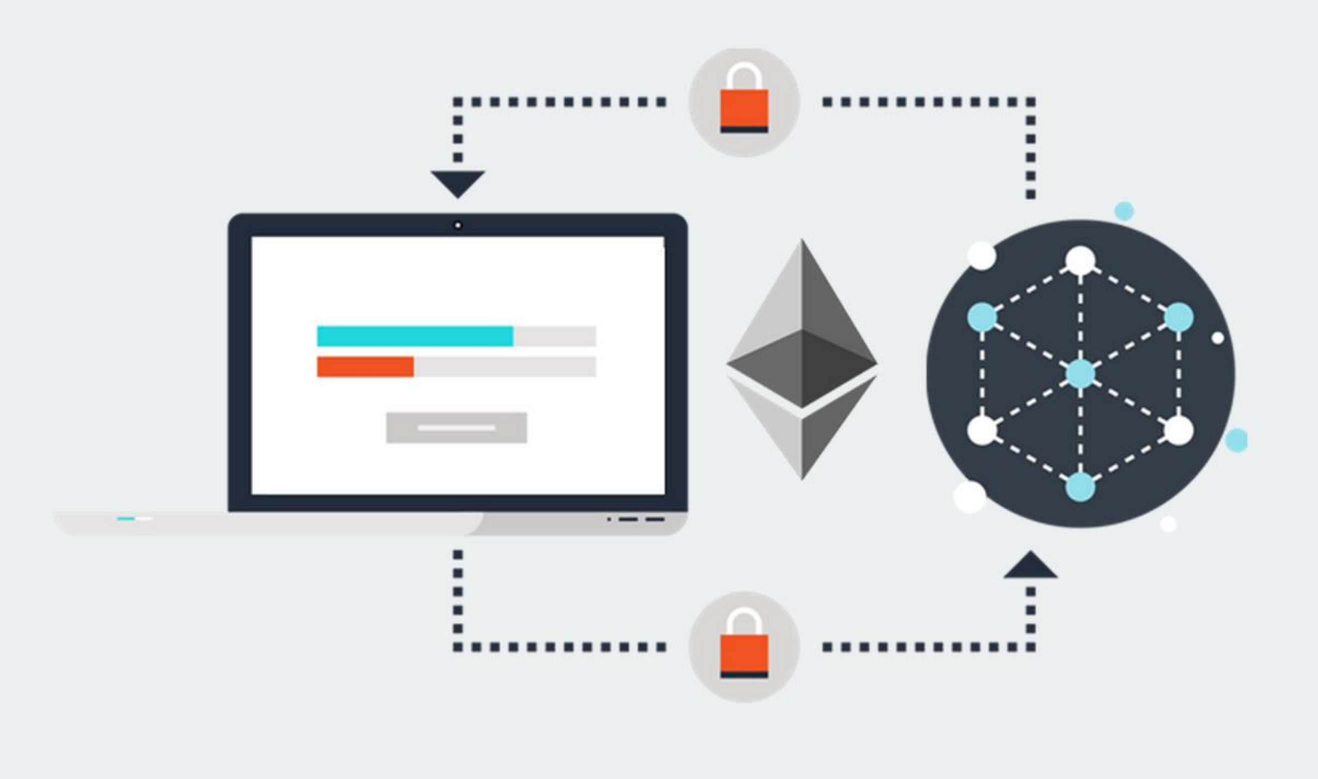


Secure Smart Contracts Development using SCSVS

Damian Rusinek

Introducing Decentralized Applications by analogy to Web Apps

Damian Rusinek

drdr_zz

damianrusinek @ github

- Senior Security Consultant
- Researcher (blockchain and smart contracts)

Outsmarting Smart Contracts

https://youtu.be/EKU8T58kYCw

securing

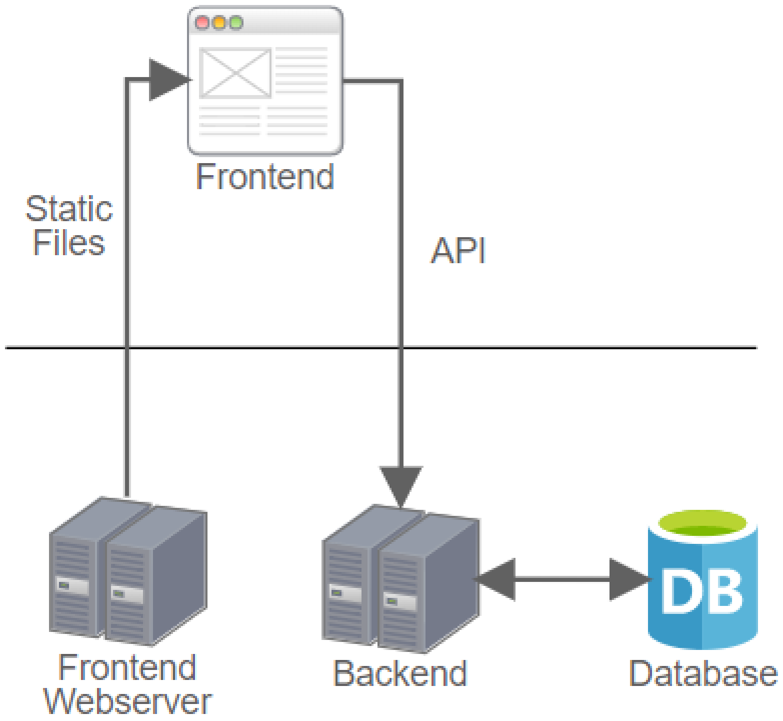
Senior Security Consultant
Security Researcher

