Smart Contract Security

Assessing Solidity smart contracts

OWASP

The Open Web Application Security Project

About Me

OWASP
The Open Web Application Security Project

- Evangelos Deirmentzoglou
- Security Consultant
- Smart contract audits

- Nmap/Ncrack contributor
- Certs: OSCE, OSCP, OSWP

Positive
com

OWASP
The Open Web Application Security Project

- Blockchain Basics
- Front Running
- Reentrancy
- External Calls
- Integer Overflow
- tx.origin
- Gas limit

OWASP
The Open Web Application Security Project

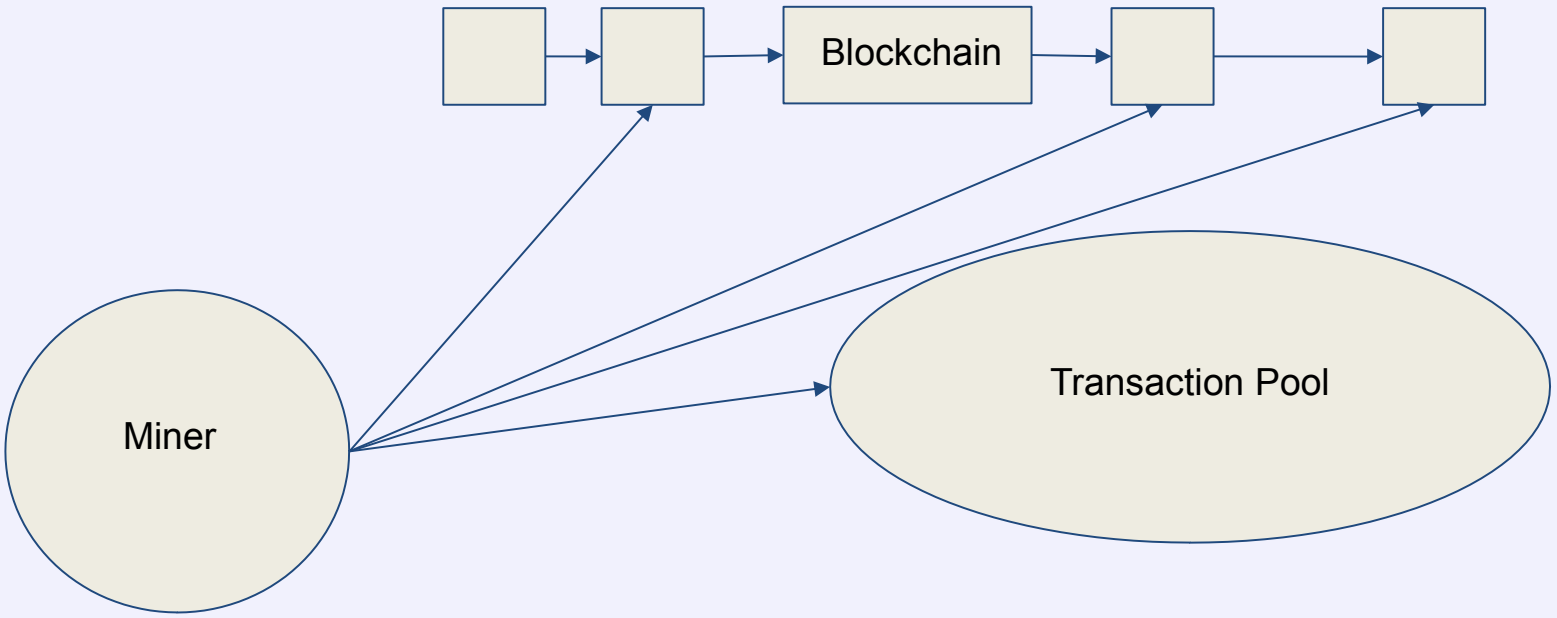
Ethereum Gas

- Introduced to reduce abuse
- transaction cost (EVM operation)
- gas price (in eth)

