QuillAudits

Smart Contracts Audit

What is Smart Contract ?

A **Smart contract** is just a simple code that runs on top of a blockchain containing set of rules under which the stakeholders agree to interact with each other.

Smart Contracts are self executing and self verifying.

Sample Smart Contract Code

```
pragma solidity 0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
```

Issues with Smart Contracts (Ethereum)

QuillAudits

Ownership (Parity attack (**30 mil Usd**) and Oyester pearl attack)

Reentrancy (The famous Dao attack **50 mil Usd**)

Underflow and Overflow

Short Address attack (Golem Attack)

External Calls — Every external contract call is a risk(make sure all the internal work(state condition) is complete before calling external functions)

Storage injection vulnerability in NEO Smart Contracts (which allows anyone to change the token's total supply limit by transferring their own tokens to an unspecified address.)

Ownership Attack

QuillAudits

150,000 ETH (~30M USD)

This causes all public functions from the library to be callable by anyone, including initWallet, which can change the contract's owners. Unfortunately, initWallet has no checks to prevent an attacker from calling it after the contract was initialized. The attacker exploited this and simply changed the contract's m_owners state variable to a list containing only their address, and requiring just one confirmation to execute any transaction:

Reentrancy

QuillAudits

```
function withdraw(uint amount)
{
    if (credit[msg.sender] >= amount) {
     msg.sender.call.value(amount)();
     credit[msg.sender] -= amount;
}
```

```
function() public {
    dao.withdraw(dao.assignedCredit(this));
  }
```

Underflow and Overflow

QuillAudits

First things first, let's make sure we understand what an uint256 is. A uint256 is an unsigned integer of 256 bits (unsigned, as in only positive integers). The Ethereum Virtual Machine was designed to use 256 bits as its word size, or the number of bits processed by a computer's CPU in one go. Because EVM is limited to 256 bits in size, the assigned number range is 0 to 4,294,967,295 (2^{256}). If we go over this range, the figure is reset to the bottom of the range ($2^{256} + 1 = 0$). If we go under this range, the figure is reset to the top end of the range ($0–1= 2^{256}$).

External Calls

External Calls — Every external contract call is a risk, make sure all the internal work (state condition) is complete before calling external functions or contracts .

All the transfer calls should be made after completing internal work first.

Issues with Smart Contracts (EOS)

QuillAudits

Buffer overflows

Proper memory management

dangling pointers

Permissions to action mapping

Optimising RAM usage

Buffer overflows

QuillAudits

Maliciously Exploitable

Hotspots : Array size, Numerical flow

Proper memory management

QuillAudits

- Check overflows of RAM values

- prevent loss of data

Dangling pointers

QuillAudits

- Pointers pointing to stray locations

- Locations can be maliciously targeted

Permissions to action mapping

QuillAudits

- Cause of many hacks of EOS Dapps
- Check for actions to permission mapping
- Hotspots: Transfer receipts and actions

Optimising RAM usage

QuillAudits

- Critical and costly resource

- Know when to buy and release RAM

- Know when to allocate and deallocate RAM storage

Why is Security Audit necessary

QuillAudits

- To catch the bugs that humans missed.
- To reduce known attacks on your Smart Contract.
- To ensure all paths of functions are functioning as intended to be.

