Securing the Unknown

A Methodology for Auditing Smart Contracts

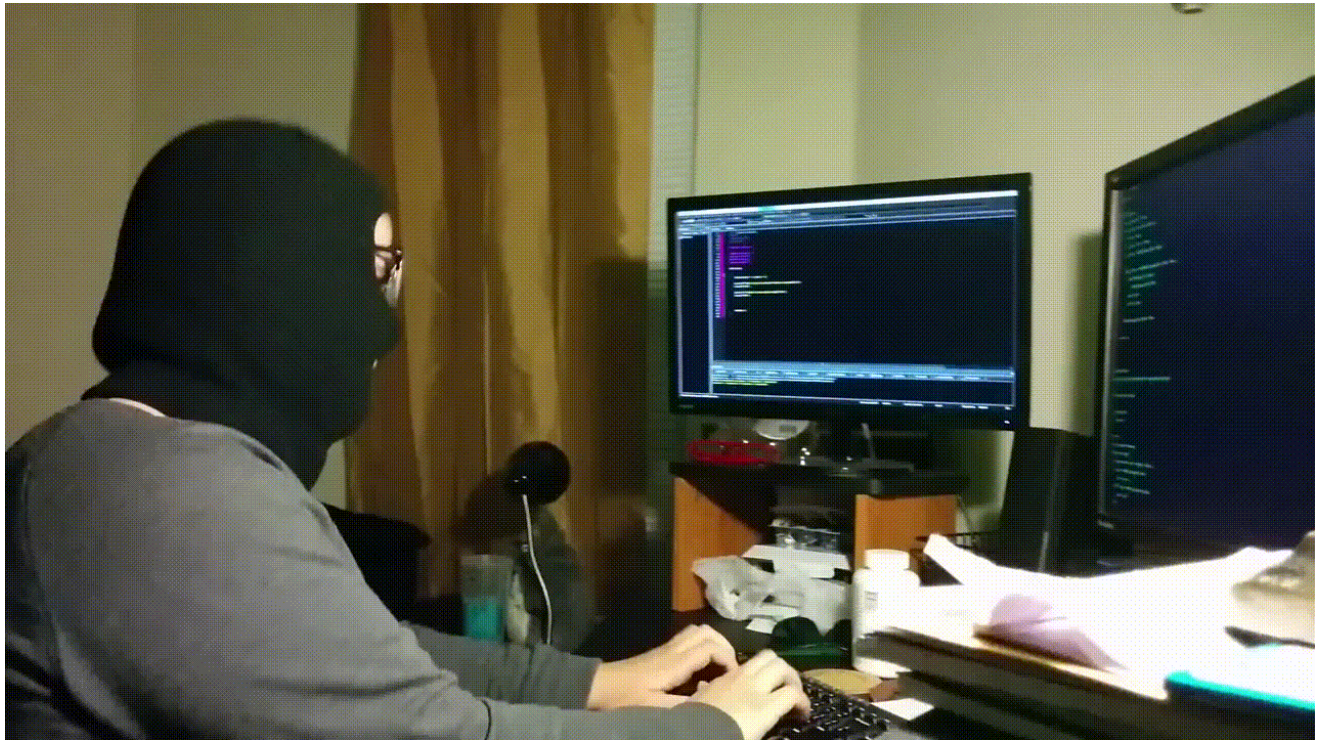
SECURITY INNOVATION

About Me

Ben Stewart

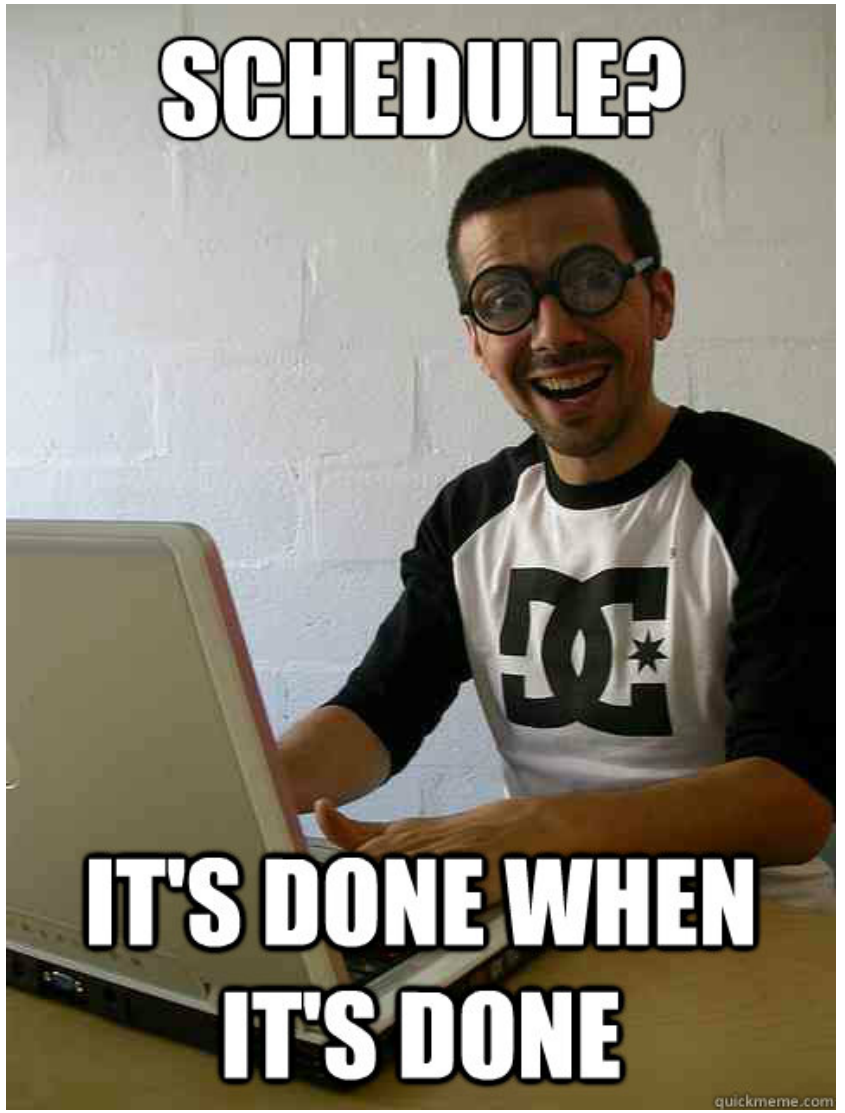
@bigbenstew

- Security engineer & researcher
- Automation aficionado
- BSAB



Agenda

- Vulnerability landscape
- Methodology 101
- **U**nderstand use case
- **M**anual review
- **A**utomated tools
- **C**ommunication
- **A**nalysis
- **R**eporting



Vulnerability Landscape

AppSec Specific Vulnerabilities

OWASP Top 10 Application Security Risks

- Broken authentication mechanisms
- Using components with known vulnerabilities
- Insufficient logging and monitoring
- Insecure business logic
- Insufficient cryptography strength

Common Weakness Enumeration – CWEs

- Common software weaknesses that can occur in software's architecture, design, code or implementation that can lead to exploitable security vulnerabilities.