UMCS

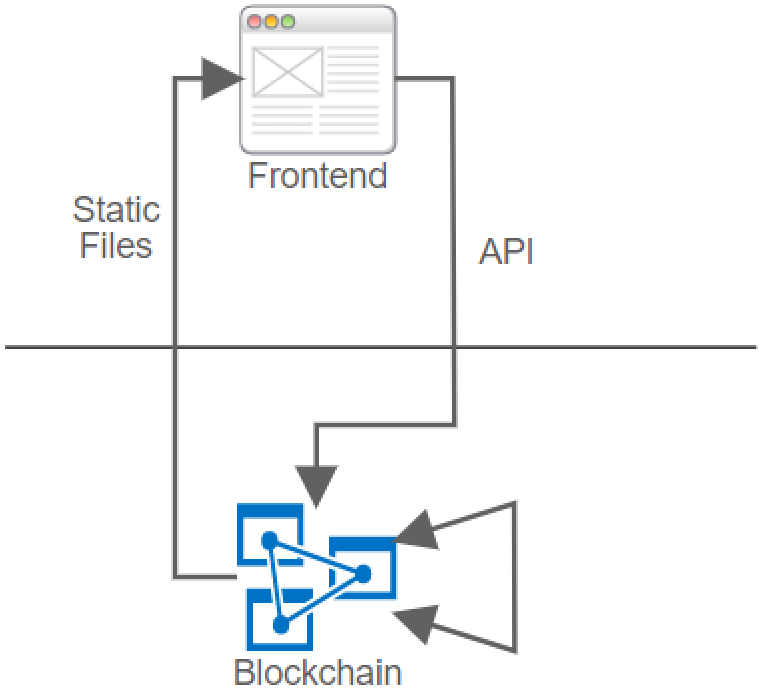
Assistant Professor

Where is the main difference?
Architecture

Web Application

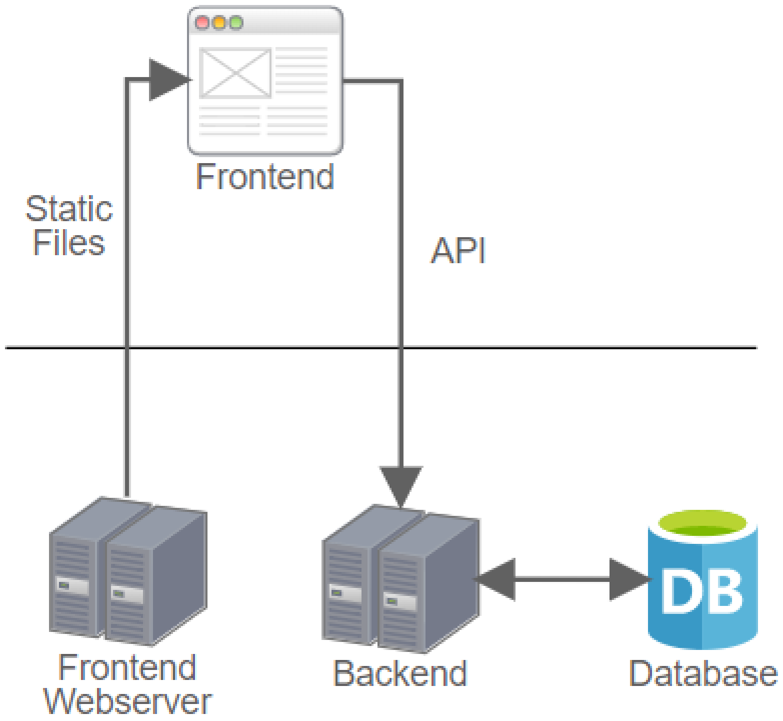


Decentralized Application

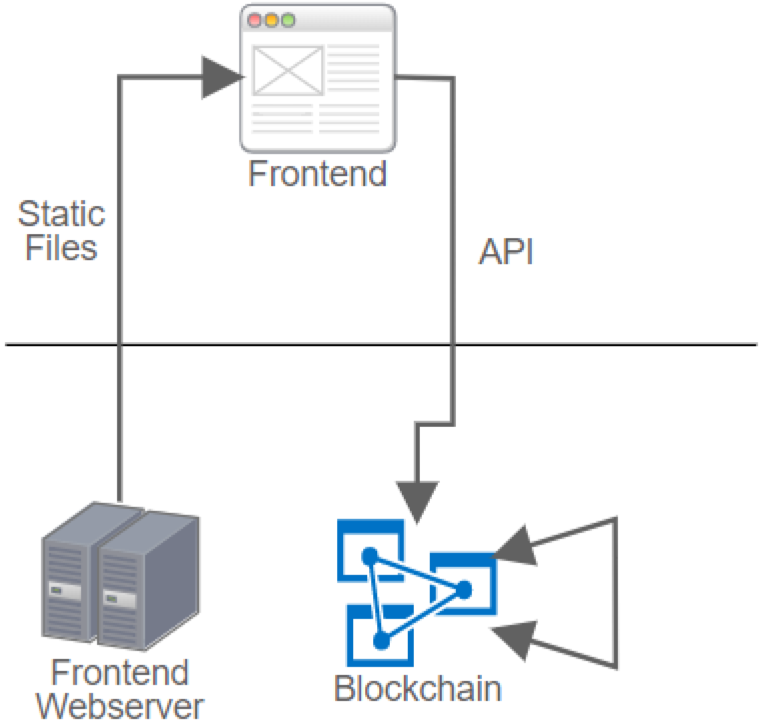


Where is the main difference? Architecture

Web Application



Hybrid Decentralized Application



What is so special about Decentralized Apps?

- **Trustlessness:** Use blockchain to store code and data (state).
- No one can turn it off permanently (anyone can bring it to live).
- Everyone can have it (like keeping the database of FB or Reddit locally).

Decentralized Apps

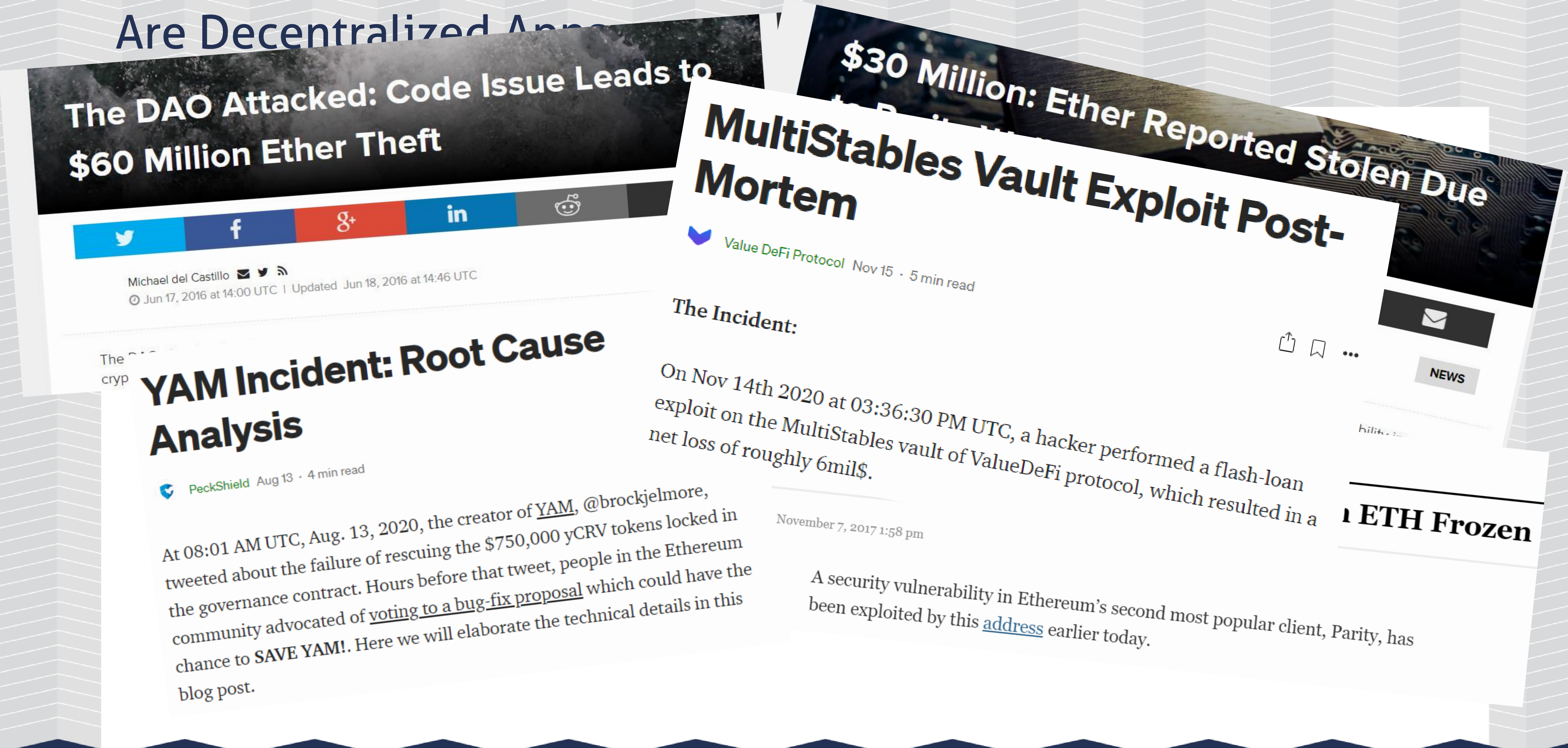
ARE THOSE SECURE?