Miner receives estimated gas spent* gas price

Remainder of gas cost is returned to the sender

Blockchain Transactions



Frontrunning

OWASP
The Open Web Application Security Project

- aka Timing attacks
- Affecting: First In First Out Contracts
- Miner frontrunning
- User frontrunning

Frontrunning

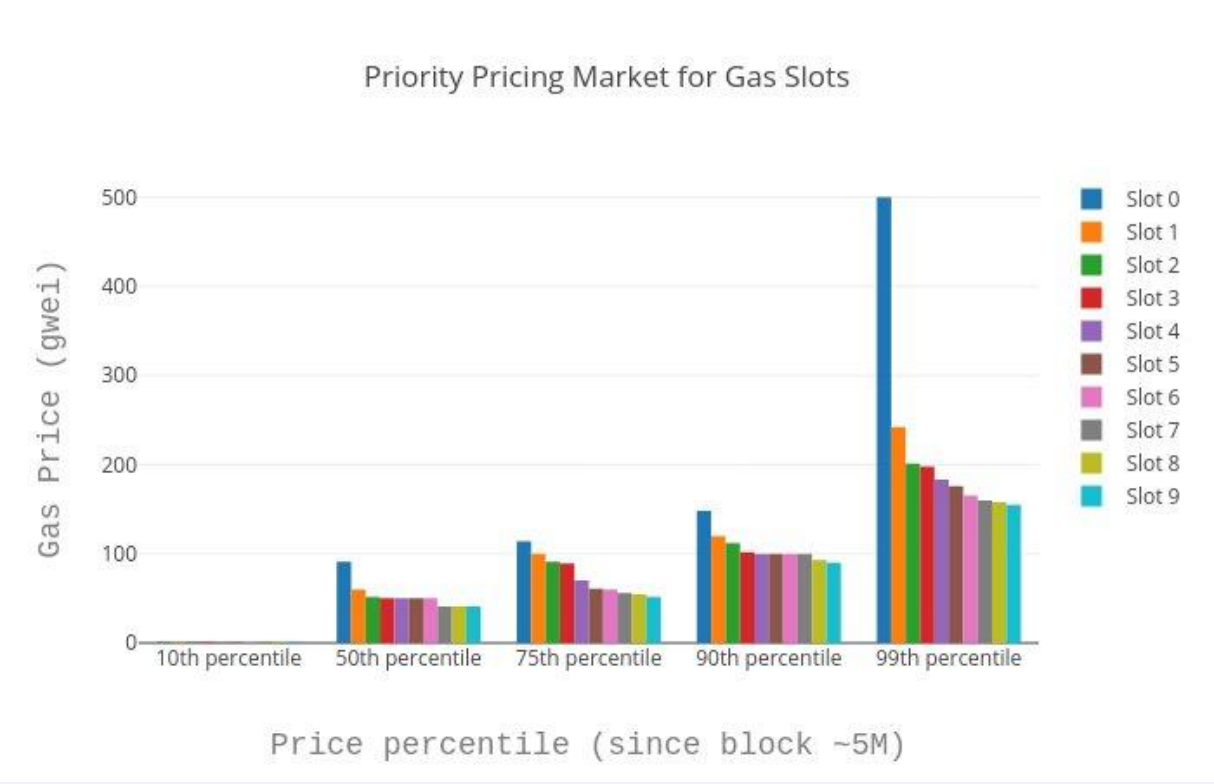
OWASP
The Open Web Application Security Project

- Examples: DEX, Bancor, random ICOs

```
1   function freeEth(){
2       msg.sender.transfer(1000 ether);
3   }
```

Reference: https://hackernoon.com/front-running-bancor-in-150-lines-of-python-with-ethereum-api-d5e2bfd0d798