Blockchain Specific Vulnerabilities

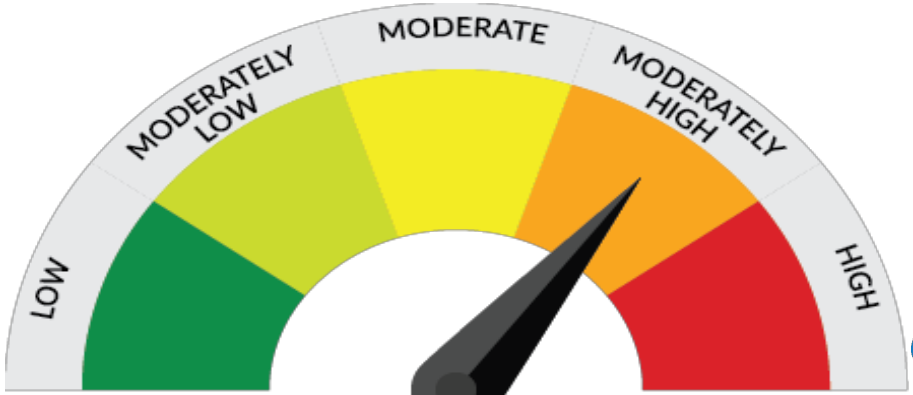
- SWC Registry
  - Function Default Visibility [SWC-100]
  - Integer Overflow and Underflow [SWC-101]
  - Outdated Compiler Version [SWC-102]
  - Shadowing State Variables [SWC-119]
  - Presence of unused variables [SWC-131]

The entire list can be found on the SWC Registry Github

End Result

But seriously…

- One study found that upwards of **45%** of smart contracts written in Solidity had some form of vulnerability.
- Challenges arise from the **non-standard lifecycle** of smart contracts which differs from software that can just be updated or patched
- The impact of security vulnerabilities can be **immediate** and **devastating** for the developer, organization, and ecosystem.