Are Decentralized Apps secure?

- **Indestructible**: No one can turn it off
- **Cryptographically secure**: All transactions are digitally signed
- **Publicly verifiable**: Anyone can verify the code of smart contracts
- But still….

drdr_zz

Are Decentralized App



The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

Michael del Castillo

Jun 17, 2016 at 14:00 UTC | Updated Jun 18, 2016 at 14:46 UTC

The

cryp

$30 Million: Ether Reported Stolen Due

NEWS

MultiStables Vault Exploit Post-Mortem

Value DeFi Protocol Nov 15 · 5 min read

The Incident:

On Nov 14th 2020 at 03:36:30 PM UTC, a hacker performed a flash-loan exploit on the MultiStables vault of ValueDeFi protocol, which resulted in a net loss of roughly 6mil$.

YAM Incident: Root Cause Analysis

PeckShield Aug 13 · 4 min read

At 08:01 AM UTC, Aug. 13, 2020, the creator of YAM, @brockjelmore, tweeted about the failure of rescuing the $750,000 yCRV tokens locked in the governance contract. Hours before that tweet, people in the Ethereum community advocated of voting to a bug-fix proposal which could have the chance to **SAVE YAM!**. Here we will elaborate the technical details in this blog post.

ETH Frozen

November 7, 2017 1:58 pm

A security vulnerability in Ethereum's second most popular client, Parity, has been exploited by this address earlier today.

drdr_zz

The D

$60 M

Micha

Ju

The DAO

109 #672664

Sta

Disclos

Reported

A

Weak

Bo

31 #377398 A

State

Disclosed

Reported To

Asset h (S

Weakness Bu

Bounty $5,

ning DSK inte

Severity High

Participants

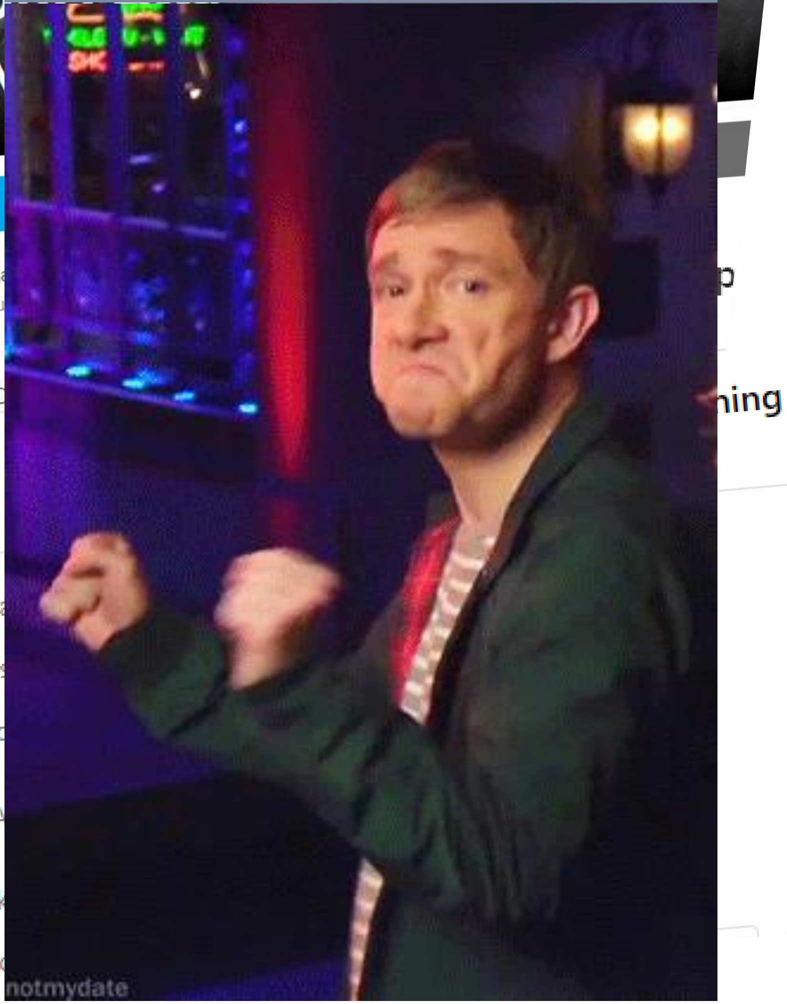
Visibility Disclosed

High (7 ~ 8.9)

ed (Full)

Frozen

Expectations



Reality

Web Apps vs Decentralized Apps

WE NEED SECURITY!

Security needs

Technical

- Build secure applications.
  - Omit the insecure patterns.
- Find ane remediate the security bugs (vulnerabilities).

Business

- Make sure that the application is secure.
- The status: List of green and red points.

Security Projects & Standards

Web Apps

- Most common vulnerabilities?
  - **OWASP Top 10**
- The end to end security checklist to perform an audit?
  - **OWASP ASVS**
    Application Security Verification Standard

Decentralized Apps

- Most common vulnerabilities?
  - **DASP Top 10** (https://dasp.co)
- The end to end security checklist to perform an audit?