Importance of Security Audit

QuillAudits

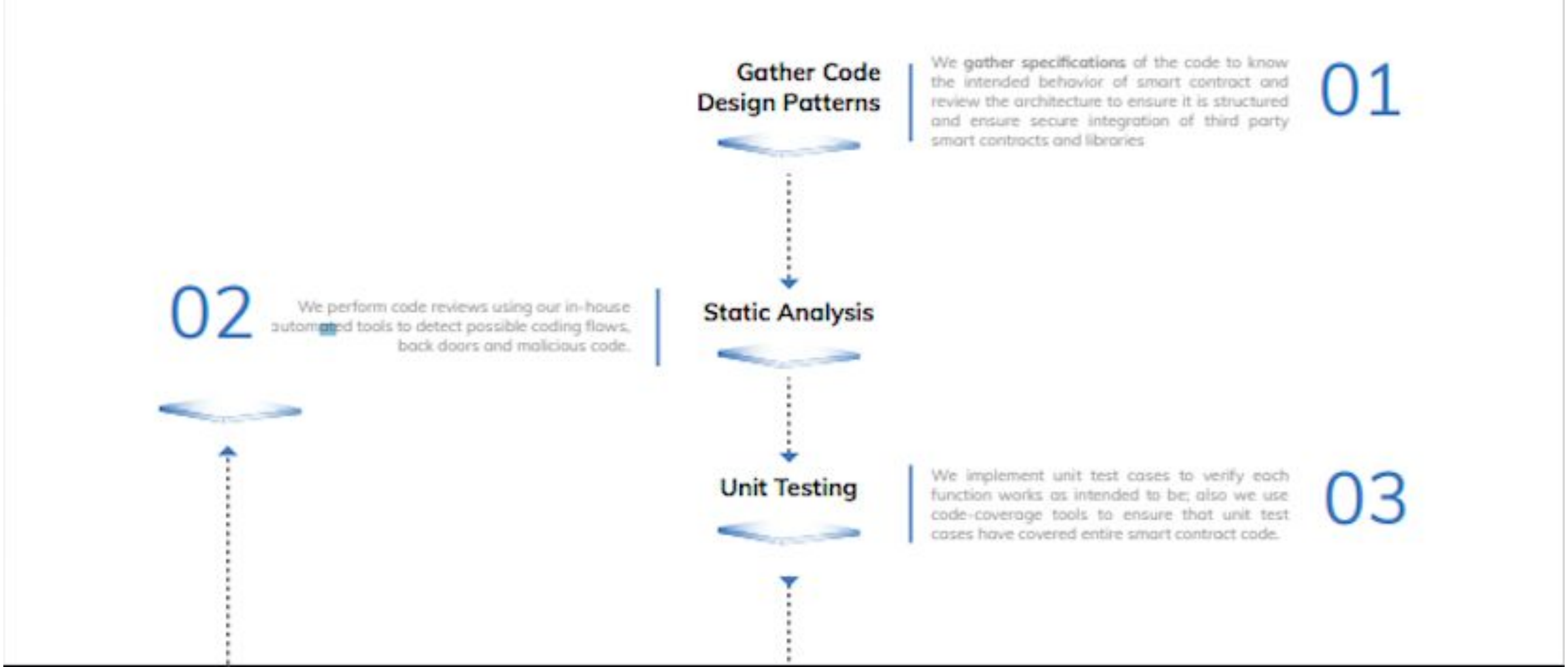
Marketing

(Secure contract attract more investors and users)

Security

Validate Use-Case

An Ideal Audit Process



QuillAudits

Types of Tools

Security Tools (Mythril, Oyente, Slither)

Testing Tools (Truffle)

Monitoring Tools (Quill-SDK, Neufund~ to monitor transaction)

Mythril Classic

Mythril Classic is an open-source security analysis tool for Ethereum smart contracts. It uses **concolic analysis, taint analysis and control flow checking** to detect a variety of security vulnerabilities.

Features of Mythril

QuillAudits

Concolic analysis : Symbolic Analysis, treating variable as a symbol to generate new concrete inputs(test suits), aim of maximizing code coverage.

Taint analysis : The taint analysis is a popular method which consists to check which variables can be modified by the user input.

Control flow checking : To check order in which statements, instruction and function calls taking place.

Oyente

QuillAudits

Oyente is able to detect some of the latest security flaws of Ethereum, including TheDAO bug, which

```
root@d6092bf156c9:/oyente/oyente# python oyente.py -s greeter.sol
WARNING:root:You are using evm version 1.8.2. The supported version is 1.7.3
WARNING:root:You are using solc version 0.4.21, The latest supported version is
 0.4.19
INFO:root:contract greeter.sol:greeter:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                       99.5%
INFO:symExec:     Integer Underflow:                       False
INFO:symExec:     Integer Overflow:                        False
INFO:symExec:     Parity Multisig Bug 2:                   False
INFO:symExec:     Callstack Depth Attack Vulnerability:    False
INFO:symExec:     Transaction-Ordering Dependence (TOD):   False
INFO:symExec:     Timestamp Dependency:                    False
INFO:symExec:     Re-Entrancy Vulnerability:               False
INFO:symExec:   ====== Analysis Completed ======
```

