Hacking Smart Contracts

Introduction to Smart Contract Security

Ács Dávid – 2022 – UTCN CTF group

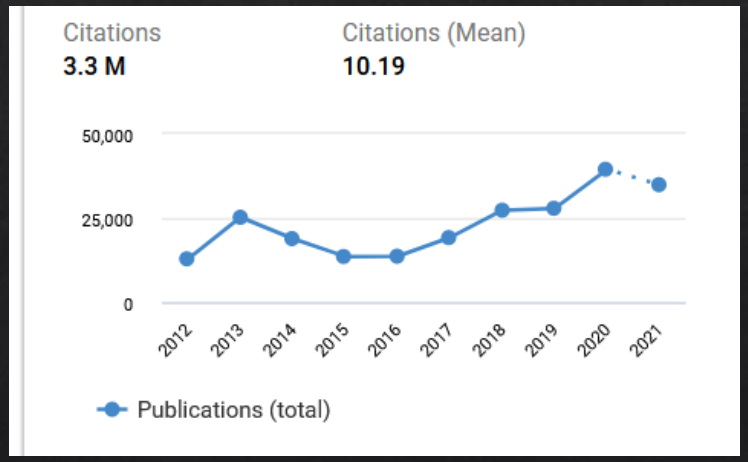
Main message

- Ethereum smart contracts are written by programmers (humans), who can make mistakes resulting in vulnerabilities.

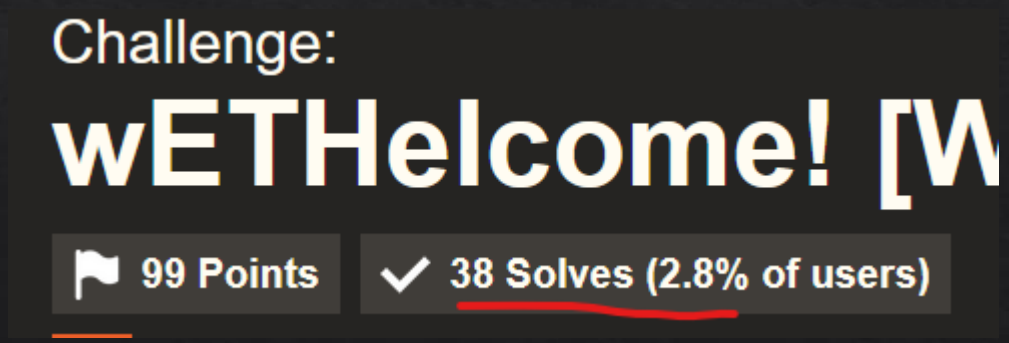
- I will show you how to:
  - Build a basic smart contract
  - Spot and exploit smart contracts vulnerable to re-entrancy attacks.

Why?

- General considerations:
  - Smart contracts are relatively new
  - Inexperienced devs => vulns
  - Area of active research: publications containing "smart contract security"



- From CTF point of view:
  - Few players understand ETH, thus:
  - ETH challenges are often ignored
  - ETH challenges have more points
  - From 1300+ teams at X-MAS 2021:

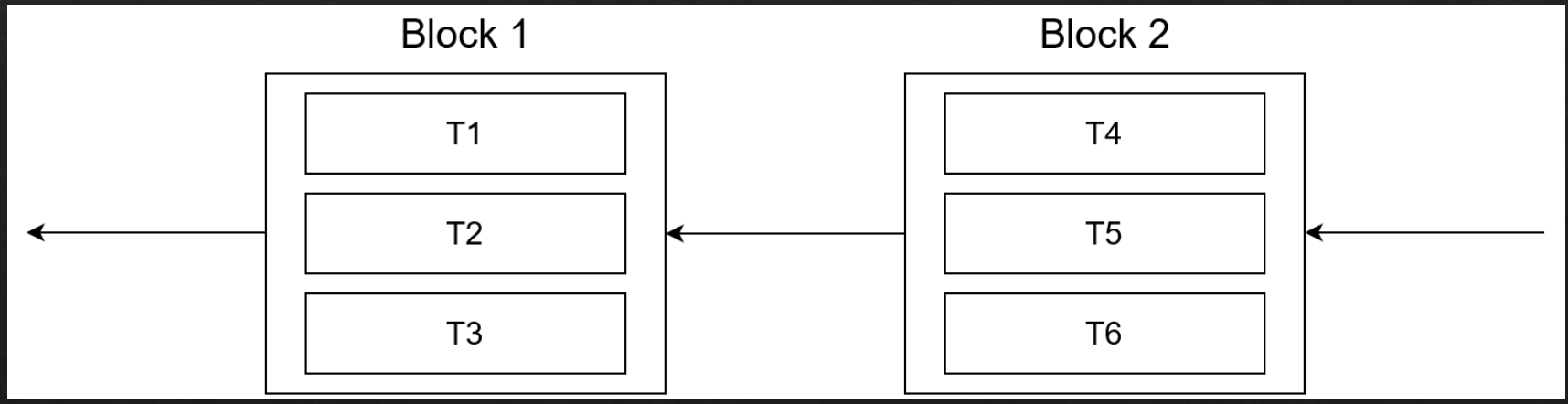


Building Smart Contracts

- “What I cannot create, I do not understand” - Richard Feynman
- “What I do not understand, I cannot hack” – me ☺
- Let’s write some smart contracts ☺

Background: Blockchain

- Public, append only database (data layer)
- Shared across many computers (nodes) in a network
- Block: batches of transactions
- Chain: contains hash of previous block (parent) in the chain



Background: Ethereum protocol/universe

- Ethereum Virtual Machine (EVM)
  - Built on top of blockchain
  - Every node keeps a copy of the state of the EVM computer
- Two operations:
  - Query / Read => can be performed locally
  - State changing (transactions) => broadcast and verify by all nodes
- Do not confuse with Ether, which is the currency.

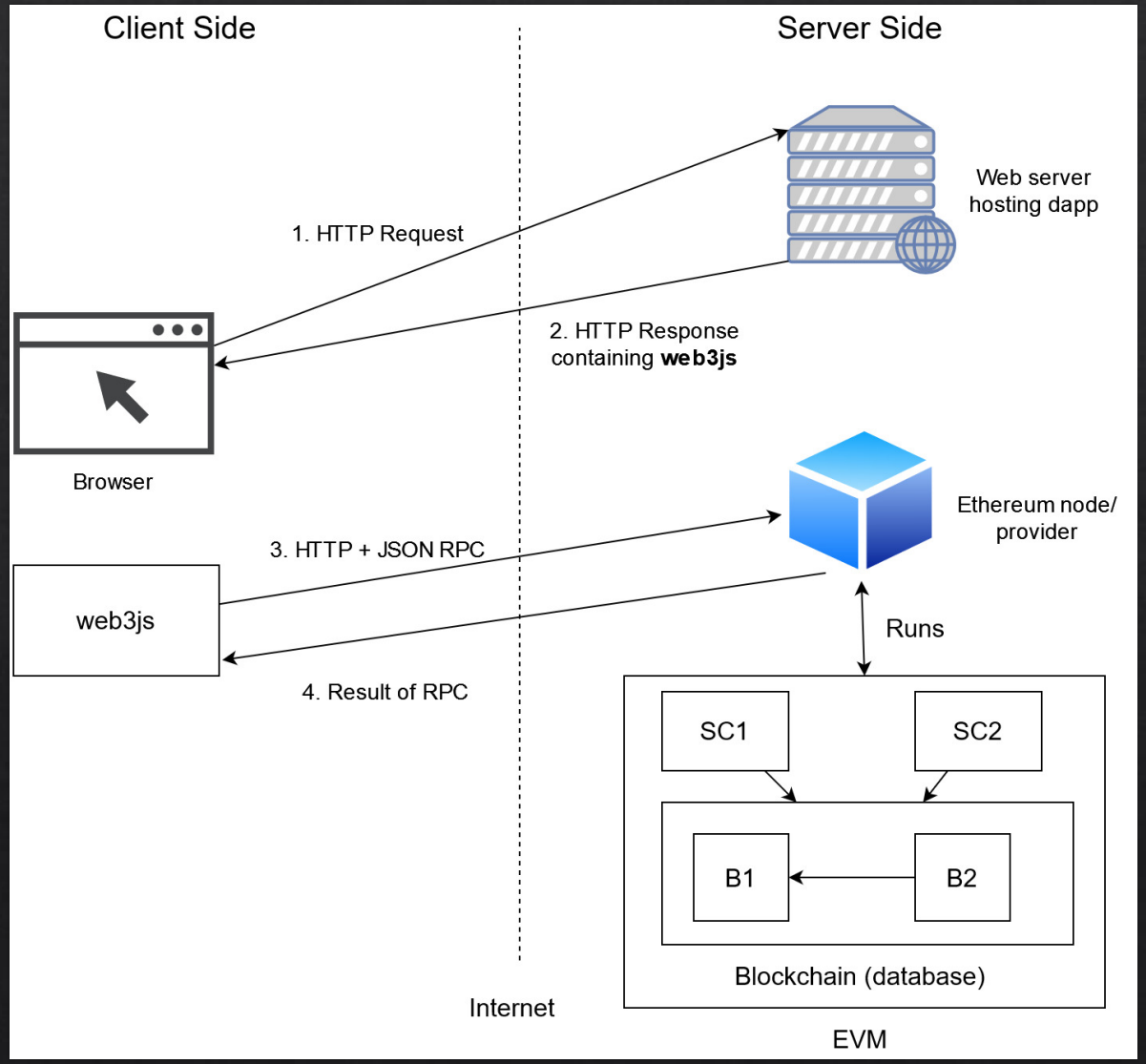
Background: Smart Contracts (SCs)