Frontrunning



Reference: https://twitter.com/phildaian/status/1032679149960355840

Frontrunning - Mitigation

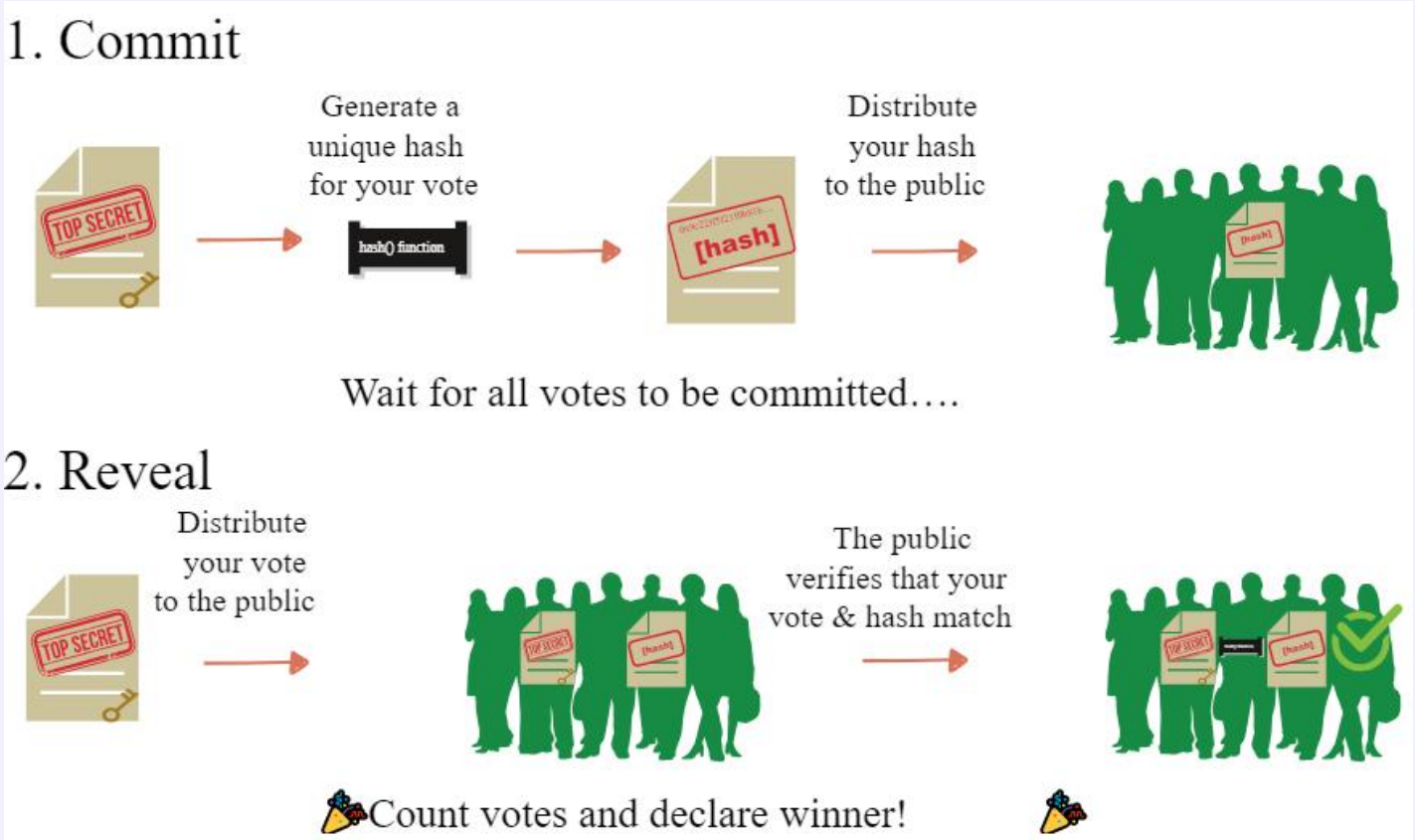
OWASP
The Open Web Application Security Project

- Set upper gas limit
- Set a fixed gas limit
- tx.gasprice
- gasleft()

```
1  function () payable {
2      require(tx.gasprice <= max_gas_price);
3      require(gasleft() <= max_gas_limit);
4      //...
5  }
```