Methodology 101

What is a methodology?

"A system of methods used in a particular area of study or activity."

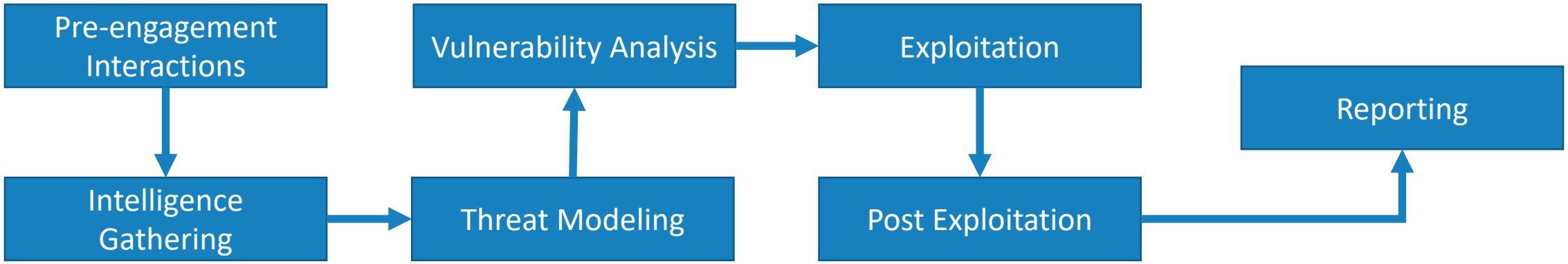
-Wikipedia

Examples:

- Making a peanut butter and jelly sandwich
- The scientific method
- Extreme programming or test driven development
- Anything…

Methodologies exist in security as well

- Penetration Testing Execution Standard (PTES)
- PCI Penetration testing guide
- Information Systems Security Assessment Framework (ISSAF)
- Open Source Security Testing Methodology Manual (OSSTMM)



UMACAR? Yes!

- **U**nderstand use case
- **M**anual review
- **A**utomated tools
- **C**ommunication
- **A**nalysis
- **R**eporting

Real World Example: Metacash

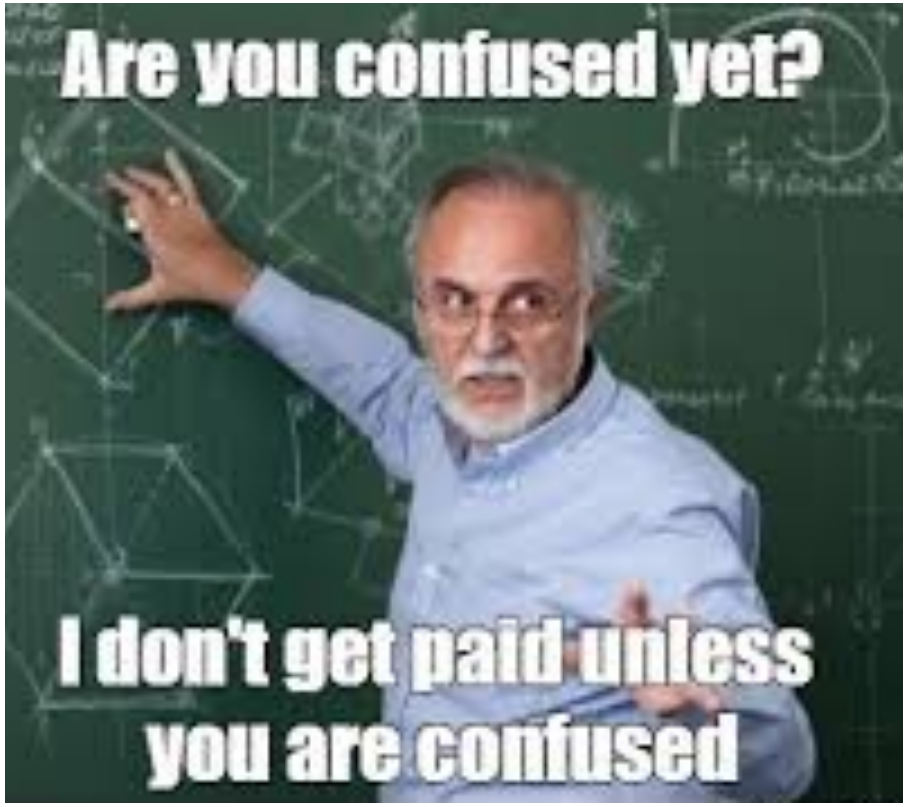
What is Metacash?