Oyente Features

QuillAudits

Attacks checked by Oyente :

Timestamp dependence attack

Reentrancy bug

Concurrency bug

Overflow/Underflow

Slither Tool

QuillAudits

Slither is a Solidity static analysis framework written in Python 3. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

Slither Features

QuillAudits

Attacks check by Slither

- ➢ Suicidal (self destructing)
- ➢ uninitialized-state
- ➢ External-function call
- ➢ reentrancy

Output by Slither Tool

QuillAudits

```
+------------------+------------+------------+----------------+--------+------------------------+------------
---+
|     Function     | Visibility | Modifiers |      Read      | Write |      Internal Calls      |
External Calls |
+------------------+------------+------------+----------------+--------+------------------------+------------
---+
| i_am_a_backdoor |   public   |     []     | ['msg.sender'] |   []   | ['selfdestruct(address)'] |
[]       |
+------------------+------------+------------+----------------+--------+------------------------+-----------
```

Sūrya Tools

It provides a number of visual outputs and information about the contracts' structure. Also supports querying the function call graph in multiple ways to aid in the manual inspection of contracts.

Solidity-coverage

QuillAudits

This tool identify how efficient Unit testing were, or how much code is covered in unit testing, This step is important because it is required to know whether unit tests touch each line of code or not to

File | file:///home/abhi_21094/Desktop/Home-projects/maester-contract/coverage/contracts/index.html

all files contracts/

88.49% Statements 223/252 **57.73%** Branches 112/194 **88.52%** Functions 54/61 **88.1%** Lines 185/210

File		Statements		Branches		Functions		Lines	
Crowdsale.sol		90%	108/120	63.64%	70/110	88.89%	24/27	87.91%	80/91
MAP.sol		88.98%	113/127	51.25%	41/80	90.32%	28/31	89.47%	102/114
Ownable.sol		40%	2/5	25%	1/4	66.67%	2/3	60%	3/5

Unit Test Cases by QuillAudits

QuillAudits

```
✓ Should correctly initialize constructor values of Token Contract (255ms)
✓ Should Deploy Crowdsale only (160ms)
✓ Should Deploy Vesting Contract only (91ms)
✓ Should set Vesting Contract Address to Crowdsale Contract (43ms)
✓ Should Activate Sale contract (76ms)
✓ Should check balance of Crowdsale after, crowdsale activate from token contract
✓ Should Authorize KYC for account 2 (95ms)
✓ Should Freeze Account (118ms)
✓ Should Start CrowdSale  (255ms)
✓ Should be able to pause and unPause Crowdsale contract (266ms)
✓ Should add Strategic Investor before they participate in Private Sale  (124ms)
✓ Should set Price of Token per dollar for Strategic Investor (94ms)
✓ Should Authorize KYC for account 6 (141ms)
✓ Should be able to buy Tokens according to private sale round one as a Strategic Investor (4763ms)
✓ Should be able to buy Tokens  according to private sale (4904ms)
✓ Should Authorize KYC for account 3 (108ms)
✓ Should be able end private sale round one and buy Tokens according to private sale round two (2218ms)
✓ Should Authorize KYC for account 4 (112ms)
✓ Should be able end private sale round Two and buy Tokens according to pre sale  (2523ms)
✓ Should Authorize KYC for account 5 (118ms)
✓ Should be able end pre sale and buy Tokens according to public Sale  (2697ms)
✓ Should be able to end public sale and finalize sale (199ms)
✓ Should vest Token of Team  (59ms)
✓ Should vest Token of Advisor  (48ms)
✓ Should be able to finalize Sale after sale is Over (185ms)
✓ Should send Bounty Tokens   (127ms)
✓ Should be able to transfer Tokens got in pre Sale after Sale is Over  (162ms)
✓ Should be able to transfer Tokens got public Sale after Sale is Over
```

REDEFINING BLOCKCHAIN SECURITY STANDARD

Reach us at:

https://audits.quillhash.com

hello@quillhash.com

Follow Us On:

QuillAudits