- Program that is executed by EVM => business logic
- Developers pay a fee to upload it to the network
- Users pay a fee to interact with it (make transactions)
- Example smart contract – Election System:
  - everyone has one vote
  - no double voting
  - no voting after end of election

Background: Interacting with Smart Contracts

- Web3js library => part of front end
- Connect to a node (provider) via:
  - HTTP + JSON RPC, if node is remote or local
  - Websocket + JSON RPC, if node is remote or local
  - IPC (pipes) + JSON RPC, if node is local only
- Application Binary Interface (ABI):
  - JSON describing interface of Smart Contract – i.e. what can it do?

Background: putting it all together => dApp



Hello World of SCs: dev env

- Truffle docs & more: http://trufflesuite.com/index.html
- Install: `npm install -g truffle`
- Most important features:
  - Creating projects (`init`)
  - Compile all code (`compile`)
  - Deploying code to network (`migrate`)
  - Interact with or test SCs (`console/test`)
  - Debug transactions (`debug`)

Hello World of SCs: programming language

- Solidity: docs, source code
- OOP language: influenced by C++, Python, JS
- Compiles to EVM bytecode
- Contract is equivalent of a class
- Good resource: https://solidity-by-example.org/

Hello World of SCs: show me the code!

D: > tmp > eth_storage > contracts > SimpleStorage.sol

```
1   // this source code is compatible with solidity version 0.4.16 -> 0.9.0
2   pragma solidity >=0.4.16 <0.9.0;
3
4   // a contract is like a class, syntax similar to C++/Java
5   contract SimpleStorage {
6       // a contract can have member variables.
7       uint storedData;
8
9       // state changing function (needs transaction):
10      function set(uint x) public {
11          storedData = x;
12      }
13
14      // query only function (does not need transaction, equivalent of const method in C++):
15      function get() public view returns (uint) {
16          return storedData;
17      }
18  }
19
```

Hello world of SCs: deployment code

- Migrations folder contains deployment code
- Each file starts with a number, used to order migrations

D: > tmp > eth_storage > migrations > JS 2_storage_migration.js > ...

```
1  const SimpleStorage = artifacts.require("SimpleStorage");
2
3  module.exports = function (deployer) {
4    deployer.deploy(SimpleStorage);
5  };
6
```