A gasless non-custodial DAI wallet

SECURITY INNOVATION

How does it work?

- *"An alternative wallet architecture that combines meta-transactions, smart contract wallets, and the new Ethereum Constantinople CREATE2 opcode to replace the Ethereum external account model"*
- *"Instead of creating **token standards**, we can create **smart wallet standards**."*
- "We can also put meta-transaction functionality at the **wallet layer** instead of the **token contract layer**."



Understand use case

Open Source Intelligence (OSINT)

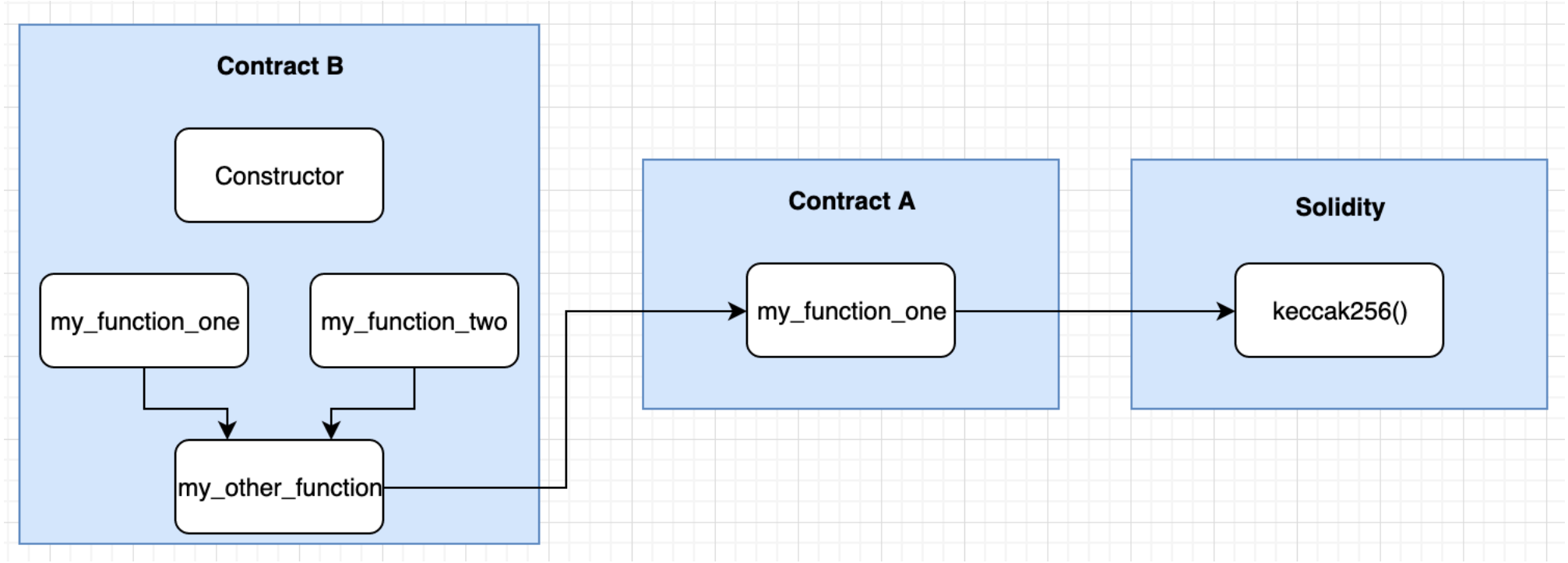
- Search up a storm
  - Google, Github, Medium, Reddit, etc
- Gather relevant assets
  - Source code, test net implementations or other interfaces
- Interact with available interfaces
  - Test net, Dapps, Android wallet, other

Importance of understanding the use case

- Business logic & requirements
  - Understand what belongs and what doesn't
  - Prevent future attacks
- A threat model or attack surface analysis can help guide attack
  - Assumptions
    - e.g. Android and iOS wallet applications are out of scope
  - Assets
    - e.g. Wallet Assets: Ether or ERC20 tokens
  - Risks
    - e.g. An anonymous party can steal or lock user funds in the wallet

Identifying Areas of Opportunity

- Call graphs, inheritance graphs, other visualizations..



Extra Credit: Establish Point of Contact

- Improves efficiency at which you can understand a system
- Access to additional and/or non-public information
- Insight into the minds behind the madness ☺

Manual Review

Code Review Guide

Line by line, little by little, step by step

- How many contracts?
  - How many of those are boilerplate?
  - Can you describe the purpose of each one?
- What functions are available and to whom?
- Is the code well commented?
- Does the code smell?
- Is this how you would do it?