Smart Contracts
Security Verification Standard

SCSVS

- Smart Contracts Security Verification Standard

drdr_zz

wh01s7

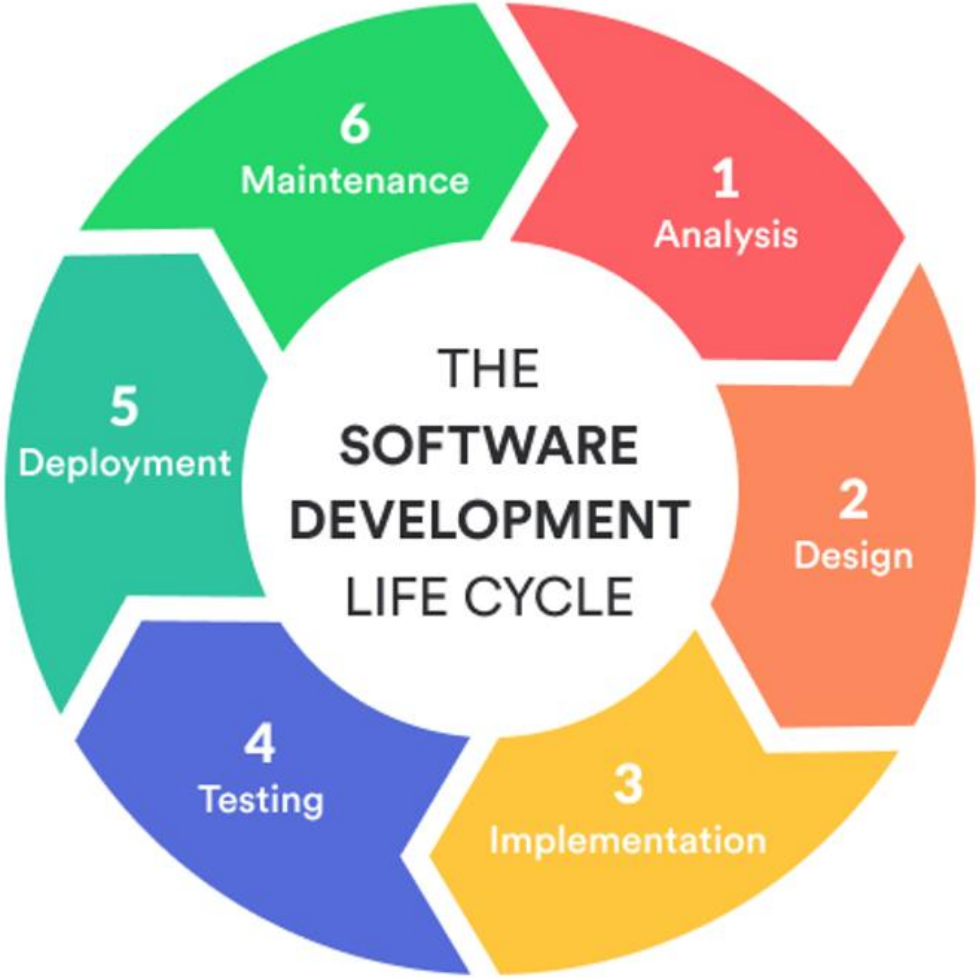
SCSVS - Objectives

- Objectives:
  - A checklist for architects, developers and security reviewers.
- Technical needs
  - Help to mitigate known vulnerabilities by design.
  - Help to develop high quality code of the smart contracts.
- Business needs
  - Provide a clear and reliable assessment of how secure the smart contract is in relation to the percentage of SCSVS coverage.
- ~~13~~ 14 categories of security requirements.
- Format similar to ASVS.

Smart Contracts
Security Verification Standard

Software Development Life Cycle

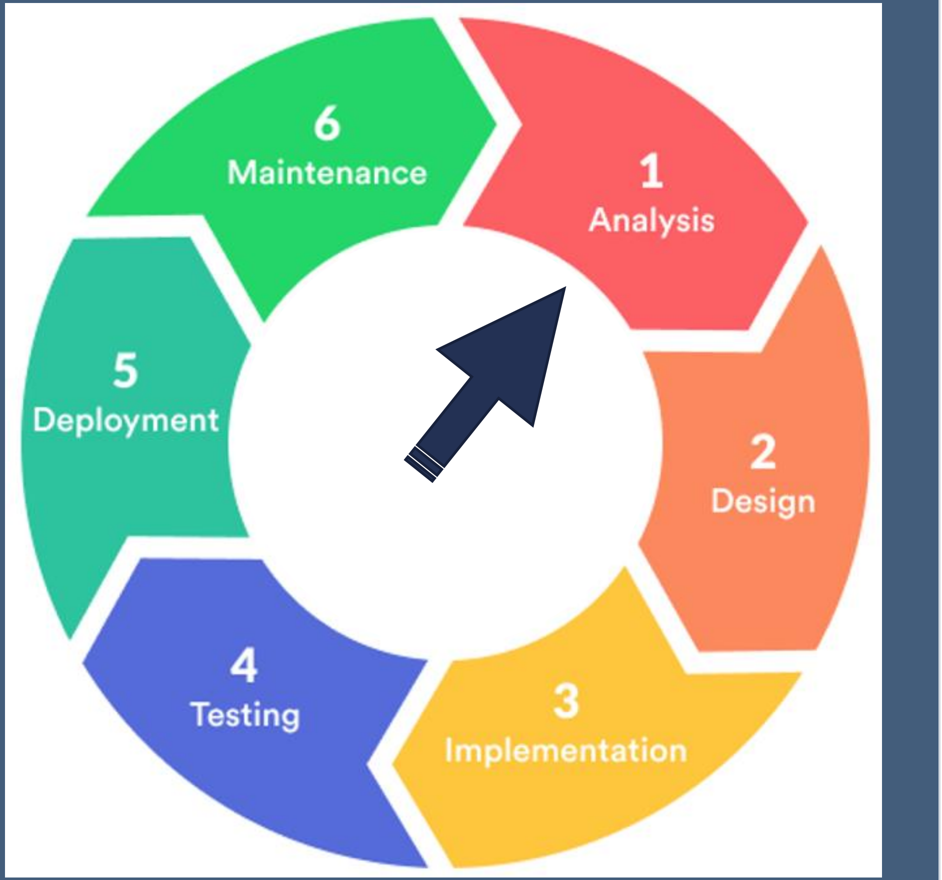
SCSVS covers all stages of SDLC process.



Web Apps vs Decentralized Apps

SDLC

- Analysis & Requirements



SDLC – Analysis & Requirements

Similiarities

- Threat modelling

Smart Contracts
Security Verification Standard

1.1 Verify that the every introduced design change is preceded by an earlier threat modelling.

Smart Contracts
Security Verification Standard

1.2 Verify that the documentation clearly and precisely defines all trust boundaries in the contract (trusted relations with other contracts and significant data flows).

SDLC – Analysis & Requirements

Differences – Sensitive data

Web Apps

- Stored in protected database

Decentralized Apps

- Stored on public blockchain
    - Forever
    - Anyone can read

Smart Contracts
Security Verification Standard

3.1 Verify that any data saved in the contracts is not considered safe or private (even private variables).

Smart Contracts
Security Verification Standard

3.2 Verify that no confidential data is stored in the blockchain (passwords, personal data, token etc.).

SDLC – Analysis & Requirements

Differences – Randomness

Web Apps

- A matter of a function call

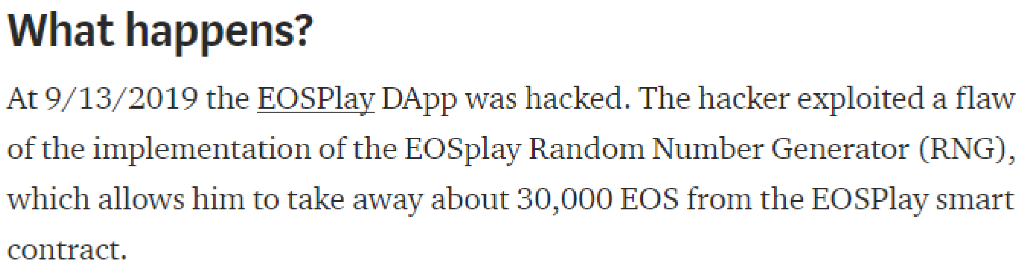
Decentralized Apps

- Not trivially achieved in the decentralized computer
- No local parameters can be used

SDLC – Analysis & Requirements

Differences – Randomness

- EOSPlay hack
  - 30k EOS stolen



What happens?

At 9/13/2019 the EOSPlay DApp was hacked. The hacker exploited a flaw of the implementation of the EOSplay Random Number Generator (RNG), which allows him to take away about 30,000 EOS from the EOSPlay smart contract.

- SmartBillions Lottery hack
  - 400 ETH stolen
  - https://bit.ly/2jJEKPd

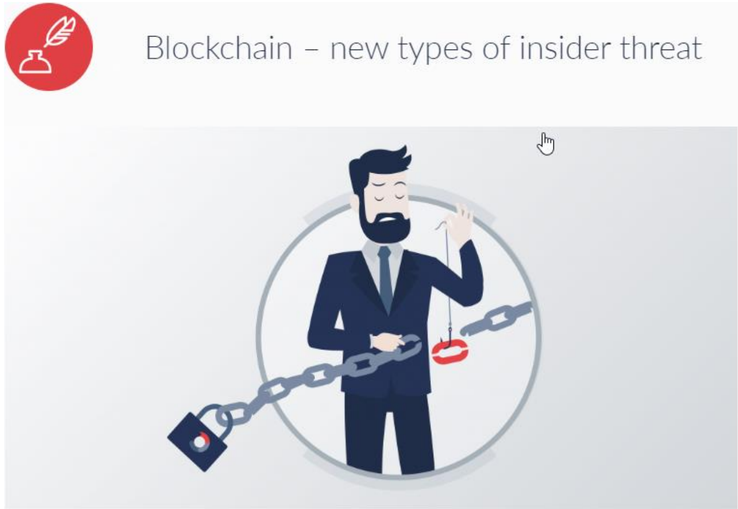
Smart Contracts Security Verification Standard

7.5 Verify that the contract does not generate pseudorandom numbers trivially basing on the information from blockchain (e.g. seeding with the block number).

