$ zos push -n ropsten
$ zos update ETHBerlinToken -n ropsten
```

Upgradeability

Extending functionality

ZeppelinOS

```
> token = ETHBerlinToken.at('0x2c2b9c9a4a25e24b174f4…')
> token.balanceOf(owner)
# 100


> token.burn(100, { from: owner })
# true, burn method added in upgrade
```

Upgradeability

Conclusion

- we can fix bugs
- we can add new functionality

Upgradeable smart contracts

On-chain standard libraries

On-chain standard libraries

ERC721 example

ZeppelinOS

```
$ zos add Deck
$ zos link openzeppelin-zos@1.9.3
$ zos push -n ropsten
```

On-chain standard libraries

ERC721 example

ZeppelinOS

```
contract Deck is Initializable {
  ERC721 public erc721;

  function initialize(ERC721 _erc721) public isInitializer {
    erc721 = _erc721;
    erc721.initialize();
  }

  function pick(uint256 number) public {
    require(!erc721.exists(number));
    erc721.mint(msg.sender, number);
  }
}
```

On-chain standard libraries

ERC721 example

ZeppelinOS

```
$ zos create ERC721 -n ropsten
> 0x2c2b9c9a4a25e24b174f26114…

$ zos create Deck --args 0x2c… -n ropsten
> 0x30753e4a8aad7f8597332e813…
```

On-chain standard libraries

ERC721 example

ZeppelinOS

```
> deck = Deck.at('0x30753e4a8aad7f8597332e813…')
> deck.pick(10, { from: me })
# success

> erc721 = deck.erc721()
> erc721.ownerOf(10)
# me
```


AMAZING

On-chain standard libraries

ZeppelinOS

ERC721 example

On-chain standard libraries

ERC721 example

ZeppelinOS

```
$ zos link openzeppelin-zos@1.9.4    //latest
$ zos push -n ropsten
$ zos update ERC721 -n ropsten
```

On-chain standard libraries

ERC721 example

ZeppelinOS

```
> deck = Deck.at('0x30753e4a8aad7f8597332e813…')
> deck.pick(11, { from: me })
# success with less gas

> erc721 = deck.erc721()
> erc721.ownerOf(11)
# me
```

On-chain standard libraries

Conclusion

ZeppelinOS

- it simplifies how we use dependencies
- we don't need to deploy code to reuse it
- we can upgrade dependencies without changing our contracts

ZeppelinOS

npm install -g zos


ASK FOR HELP.
I'M HAPPY TO HELP.

ZeppelinOS

Find out more at

zeppelinos.org

Documentation and guides

docs.zeppelinos.org

Learn more at

blog.zeppelinos.org

Exercise

ZeppelinOS

<zpl.in/zos-ethberlin>

```
$ npm install -g zos
$ git clone git@github.com:zeppelinos/zos-ethberlin-exercise.git
$ npm install
```

RELEASE
CANDIDATE

openzeppelin

2.0.0-rc

Thank you!

Learn more

zeppelin.solutions
openzeppelin.org
zeppelinos.org

Contact

facu@zeppelin.solutions
@facuspagnuolo
facuspagnuolo.com