Hacking smart contracts

@konstanthacker

Konstantinos Karagiannis
CTO, Security Consulting
ethereum is not bitcoin

“The key component is this idea of a Turing-complete blockchain”

--Vitalik Buterin
smart contracts

• Business logic programs
• Semi autonomous
• Move value, enforce agreements
• Creativity the limit
No DAO funds at risk following the Ethereum smart contract ‘recursive call’ bug discovery

Our team is blessed to have Dr. Christian Reitwießner, Father of Solidity, as its Advisor. During the early development of the DAO Framework 1.1 and thanks to his guidance we were made aware of a generic vulnerability common to all Ethereum smart contracts. We promptly circumvented this so-called “recursive call vulnerability” or “race to empty” from the DAO Framework 1.1 as can be seen on line 580:
caveats

• No zero days
• No customer code
• Yes, a methodology
• No, I doubt smart contracts will get that smart
solidity

- Language of choice
- High level, compiles to bytecode
- Similarities to JavaScript and C
- Supports:
  - libraries
  - inheritance
  - user-defined types
  - assembly inline
dev tools

- .sol files > bytecode > blockchain
- Auditing .sol easier with highlighting
- Atom my fave, with plugins
  - language-ethereum
  - etheratom
- Remix—browser based
contract MyContract {
    uint balance;

    function MyContract() {
        Mint(10000000);
    }

    function Mint(uint amount) internal {
        balance = amount;
    }

    function withdraw() {
        msg.sender.send(balance);
    }

    function GetBalance() constant returns(uint) {
        return balance;
    }
}
oyente

• Symbolic execution tool
• Works with EVM byte code or .sol files
• Detects 4* vulns
• Low false positive rate
basic methodology

• Interview devs
• Load .sol file, preferably with highlighting
• Try compiling
• Dissect code flow—optional solgraph
• Run oyente (cross fingers)
• Manually verify 3/4 vuln yay/nays
• Proceed to manually check for following vulns...
contract ReEntrancy {

    mapping (address => uint) private expendableTokens;

    function stealTokens() public {
        uint amountToLose = expendableTokens[msg.sender];
        if (!(msg.sender.call.value(amountToLose)())) { throw; }
        expendableTokens[msg.sender] = 0;
    }
}
leave off the first “re” for savings

```solidity
contract Entrancy {
    mapping (address => uint) private expendableTokens;

    function stealTokens() public {
        uint amountToLose = expendableTokens[msg.sender];
        expendableTokens[msg.sender] = 0;
        if (!(msg.sender.call.value(amountToLose)())) { throw; }
    }
}
```
unchecked send in king of the ether

```solidity
uint compensation = valuePaid - wizardCommission;

if (currentMonarch.etherAddress != wizardAddress) {
    currentMonarch.etherAddress.send(compensation);
} else {
    // When the throne is vacant, the fee accumulates for the wizard.
}
```
unchecked send

```java
if (kingOfLosingDone && !compensationSent) {
    monarch.send(500);
    compensationSent = True;
}
```

```java
if (kingOfLosingDone && !compensationSent) {
    if (monarch.send(500))
        compensationSent = True;
    else throw;
}
```
gas limits
withdraw don’t send

```solidity
contract SendContract {
    address public richest;
    uint public mostSent;

    function SendContract() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() payable returns (bool) {
        if (msg.value > mostSent) {
            richest.transfer(msg.value);
            richest = msg.sender;
            mostSent = msg.value;
            return true;
        } else {
            return false;
        }
    }
}
```
 withdrawn not sent

```
contract WithdrawalContract {
    address public richest;
    uint public mostSent;

    mapping (address => uint) pendingWithdrawals;

    function WithdrawalContract() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() payable returns (bool) {
        if (msg.value > mostSent) {
            pendingWithdrawals[richest] += msg.value;
            richest = msg.sender;
            mostSent = msg.value;
            return true;
        }
        else {
            return false;
        }
    }

    function withdraw() {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        msg.sender.transfer(amount);
    }
}
```
encryption
transaction-ordering dependence

```solidity
contract Puzzle{
    address public owner;
    bool public locked;
    uint public reward;
    bytes32 public diff;
    bytes public solution;

    function Puzzle(){
        owner = msg.sender;
        reward = msg.value;
        locked = false;
        diff = bytes32(11111);
    }

    function(){
        if (msg.sender == owner){
            if (locked) throw;
            owner.send(reward);
            reward = msg.value;
        }
        else
            if (msg.data.length > 0){
                if (locked) throw;
                if (sha256(msg.data) < diff){
                    msg.sender.send(reward);
                    solution = msg.data;
                    locked = true;
                }
            }
    }
```
call-stack depth limit

Announcement of imminent hard fork for EIP150 gas cost changes:

During the last couple of weeks, the Ethereum network has been the target of a sustained attack. The attacker(s) have been very crafty in locating vulnerabilities in the client implementations as...
variable or function ambiguity

```java
Player[] public persons;

uint public payoutCursor_Id_ = 0;
uint public balance = 0;

address public owner;

uint public payoutCursor_Id=0;
...

while (balance > persons[payoutCursor_Id_].deposit / 100 * 115) {
    uint MultipliedPayout = persons[payoutCursor_Id_].deposit / 100 * 115;
    persons[payoutCursor_Id].etherAddress.send(MultipliedPayout);
    balance -= MultipliedPayout;
    payoutCursor_Id_++;
}
```
odds and ends

• Input validation – require(condition)
• Timestamp dependence
• Business logic flaws
• Separating public/private data
get involved
dox me ... or just keep in touch

@konstanthacker

konstantinos.karagiannis@bt.com