SDLC – Requirements & Analysis

New threat actors for Decentralized Apps

- Miners/Validators
  - Validate transactions and add new blocks



Blockchain – new types of insider threat

SDLC – Requirements & Analysis

New threat actors for Decentralized Apps

Smart Contracts Security Verification Standard

8.1 Verify that the contract logic implementation corresponds to the documentation.

Smart Contracts Security Verification Standard

8.3 Verify that the contract has business limits and correctly enforces it.

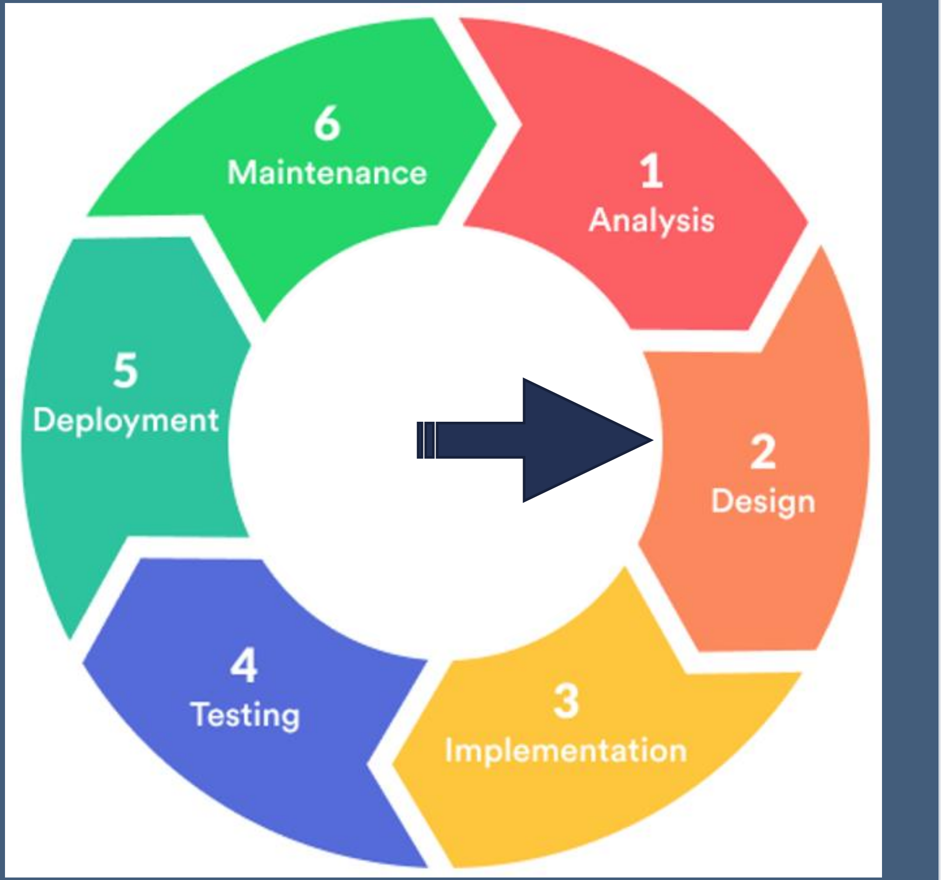
Smart Contracts Security Verification Standard

9.3 Verify that the contract logic does not disincentivize users to use contracts (e.g. the cost of transaction is higher than the profit).

Web Apps vs Decentralized Apps

SDLC

- Design



SDLC – Design

Similiarities

- Least privilege rule
- Access control
  - Public and known to everyone
  - Centralized and simple

Smart Contracts
Security Verification Standard

2.3 Verify that the creator of the contract complies with the rule of least privilege and his rights strictly follow the documentation.

Smart Contracts
Security Verification Standard

2.11 Verify that all user and data attributes used by access controls are kept in trusted contract and cannot be manipulated by other contracts unless specifically authorized.

SDLC – Design

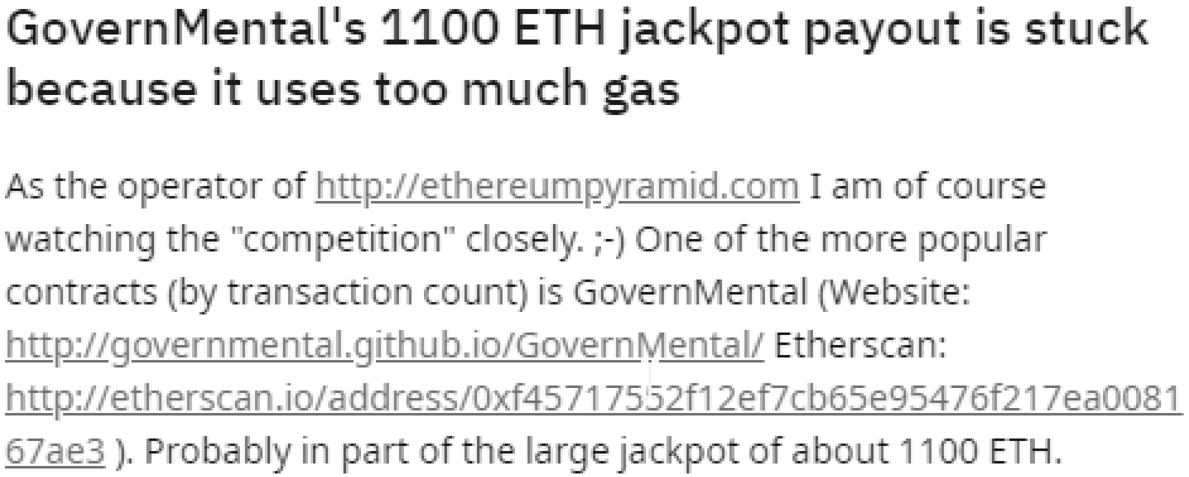
Differences – Loops

Web Apps	Decentralized Apps
• Infinite loops -> DoS	• Unbound loops -> DoS

SDLC – Design

Differences – Loops

- GovernMentals
  - A ponzi scheme
  - Iteration over a huge array
  - 1100 ETH frozen
  - https://bit.ly/2kVXwaj



GovernMental's 1100 ETH jackpot payout is stuck because it uses too much gas

As the operator of http://ethereumpyramid.com I am of course watching the "competition" closely. ;-) One of the more popular contracts (by transaction count) is GovernMental (Website: http://governmental.github.io/GovernMental/ Etherscan: http://etherscan.io/address/0xf45717552f12ef7cb65e95476f217ea008167ae3). Probably in part of the large jackpot of about 1100 ETH.

7.3 Verify that the contract does not iterate over unbound loops.

8.8 Verify that the contract does not send funds automatically but it lets users withdraw funds on their own in separate transaction instead.

