



Securing the Unknown

An efficient methodology for securing smart contracts throughout their lifecycle



About Me

Ben Stewart

- Security engineer & researcher
- Automation aficionado
- Basketball player



Agenda

- Vulnerability landscape
- Methodology 101
- **Understand** use case
- **Manual** review
- **Automated** tools
- **Communication**
- **Analysis**
- **Reporting**

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- Outdated Compiler Version [SWC-102]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.”

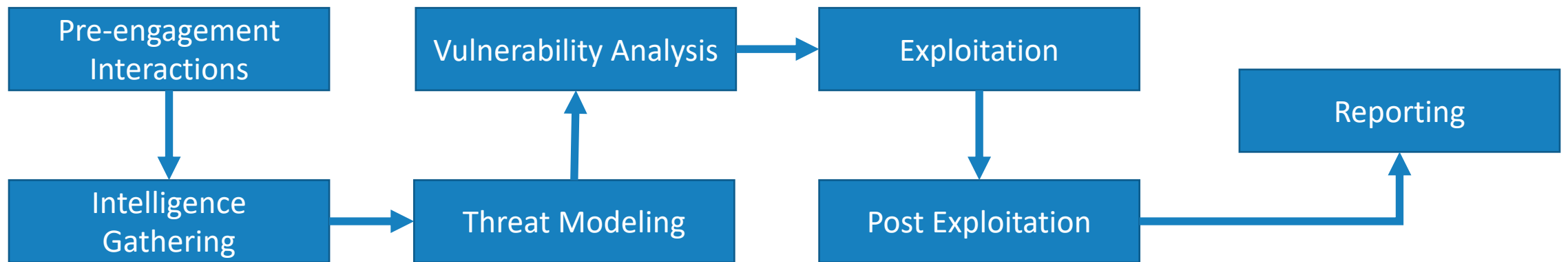
-Wikipedia

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

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

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?

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``

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

Truffle Teams & Other Tools

- Truffle Teams
 - Continuous integration
 - Automated deployments
 - State management
- Aggregated [list](#) of security tools
 - Visualization
 - Additional static and dynamic analyzers
 - Test cases and coverage tools
 - Solidity linters

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 – Spot the Vulnerability!

```
1  pragma solidity ^0.5.3;
2
3  contract Ownable {
4      address private _owner;
5
6      event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
7
8      constructor () internal {
9          _owner = msg.sender;
10         emit OwnershipTransferred(address(0), _owner);
11     }
12
13     function owner() public view returns (address) {
14         return _owner;
15     }
16
17     modifier onlyOwner() {
18         require(isOwner());
19         _;
20     }
21
22     function isOwner() public view returns (bool) {
23         return msg.sender == _owner;
24     }
25 }
```

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 ICOs and automated royalty agreements. Each of these applications contain a vulnerability commonly found in smart contracts. Participants can practice exploiting these bugs to steal fake crypto-currencies and win points on our leaderboard.

Reporting

Components of a finding

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



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#imgcr=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..0i67.7fc6lbcZMI0&ved=0ahUKEwjZ9ra_vOLjAhXKuZ4KHWxeBwEQ4dUDCAY&uact=5#imgcr=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#imgcr=vcmDyfrQjsBFqM:
- https://www.google.com/search?client=firefox-b-1-d&biw=1680&bih=954&tbm=isch&sa=1&ei=lk5DXa6zl8rQ-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=0ahUKEwizPz3w-LjAhVKqJ4KHdNjCyoQ4dUDCAY&uact=5#imgcr=91ZMe2AoSEGSRM:

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