Importance of test cases

- CWEs to SWCs is a good start
- Guide the attack and stay on track
- Custom test cases
  - Business logic
  - Necessary vs. unnecessary functionality
  - Ensure coverage
- Scope creep is real
- Hard to automate everything

Gas as an attack vector

- Gas brings an additional layer of logic to contract code execution
- Attack Examples
  - Attacker causes you to send a transaction and that transaction consumes a lot of gas
    - Mitigation: Always set reasonable gas limit
  - Attacker exceeds block gas limit resulting in a denial of service
    - Mitigations: Avoid loops when possible
- The EVM operation code table may help grasp the idea of gas limits.

Back to Metacash

SmartWallet

This contract is deployed once and is to be used as a library by individual Proxies through delegatecall. It contains all the functionality of the wallet and stores two additional fields in the storage array:

`owner` - the user of the wallet

`nonce` - a incrementing uint used for replay protection

SmartWallet defines the following functions:

initiate(address owner)

- Called only by Factory to initialize owner and nonce

initiate(address owner, address relay, uint fee, address token)

- Initializes owner and nonce and pays a fee to a relayer

Automated Tools

Static Analysis

One of my favorite Solidity static text analyzers is Slither.

1. Install Slither:
   - `pip install slither-analyzer`
2. Run Full Test Suite:
   - `slither contracts/smartwallet.sol`
3. Unused Return Value:
   - `slither contracts/smartwallet.sol --detect unused-return`

Improving The Workflow

- Truffle for the Win
  - Version 5.0.0+
    - Indicate compiler version in truffle.js
  - Downloads the required compiler and launches
  - Change versions and test easily
- Other ways to use different versions of solc with analyzers
  - Replacing /usr/bin/solc in Docker file
  - [LAST RESORT] /usr/bin/solc in current system
- Contract Flatteners with Remix IDE
  - https://github.com/nomiclabs/truffle-flattener

Dynamic Analysis

- Mythril
  - Symbolic execution, SMT solving and taint analysis to detect a variety of security vulnerabilities.
  - EVM-compatible blockchains (Quorum, VeChain, Tron)
- MythX
  - Security analysis as a service
  - Truffle plugin
- Manticore
  - Symbolic execution tool for analysis of smart contracts and binaries
  - CLI or API

Extra Credit – Automation

- Learn from the Scrooge McEtherFace Auto Loader

1. Obtain the contract bytecode by compiling a Solidity file or loading it from an Ethereum node.
2. Initialize the contract account state either by running the creation bytecode (if source code was provided) or by retrieving data from an Ethereum node on-demand.
3. Symbolically execute the code to explore all possible program states over n transactions, whereby n defaults to two but can be set to an arbitrary number;
4. Whenever an undesirable states is encountered (such as “the contract kills itself”), logically prove or disprove the reachability of those states given certain assumptions (e.g. “given the right inputs anybody can kill the contract“).

Communication

Stay Plugged In

- Don’t hide under a rock
  - Research hunches
  - Stay up to date on relevant cybersecurity news
- If you have a point of contact
  - Ask questions
  - See something, say something
- If you don’t have a point of contact
  - Find someone you can ask questions to or use as sounding board

Analysis

Vulnerability Analysis

Goal = Discover flaws which could be leveraged by an attacker

- Testing should be focused and efficient
  - Was proper coverage achieved? (Depth & Breadth)
  - Use your knowledge of the system and methodically work through scenarios
- Unleash your inner hacker!
- Proof-of-concept Development
  - Time permitted
  - Remix is a great tool for rapid prototyping