SDLC – Design

Decreasing the risk

- Decentralized Applications keep cryptocurrencies
- The higher the amount the bigger the incentive for hackers

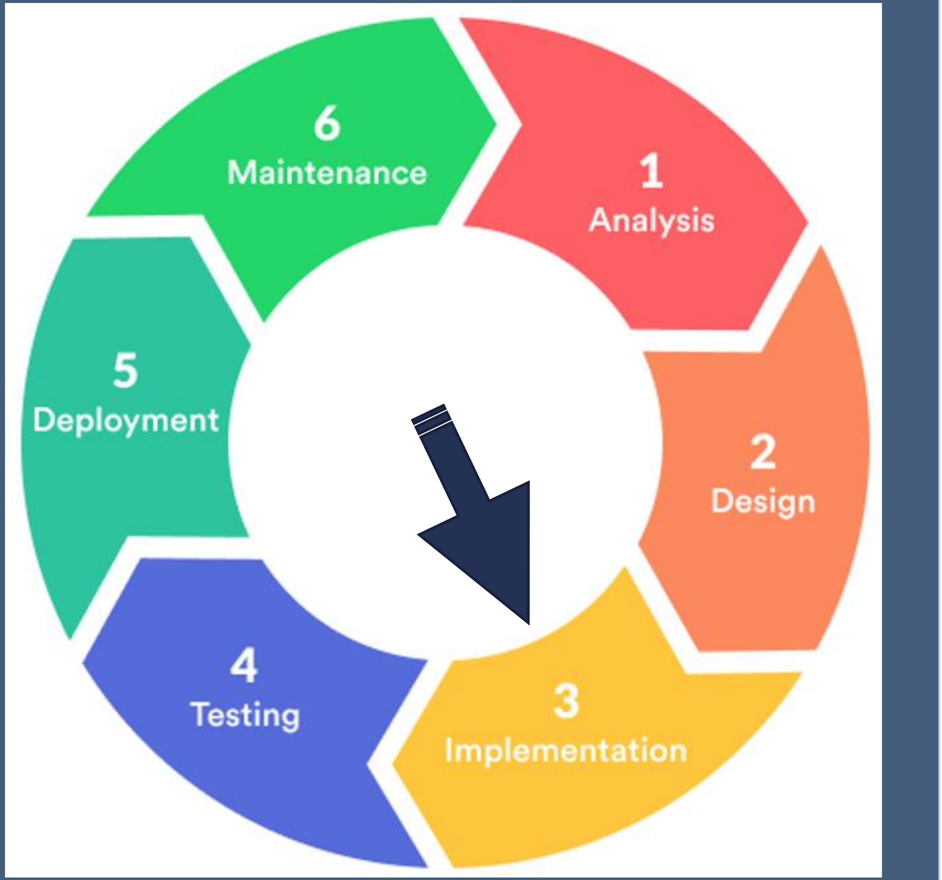
Smart Contracts
Security Verification Standard

1.8 Verify that the amount of cryptocurrencies kept on contract is controlled and at the minimal acceptable level.

Web Apps vs Decentralized Apps

SDLC

- Implementation



SDLC – Implementation

- Great tools

TRUFFLE remix Ethereum Studio Brownie

  - Perform basic security analysis

- But we still make bugs.
- Sounds familiar? ☺

SDLC – Implementation

Similarities – Arithmetic bugs

Web Apps

- Not that common

Decentralized Apps

- Overflows and underflows

SDLC – Implementation

Similarities – Arithmetic bugs

- Multiple ERC20 Smart Contracts
  - Allow to transfer more than decillions (10^60) of tokens
  - https://bit.ly/2lWa9ma
  - https://bit.ly/2ksNEF1

SDLC – Implementation

Similarities – Arithmetic bugs

Smart Contracts
Security Verification Standard

5.1 Verify that the values and math operations are resistant to integer overflows. Use SafeMath library for arithmetic operations.

Smart Contracts
Security Verification Standard

5.2 Verify that the extreme values (e.g. maximum and minimum values of the variable type) are considered and does change the logic flow of the contract.

Smart Contracts
Security Verification Standard

5.3 Verify that non-strict inequality is used for balance equality.

SDLC – Implementation

Differences – Recursive calls

Web Apps

- Must be explicitly included in the logic

Decentralized Apps

- Executing some logic multiple times in one call
- The DAO hack
  - Recursive withdrawals
  - 3.6 mln ETH stolen
  - BQjKq

Smart Contracts Security Verification Standard

4.5 Verify that re-entrancy attack is mitigated by blocking recursive calls from other contracts. Do not use call and send function unless it is a must.

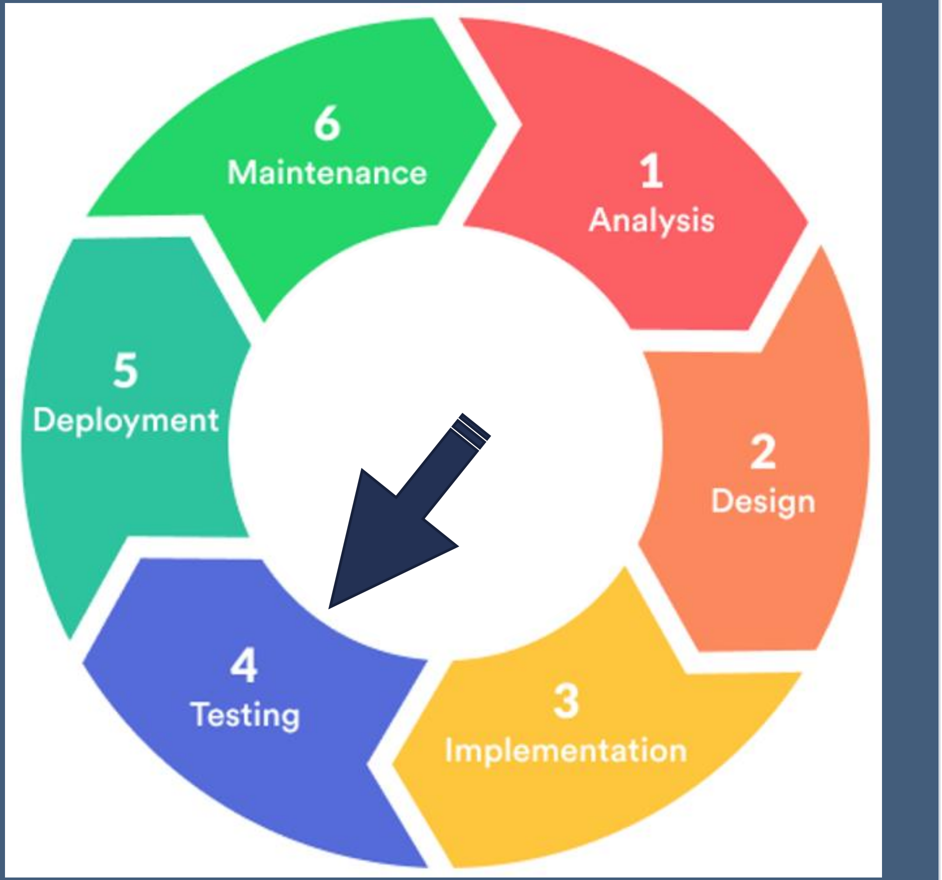
Smart Contracts Security Verification Standard

4.6 Verify that the result of low-level function calls (e.g. send, delegatecall, call) from another contracts is checked.