Hello world of SCs: deploying

```
d:\tmp\eth_storage>truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0xc0f63909571d821ca6c2b69b129facc0c2c1bf54
(1) 0xba32cea4d8c7be328129b893db54f40ce34d0ff2
(2) 0xbca21ecfd6ac90255b8c403b8e9f58a90cc03d86
(3) 0xf8f821815b3ece1330ce9f64e1ead39213e3371f
(4) 0x6fea28a192727e1fb2c7806afe5dcefa1b8e852f
(5) 0x6e1aea09f22b5a52df5da4ab9cbc772a206eae3f
(6) 0xce12973a093505ba8c53a049878408134d348dfc
(7) 0x8333a0d1296ec26fca9e6c2b5490c8f42b6629a0
(8) 0x1d14064dfa05090b59671a4a24f26296d0d84a2f
(9) 0xa3020669f1379f7cddf633b2a2e4acb7d59824ae

Private Keys:
(0) babe5f528aba959a04f757f8c6f5d273a82c8c8f15c8476971a0f49832694e52
(1) 1e9a94f40abef2771a73fb1b614613ba6f7fdf2ccb1ef193779363bacbaa0a0a
(2) 5d5a617227abd5056636f4eaccaefa2ba9269322df2e9a5ef18430853df52412
(3) b4463bc8fc1e19364f2ec61952ccf745abb3d5652c3e25bd74658ba3a6b32997
(4) b20db22af4b7dded59ab4ac0db2b9ac095dd9f4ac753a93eb78a42696a7a876b
(5) d3d91a74075397cd6f148eda121d92809b5ac171ef886e94304b3bc5ec99df0d
(6) c546fefad0d9474d914628a439ecbb472ac76fd34bce95b812be7ad07bc072a7
(7) 31721e262f5c5fd7a442832a8cafd76e3c951c95a02583a58a3851de69e8d22e
(8) f7cd836bc23f179eed244349dde0e4d798b9f54d0766d907bfb89ab85d66f1f8
(9) 512b860da7a1f91df722e35e3978f7cdd3e44e05c263c6324ef208696026ce2b

Mnemonic: muffin version imitate obtain merge away arena exact flavor attitude vast goddess

□□  Important □□  : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

```
d:\tmp\eth_storage>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\SimpleStorage.sol
> Artifacts written to d:\tmp\eth_storage\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0xed2a59890b46f5df43ddc319a7dc92b20dab85036f24c2c75257bb858f4dce49
   > Blocks: 0            Seconds: 0
   > contract address:    0x79Dc55170466B7282EB3FA3C228A22147B9A8f1C
   > block number:        1
   > block timestamp:     1640434651
   > account:             0xc0F63909571D821Ca6c2B69b129FacC0c2C1BF54
   > balance:             99.99616114
   > gas used:            191943 (0x2edc7)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00383886 ETH
```

Hello World of SCs: interacting

- In truffle `console`:

```
truffle(develop)> let ss = await SimpleStorage.deployed()
undefined
truffle(develop)> let currValue = await ss.get()
undefined
truffle(develop)> currValue
BN { negative: 0, words: [ 0, <1 empty item> ], length: 1, red: null }
truffle(develop)> currValue.toNumber()
0
truffle(develop)> let transaction = ss.set(1)
undefined
truffle(develop)> let updatedValue = await ss.get()
undefined
truffle(develop)> updatedValue.toNumber()
1
truffle(develop)>
```

Demo time: wETHelcome!

- Connect to provider
- Change a bool flag from false to true
- Get flag, it’s that simple :D

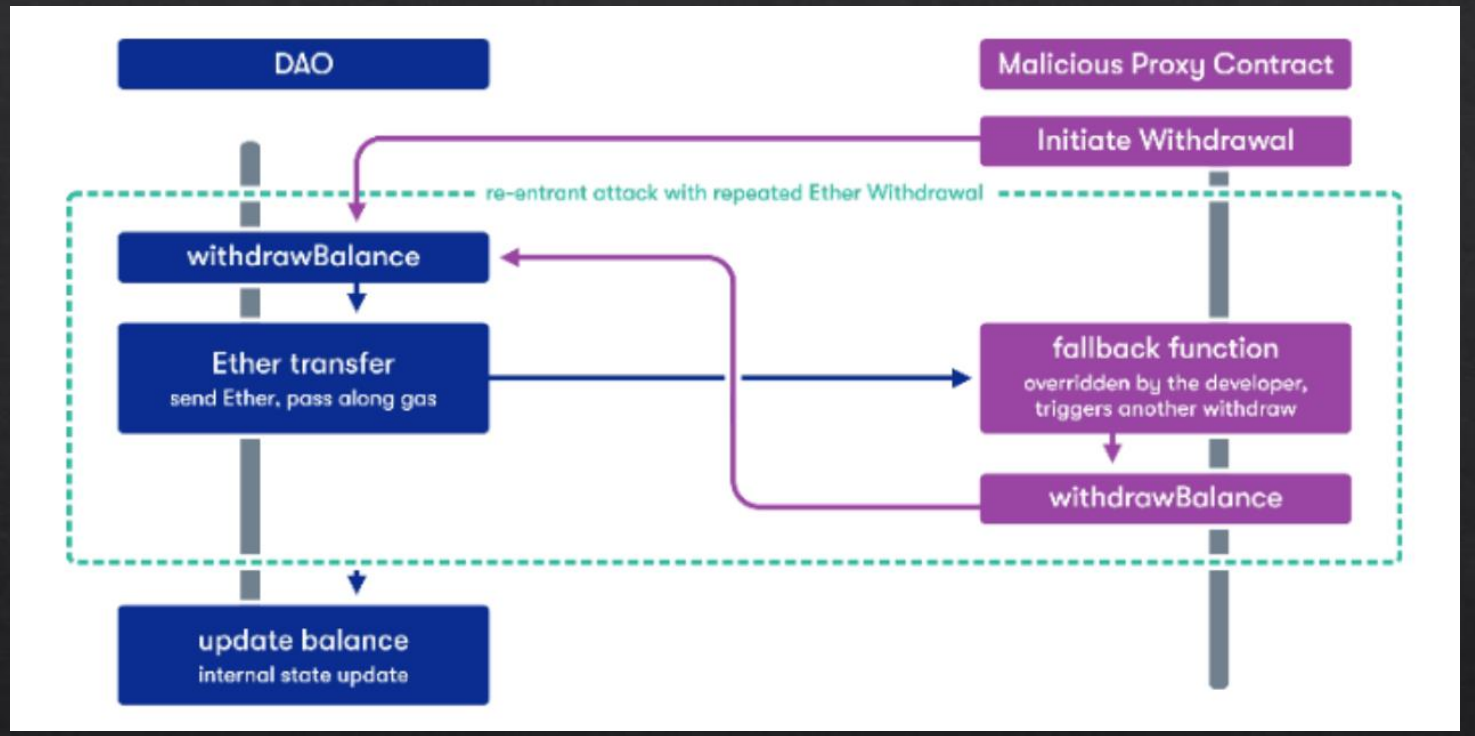
Demo time: lvl 100 cookie boss

- Connect to provider
- Find password hardcoded in SC
- Use password to get flag

Demo time: 残響 (reverberation/echo)

- "Iterator" SC vulnerable to reentrancy attack
  - Vulnerability allows attacker to withdraw all funds from SC
  - Fun fact: $150 million stolen from DAO using this technique
- SC is vulnerable if:
  - Uses `call` to send ether
  - Does not use reentrancy lock
  - subtracts balance after `call`

Flow of reentrancy attack



- Source: https://quantstamp.com/blog/what-is-a-re-entrancy-attack

Demo time: 残響 (reverberation/echo)

- Deploy a SC that:
  - has a `fallback` function to receive money
  - `fallback` function recursively withdraws money from vulnerable SC
  - has an `attack` function to perform the initial withdrawal
- Get flag

Prevent reentrancy attacks

- Checks-effects-interactions pattern:
  - Check preconditions (does user have sufficient funds?)
  - Modify internal state (subtract amount from balance of user)
  - ONLY as final step send the funds
- Reentrancy guard / mutex:
  - https://solidity-by-example.org/hacks/re-entrancy/

Thank you!
Questions?

References/further reading

- https://ethereum.org/en/developers/docs/intro-to-ethereum/
- https://www.zastrin.com/courses/ethereum-primer/lessons/1-5
- https://docs.soliditylang.org/en/latest/security-considerations.html
- https://consensys.github.io/smart-contract-best-practices/known_attacks/
- https://cryptosec.info/defi-hacks/
- http://trufflesuite.com/docs/truffle/