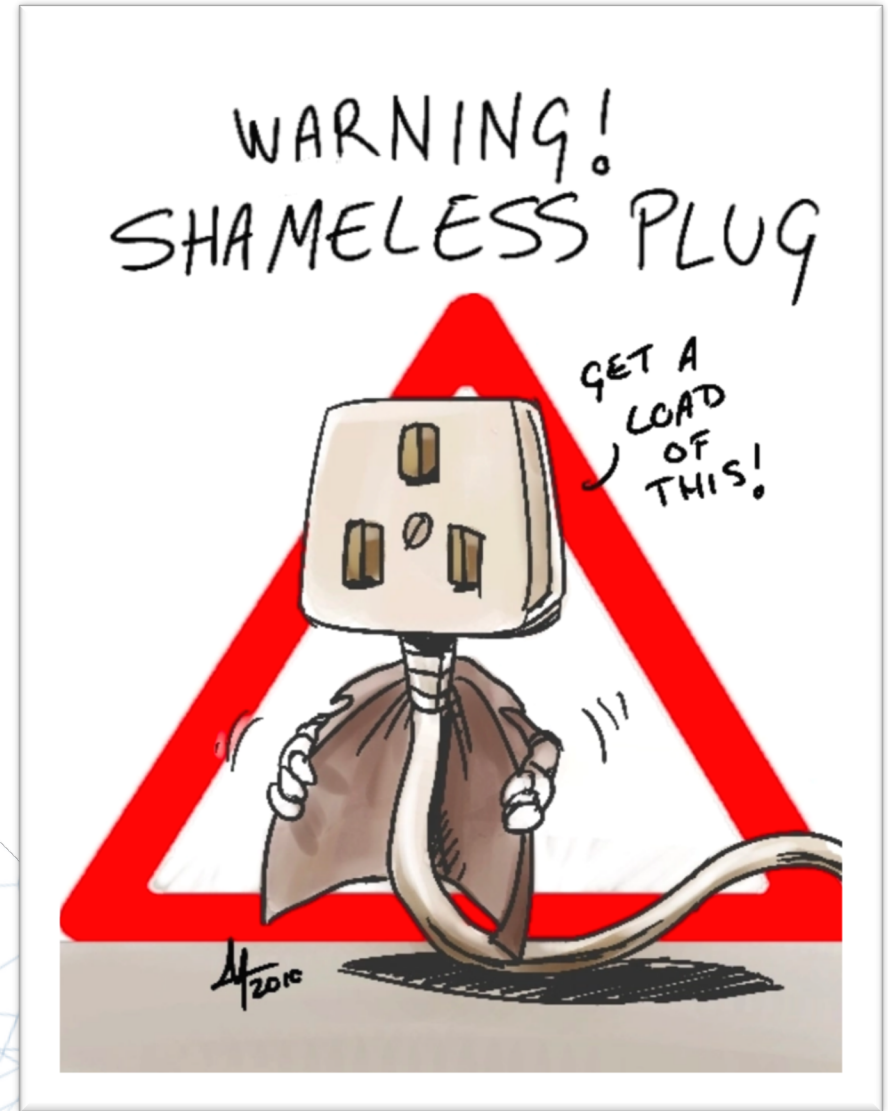
Example #1 – Let's Talk About the Vuln

```
pragma solidity ^0.5.3;

contract Ownable {
    address private _owner;

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

    constructor () internal {
        _owner = msg.sender;
        emit OwnershipTransferred(address(0), _owner);
    }

    function owner() public view returns (address) {
        return _owner;
    }

    modifier onlyOwner() {
        require(isOwner());
        _;
    }

    function isOwner() public view returns (bool) {
        return msg.sender == _owner;
    }
}
```

Example #2 – Find the Vulnerability!

```
155     /**
156      * @dev Direct token transfer. Submitted by the wallet owner
157      * @param to Recipient address
158      * @param value Transfer amount
159      * @param tokenContract Address of the token contract used for the transfer
160      */
161     function pay(address to, uint value, address tokenContract) onlyOwner public returns (bool) {
162         IERC20 token = IERC20(tokenContract);
163         require(token.transfer(to, value));
164         return true;
165     }
```

Hungry to find more vulnerabilities?



Checkout our free, interactive blockchain CTF!

A series of vulnerable smart contracts and DApps with real-life use cases, ranging from decentralized trust funds and open source lottery systems to automated royalty agreements.