Web Apps vs Decentralized Apps

SDLC

- Testing



SDLC – Testing

Similarities – Great tools for automatic scans

Web Apps

Decentralized Apps

SECURIFY
https://securify.ch

Solhint
https://bit.ly/2mpaL3U

SmartCheck
https://tool.smartdec.net

MythX
https://mythx.io/

Smart Contracts
Security Verification Standard

1.11 Verify that code analysis tools are in use that can detect potentially malicious code.

SDLC – Analysis & Requirements

Similiarities – Ensuring the testing takes place

Smart Contracts Security Verification Standard

12.1 Verify that all functions of verified contract are covered with tests in the development phase.

Smart Contracts Security Verification Standard

12.2 Verify that the implementation of verified contract has been checked for security vulnerabilities using static and dynamic analysis.

Smart Contracts Security Verification Standard

12.3 Verify that the specification of smart contract has been formally verified.

Smart Contracts Security Verification Standard

12.4 Verify that the specification and the result of formal verification is included in the documentation.

- including manual security tests

Smart Contracts Security Verification Standard

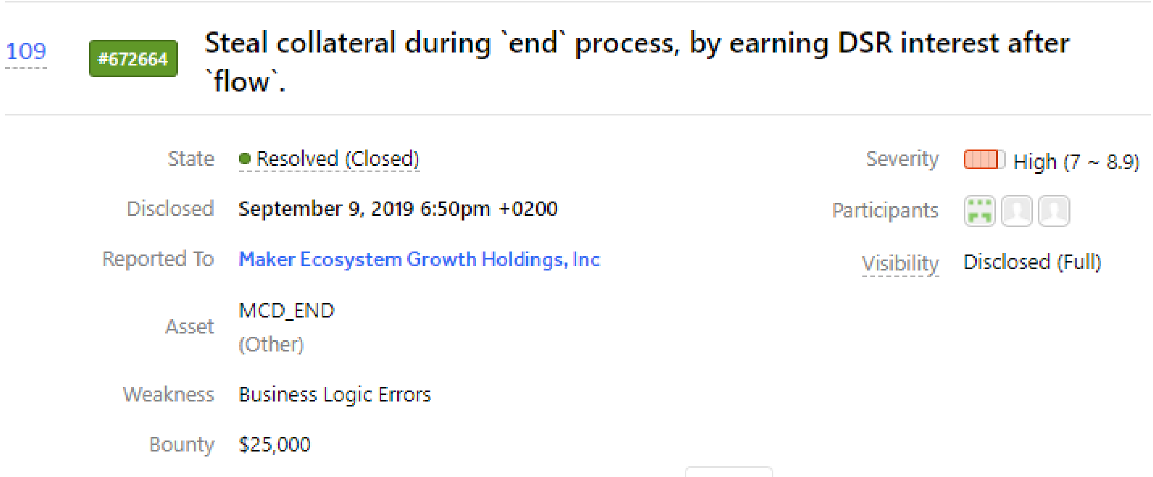
1.3 Verify that the SCSVS, security requirements or policy is available to all developers and testers.

SDLC – Analysis & Requirements

Similiarities – Business logic errors

- Hard to find using automated scans
- MakerDAO vulnerability
  - Allows to create DAI cryptocurrency without coverage
  - 25k $ bounty

https://hackerone.com/reports/672664



109 #672664 Steal collateral during `end` process, by earning DSR interest after `flow`.

State	● Resolved (Closed)	Severity	High (7 ~ 8.9)
Disclosed	September 9, 2019 6:50pm +0200	Participants	
Reported To	Maker Ecosystem Growth Holdings, Inc	Visibility	Disclosed (Full)
Asset	MCD_END (Other)		
Weakness	Business Logic Errors		
Bounty	$25,000		

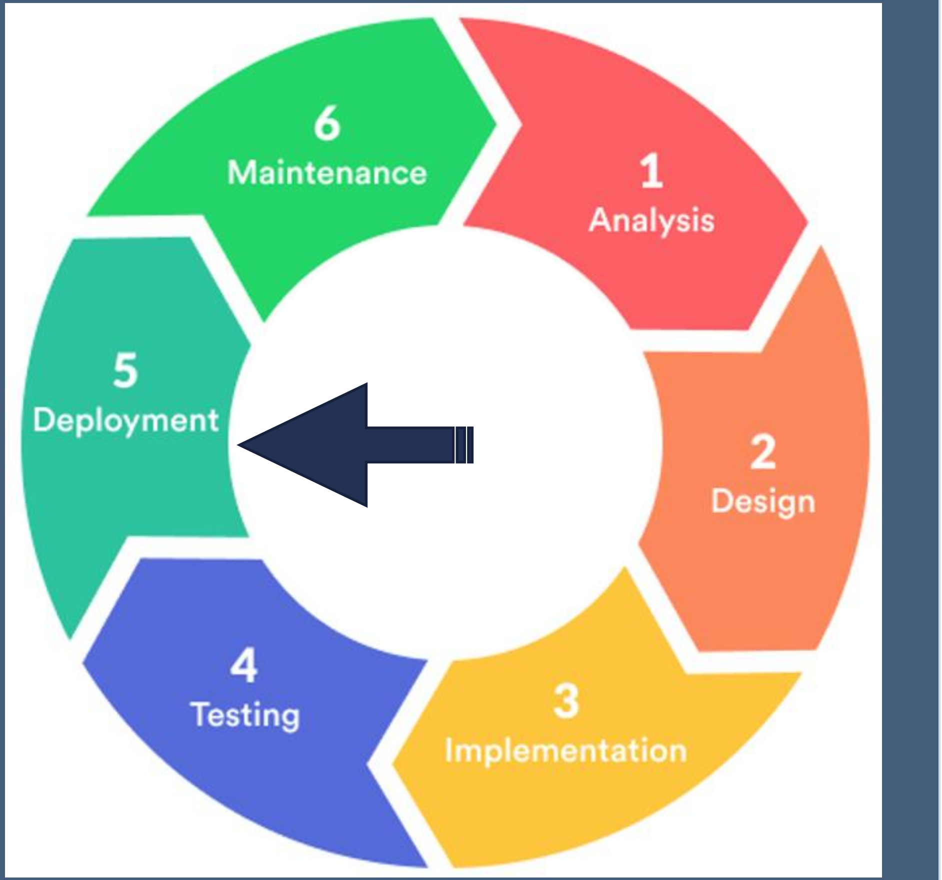
1.10 Verify that the business logic in contracts is consistent. Important changes in the logic should be allowed for all or none of the contracts.

8.2 Verify that the business logic flows of smart contracts proceed in a sequential step order and it is not possible to skip any part of it or to do it in a different order than designed.

Web Apps vs Decentralized Apps

SDLC

- Deployment



SDLC – Deployment

Differences – Initialization stage

Web Apps

- Setting up configurations and integrations
- Performed once during deployment

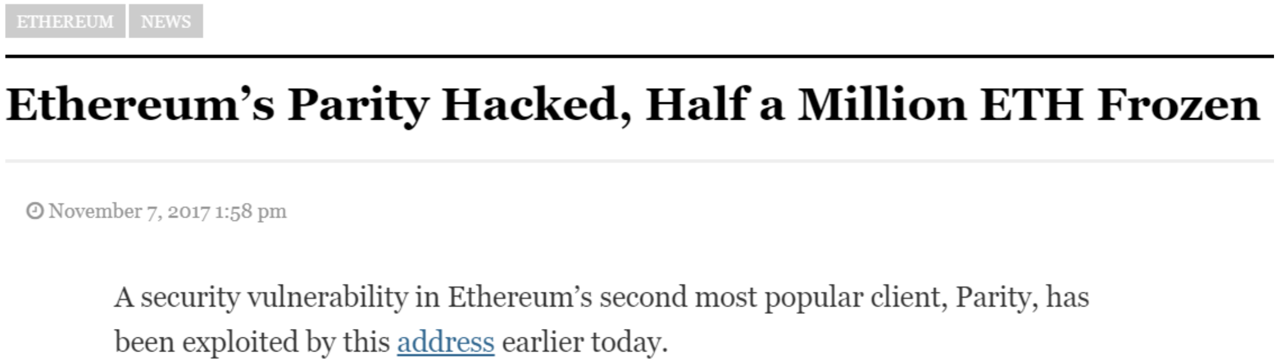
Decentralized Apps

- Setting up configurations and integrations
- What if one can (re-)initialize the contract?