OWASP
The Open Web Application Security Project

Commit/Reveal approach



Reference: https://karl.tech/learning-solidity-part-2-voting/

OWASP
The Open Web Application Security Project

Payable functions

```
1  function deposit() public payable {
2      balances[msg.sender] += msg.value;
3  }
4
5  function () payable {
6      balances[msg.sender] += msg.value;
7  }
```

Reentrancy

OWASP
The Open Web Application Security Project

150$ mil stolen

DAO & numerous other contracts

Reentrancy - Vulnerable Contract

OWASP
The Open Web Application Security Project

```
1  function deposit() public payable {
2      balances[msg.sender] += msg.value;
3  }
4
5  function withdraw(uint256 _withdraw) public {
6      require(balances[msg.sender] >=  withdraw);
7      require(msg.sender.call.value(_withdraw)());
8      balances[msg.sender] -= _withdraw;
9  }
```

Reentrancy - Attack Contract

OWASP
The Open Web Application Security Project

```
function deposit() public payable {
    balances[msg.sender] += msg.value;
}

function withdraw(uint256 _withdraw) public {
    require(balances[msg.sender] >= _withdraw);
    require(msg.sender.call.value(_withdraw)());
    balances[msg.sender] -= _withdraw;
}
```

```
function attack(address target) payable {
  c = VictimContract(target);
  c.deposit.value(0.1 ether)();
  c.withdraw(0.1 ether);
}
function() payable {
  c.withdraw(0.1 ether);
}
```

Reentrancy - Mitigation

OWASP
The Open Web Application Security Project

- Always use transfer()
- Use sensitive operations before calls to other contracts
- ReentrancyGuard by OpenZeppelin

Reference: https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/ReentrancyGuard.sol

Reentrancy - Bridges

OWASP
The Open Web Application Security Project

- Reentrancy is not really obsolete
- “A bridge to connect all bridges”
- Connects ERC20 tokens

Reentrancy - Bridges

OWASP
The Open Web Application Security Project

Not reentrancy per se

```
function gift(ERC20 _token, uint256 _amount, address _receiver){
    require(_token.transfer(_receiver, _amount);
    nativetoken.mint(msg.sender, reward);
}
```

```
function transfer(address _adress, uint _amount) {
    vuln_contract.gift(_attack_contract, _amount, _attack_contract);
}
```

More like call to the unknown

External Contracts

OWASP
The Open Web Application Security Project

- Trust that called function will succeed
- Trust that you know what the called function do

```
1  function destroy(){
2      require(!destroyed);
3      msg.sender.send(1000 ether);
4      destroyed = true;
5  }
6
```

External Contract - Mitigation

OWASP
The Open Web Application Security Project

- Ensure external contract returns expected value
- Perform sensitive operations before calling external contract
- Never assume external contract functionality

Integer Overflow

OWASP
The Open Web Application Security Project

- Proof of Weak Hands Coin - 866 Eth
- BatchOverflow

Age	From		To	Value	Token
2 hrs 8 mins ago	0x3f2dd0cb25bbf89...	OUT	0x521c526d5b50de...	57,896,044,618,658,100,000,000,000...	Erc20
2 hrs 8 mins ago	0x3f2dd0cb25bbf89...	OUT	0x4473c6396eba3d...	57,896,044,618,658,100,000,000,000...	Erc20
2 hrs 54 mins ago	0x3f2dd0cb25bbf89...	OUT	0x66f471fd1c471bb...	57,896,044,618,658,100,000,000,000...	Erc20
2 hrs 54 mins ago	0x3f2dd0cb25bbf89...	OUT	0x4473c6396eba3d...	57,896,044,618,658,100,000,000,000...	Erc20

Reference: https://medium.com/@peckshield/alert-new-batchoverflow-bug-in-multiple-erc20-smart-contracts-cve-2018-10299-511067db6536

Integer Overflow

OWASP
The Open Web Application Security Project

- Integer only
- Conversions (Unsigned -> Signed)

```
1  function transfer(int _amount) {
2      require(_amount >= 0);
3      balances[msg.sender] -= _amount;
4      msg.sender.transfer(balances[msg.sender]);
5  }
```