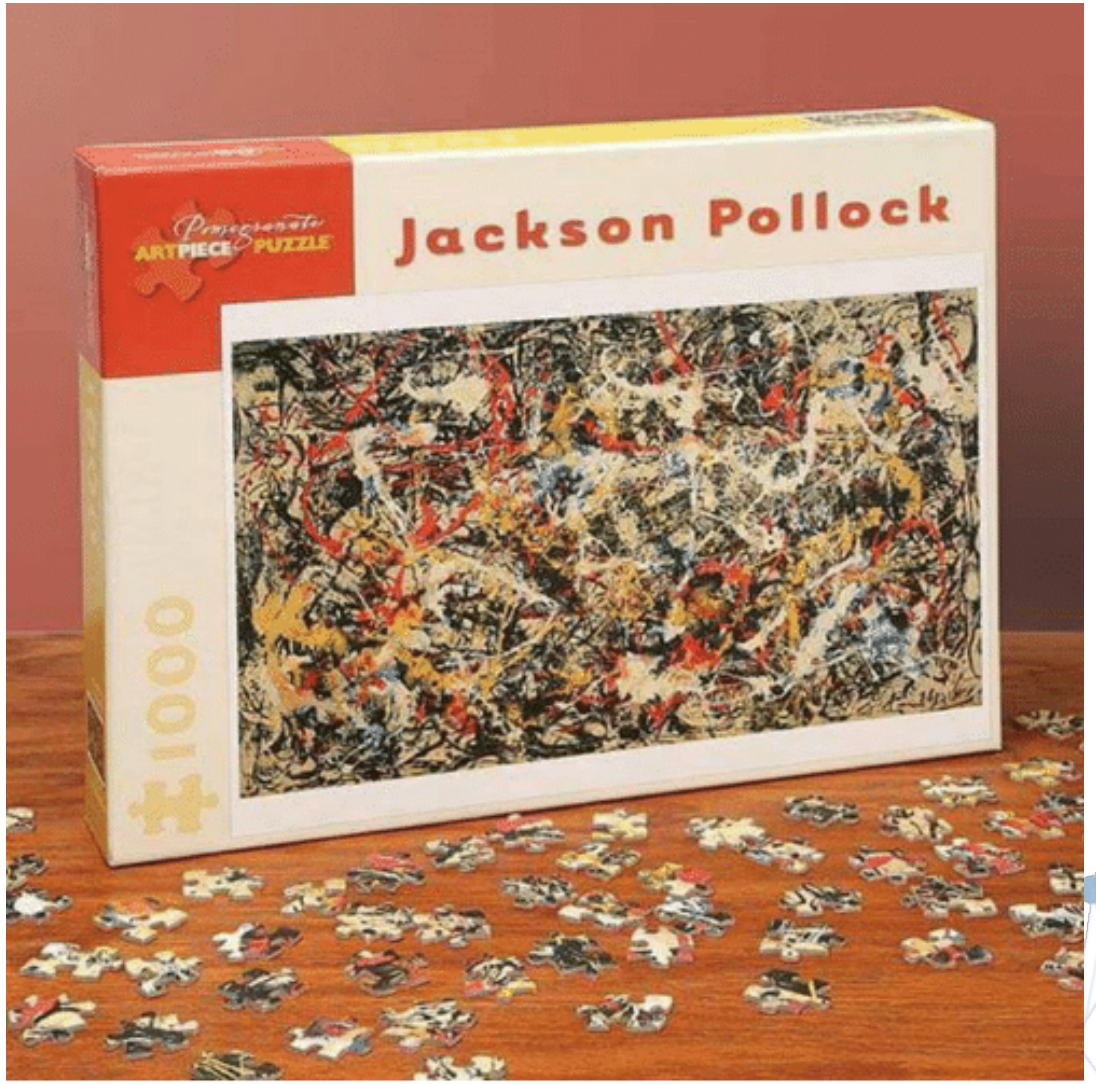
SECURITY INNOVATION

Reporting

Components of a finding

- Intro
  - Severity, Target, Line number
- Description
  - Overview, Steps to reproduce
- Remediation
  - Clear, concise, actionable
- Response

- The full, public Metacash smart contract audit report is available here



Example Intro

Non-Compliant Tokens are Not Supported

- Severity
  - Medium
- Target
  - SmartWallet
- Line Number
  - 163

Example Description

Non-Compliant Tokens are Not Supported

The ERC20 token standard describes a transfer function as returning a Boolean True value if the transfer is successful. Unfortunately, many popular tokens do not correctly implement this standard and instead do not return any value. Two such tokens include OMG and BNB. More information on the topic is available in the following

https://medium.com/coinmonks/missing-return-value-bug-at-least-130-tokens-affected-d67bf08521ca

The Metacash wallet assumes that all tokens that are transferred strictly follow this standard and wraps all token transfers in a require(...) to validate the returned value is true. This means that if a token is sent to this contract that does not correctly implement IERC20 and does not return a boolean, all calls to transfer that token will fail.Note that it is still possible to retrieve these tokens by constructing a call for execCall(...), though this is non-intuitive for an average user.

Example Remediation

Non-Compliant Tokens are Not Supported

Provide Support for Non-Compliant Tokens - Provide a separate payNoncompliant(...) function that does not expect a boolean return in the transfer so that typical wallet owners have a way of moving non-compliant tokens out of their wallet.

Example Response

Non-Compliant Tokens are Not Supported

Instead of adding a special function for non-compliant tokens, we have removed the requirement for external ERC20 token transfers to return true. This should be compatible with tokens that do return true as well as those that do not.