SDLC – Deployment

Differences – Initialization stage

- Parity Wallet hack:
    - Kill contract shared by hundreds of other contracts
    - 500k ETH frozen
    - https://bit.ly/2kIBYhA
    - https://bit.ly/2kpfKkm



ETHEREUM NEWS

Ethereum's Parity Hacked, Half a Million ETH Frozen

November 7, 2017 1:58 pm

A security vulnerability in Ethereum's second most popular client, Parity, has been exploited by this address earlier today.

SDLC – Deployment

Differences – Initialization stage

Smart Contracts Security Verification Standard — 11.7 Verify that all storage variables are initialised.

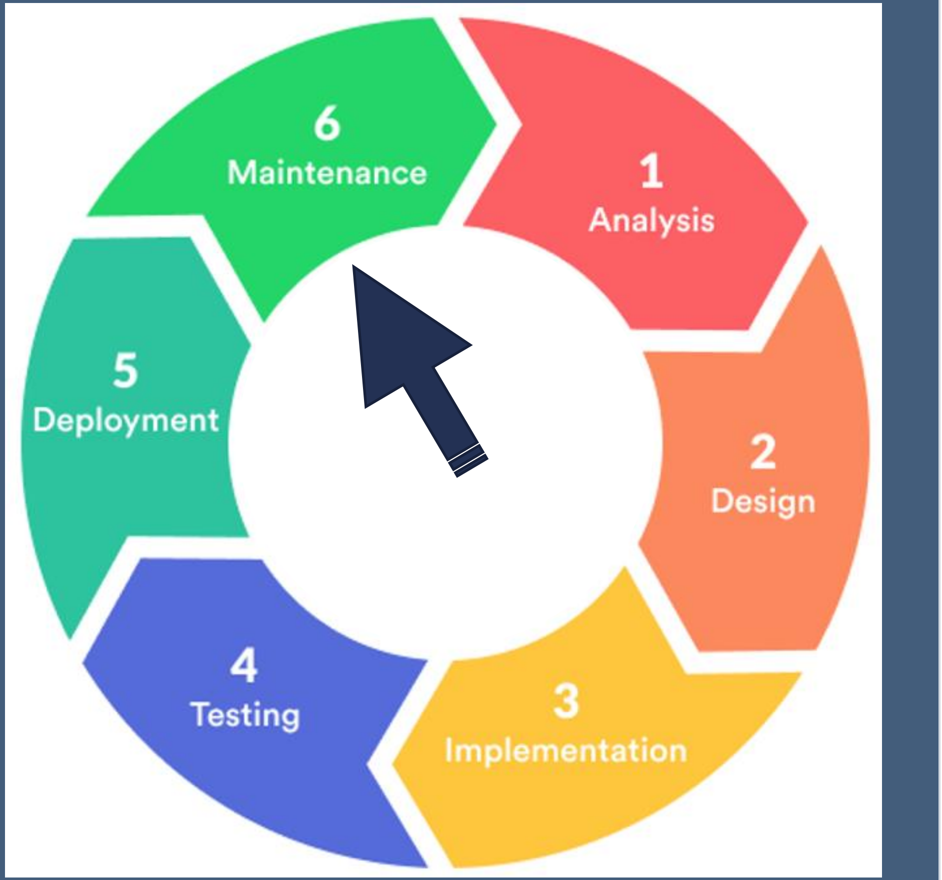
Smart Contracts Security Verification Standard — 2.8 Verify that the initialization functions are marked internal and cannot be executed twice.

Smart Contracts Security Verification Standard — 9.1 Verify that the self-destruct functionality is used only if necessary.

Web Apps vs Decentralized Apps

SDLC

- Maintenance



SDLC – Analysis & Requirements

Differences – Security Alert and Fix

Web Apps

- Application goes down
- The bug is fixed (patch)
- Application redeployed

Decentralized Apps

- ~~Smart contract goes down~~
- The bug is fixed (patch)
- Smart contract deployed again

Smart Contracts Security Verification Standard

1.6 Verify that there exists a mechanism that can temporarily stop the sensitive functionalities of the contract in case of a new attack. This mechanism should not block access to the assets (e.g. tokens) for the owners.

Smart Contracts Security Verification Standard

1.4 Verify that there exists an upgrade process for the contract which allows to deploy the security fixes.

Smart Contracts
Security Verification Standard

SCSVS – NEW CATEGORY!

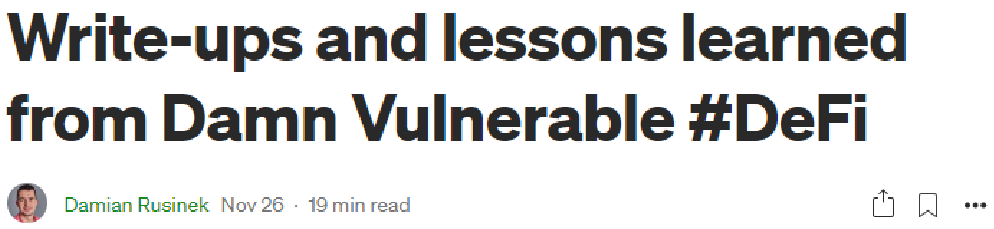
Decentralized Finance Security Requirements!

drdr_zz

wh01s7

Decentralized Finance category

- Security requirements for:
  - lending pools,
  - flash loans,
  - governance,
  - on-chain oracles,
  - etc.



Write-ups and lessons learned from Damn Vulnerable #DeFi

Damian Rusinek Nov 26 · 19 min read

Smart Contracts Security Verification Standard

14.1 Verify that the lender's contract does not assume its balance (used to confirm loan repayment) to be changed only with its own functions.

Smart Contracts Security Verification Standard

14.6Verify that the rewards cannot be calculated and distributed within the same function call that deposits tokens. That protects from the momentary fluctuations in shares.

Smart Contracts Security Verification Standard

14.11 Verify that the complex math operations that consist of both multiplication and division operations firstly perform the multiplications and then division.

Security Projects & Standards

Web Apps

- Most common vulnerabilities?
  - **OWASP Top 10**
- The end to end security checklist to perform an audit?
  - **OWASP ASVS (Application Security Verification Standard)**

Decentralized Apps

- Most common vulnerabilities?
  - **DASP Top 10** (https://dasp.co)
- The end to end security checklist to perform an audit?

SCSVS

SCSVS meets your security needs

Technical

- Build secure applications.
  - Omit the insecure patterns.
- Find ane remediate the security bugs (vulnerabilities).

Business

- Make sure that the application is secure.
- The status: List of green and red points.

Want to develop secure smart contracts?

Want a security audit of smart contract?

Go for SCSVS!

Intrested in
the Smart Contracts
Security Training?
Sing up!

Ok, Thank you! drdr_zz

Damian.Rusinek@securing.pl