More overflow cases:

https://github.com/ethereum/solidity/issues/796#issuecomment-253578925

Integer Overflow - Mitigation

OWASP
The Open Web Application Security Project

- Perform operations within boundaries
- SafeMath library

```
51      function add(uint256 _a, uint256 _b) internal pure returns (uint256) {
52        uint256 c = _a + _b;
53        require(c >= _a);
54
55        return c;
56      }
```

Reference: https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/math/SafeMath.sol

tx.origin

OWASP
The Open Web Application Security Project

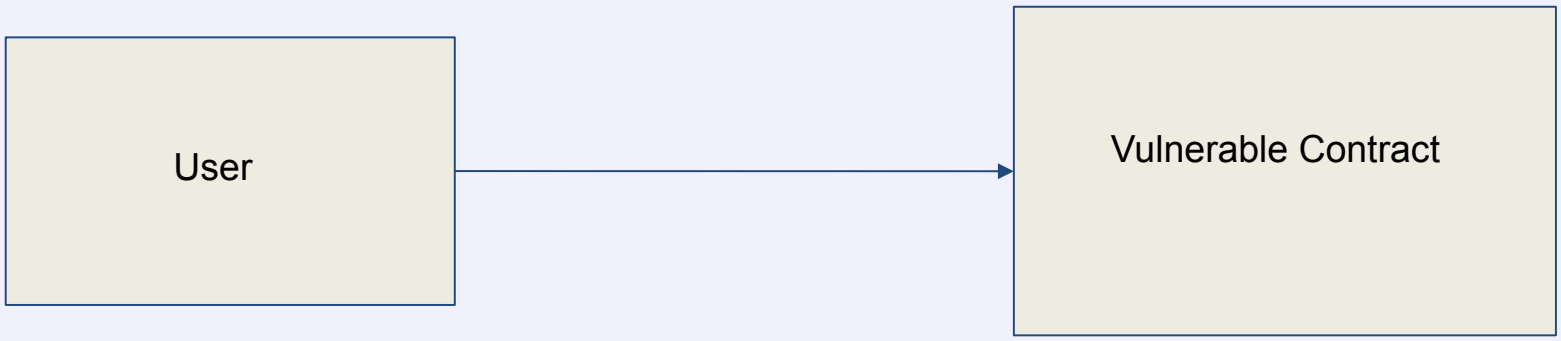
- Blockchain CSRF?
- tx.origin vs msg.sender

```
1 function withdraw(uint _amount) {
2     require(tx.origin == owner);
3     msg.sender.transfer(1000 ether);
4 }
```

tx.origin

OWASP
The Open Web Application Security Project

call()	For Vulnerable Contract
tx.origin	User
msg.sender	User



tx.origin

call()	For Malicious Contract	For Vulnerable Contract
tx.origin	User	User
msg.sender	User	Malicious Contract

tx.origin - Mitigation

OWASP
The Open Web Application Security Project

- call vs delegatecall
- Caution when using tx.origin

Gas limit attacks

OWASP
The Open Web Application Security Project

- Looping over unknown arrays
- Leads to denial of service

```
1  function distribute() public {
2      for(uint i = 0; i < investors.length; i++) {
3          // Important administrative tasks
4          // ...
5      }
6  }
```

Gas limit attacks - Mitigation

OWASP
The Open Web Application Security Project

- Avoid looping over arrays of unknown length
- Set an upper limit for the array length
- Control the loop by checking gasleft()

```
1  function distribute() public {
2      while(i < investors.length && gasleft() > 20000) {
3          // Important administrative tasks
4          // ...
5      }
6  }
```

Things we didn't cover

OWASP
The Open Web Application Security Project

- Randomness (is hard)
- Visibility (Function, secret)
- Unexpected ether (contract-suicide)
- Delegatecall (3rd party libraries)
- Storage

OWASP
The Open Web Application Security Project

Questions?