Congratulations

You have made it through **UMACAR**!

- **U**nderstand use case
- **M**anual review
- **A**utomated tools
- **C**ommunication
- **A**nalysis
- **R**eporting

Questions?

Thank You!

www.securityinnovation.com

SECURITY INNOVATION

Sources

- https://medium.com/quillhash/life-cycle-of-smart-contract-development-8929fa073b7f
- https://www.google.com/search?client=firefox-b-1-d&biw=1680&bih=954&tbm=isch&sa=1&ei=CXxAXfzyHIn5-wSn5JvwAg&q=confused+don%27t+get+paid+meme&oq=confused+don%27t+get+paid+meme&gs_l=img.3...39992.42142..42295...0.0..0.69.723.15.....0....1..gws-wiz-img.....0i7i30j0i8i7i30.KYkddeDOAZU&ved=0ahUKEwi8m56lk93jAhWJ_J4KHSfyBi4Q4dUDCAY&uact=5#imgrc=7yMHLnWiJ0C77M:
- https://www.google.com/search?client=firefox-b-1-d&biw=1680&bih=954&tbm=isch&sa=1&ei=VEZDXdn8Ocrz-gTsvJ0I&q=congrats+meme&oq=congrats+meme&gs_l=img.3..0l10.33642.37241..37363...0.0..0.91.1198.24.....0....1..gws-wiz-img.....0..0i67.7fc6IbcZMl0&ved=0ahUKEwjZ9ra_vOLjAhXKuZ4KHWxeBwEQ4dUDCAY&uact=5#imgrc=fVdtsp0AzJxAeM:
- https://www.google.com/search?q=terrified+meme&client=firefox-b-1-d&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjlt4vVv-LjAhUWup4KHcEiD74Q_AUIESgB&biw=1680&bih=954#imgrc=vcmDyfRqjsBFqM:
- https://www.google.com/search?client=firefox-b-1-d&biw=1680&bih=954&tbm=isch&sa=1&ei=Ik5DXa6zI8rQ-gTTx63QAg&q=dilbert+communication&oq=dilbert+communication&gs_l=img.3..0l3j0i5i30j0i8i30j0i24.40660.42978..43145...0.0..0.74.1018.21.....0....1..gws-wiz-img.....0i67j0i10i24.QdfJoai0Skw&ved=0ahUKEwiuzPz3w-LjAhVKqJ4KHdNjCyoQ4dUDCAY&uact=5#imgrc=91ZMe2AoSEGSRM:
- https://giphy.com/gifs/YQitE4YNQNahy
- https://www.bing.com/images/search?view=detailV2&ccid=SS%2fodwsr&id=EC2B6B8CC9D160F8B03124B163AEB0632D4C5329&thid=OIP.SS_odwsrhYc0FS0exojMRwHaJ4&mediaurl=http%3a%2f%2fwww.quickmeme.com%2fimg%2f1c%2f1cce146b8bfc7f12cd05c5cb14e435507fe9248ed21bbfc357a3ff4fc98bc101.jpg&exph=640&expw=480&q=schedule+meme&simid=607998593078922990&selectedIndex=33&ajaxhist=0
- https://www.bing.com/images/search?view=detailV2&ccid=zIAy36%2fQ&id=FF5BF63E8E3B0BD7863562590A4742BACF122FB2&thid=OIP.zIAy36_QNuko3vP_bTW-7wHaJy&mediaurl=https%3a%2f%2floveisdope.files.wordpress.com%2f2015%2f05%2fshamelessplug.jpg%3fw%3d645&exph=700&expw=530&q=shameless+plug&simid=607994422682128331&selectedIndex=2&ajaxhist=0
- https://hackernoon.com/practical-smart-contract-security-analysis-and-exploitation-part-1-6c2f2320b0c

Secure Development Lifecycle

SDL

- Created by Microsoft in 2002-ish
  - Embed security if it was going to be taken seriously by marketplace
- SDL is a process
- the SDL is a process that standardizes security best practices across a range of products and/or applications.
- If you look at the many SDLs that exist across industries, you'll find that most include the same basic security phases and activities. They may have different names for the pieces, but everyone follows roughly the same process.

Why SDL?

Security front and center + Standardized approach

- 1. The lack of a standard approach to securing products causes problems.
- 2. The second problem is that developers tend to repeat the same security mistakes
- 3. Finally, without a security standard customers have no assurance that a given product is secure

Phases

The standard approach to SDL includes requirements, design, implementation, test, and release/response

Life of a Smart Contract