D4: Unchecked Return Values For Low Level Calls

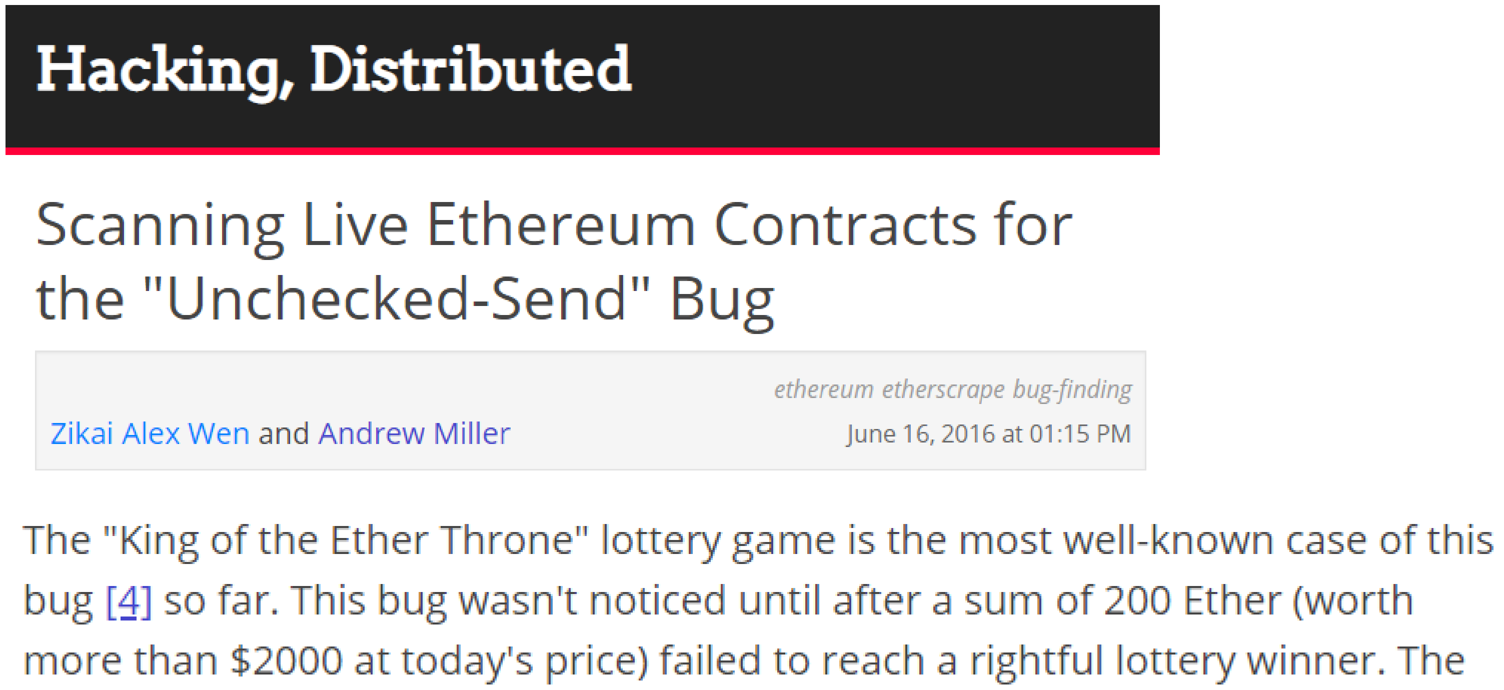
Portland State
Computer Science

#4:Unchecked Return Values For Low Level Calls

- Also known as **silent failing sends**, **unchecked-send**
  - Inconsistent Exception Handling in EVM
  - Errors in calls typically lead to transaction failure and a total reversion of the execution
  - Low level functions call(), callcode(), delegatecall() and send() with different error handling than regular Solidity functions
    - Errors do not propagate (e.g. bubble up via exception)
      - Return "false" upon failure
      - If return value not checked, code can be incorrect
    - In Solidity, such calls should be avoided whenever possible
    - Note: Beyond 0.4.13, usage is now flagged upon compilation

Example #1

- "King of the Ether Throne"
  - https://github.com/kieranelby/KingOfTheEtherThrone/blob/v0.4.0/contracts/KingOfTheEtherThrone.sol
  - http://www.kingoftheether.com/postmortem.html



Hacking, Distributed

Scanning Live Ethereum Contracts for the "Unchecked-Send" Bug

ethereum etherscrape bug-finding

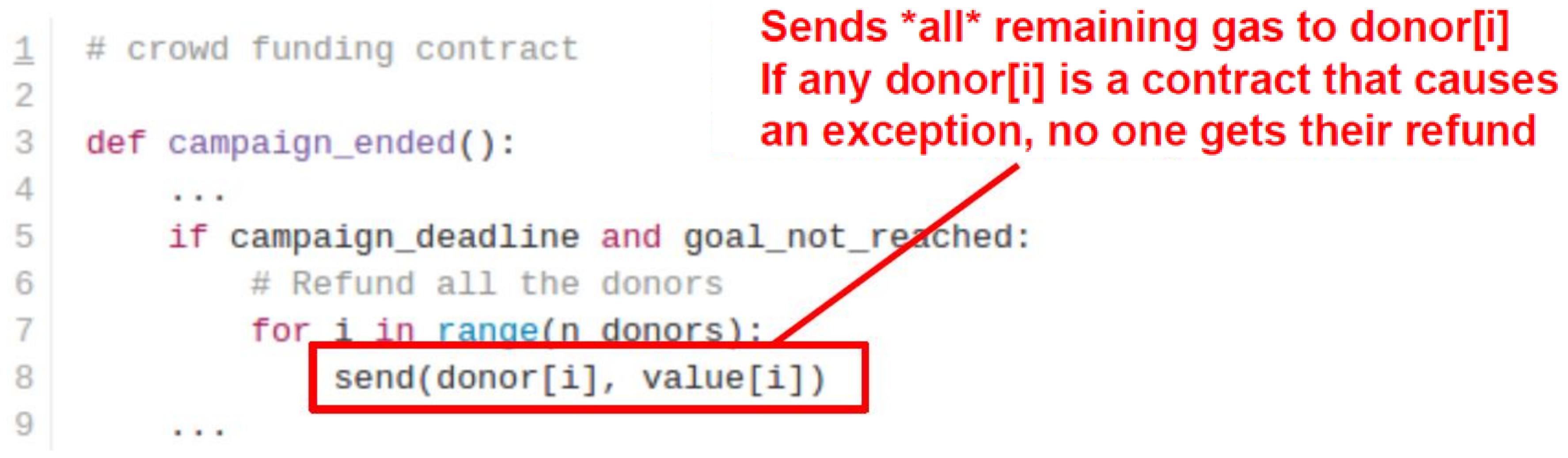
Zikai Alex Wen and Andrew Miller

June 16, 2016 at 01:15 PM

The "King of the Ether Throne" lottery game is the most well-known case of this bug [4] so far. This bug wasn't noticed until after a sum of 200 Ether (worth more than $2000 at today's price) failed to reach a rightful lottery winner. The

Code vulnerability example #1

- If fundraising goal not met, return money
  - Donors can be either wallets or smart contracts
  - Sending Ether to a smart contract invokes fallback function on that contract
    - Use of `send()` forwards all gas to donor's fallback function
    - If gas runs out in fallback, `send()` returns false and the rest of refund loop fails (no more gas)
    - Return not checked for success
    - Funds may be locked up for good with one rogue donor

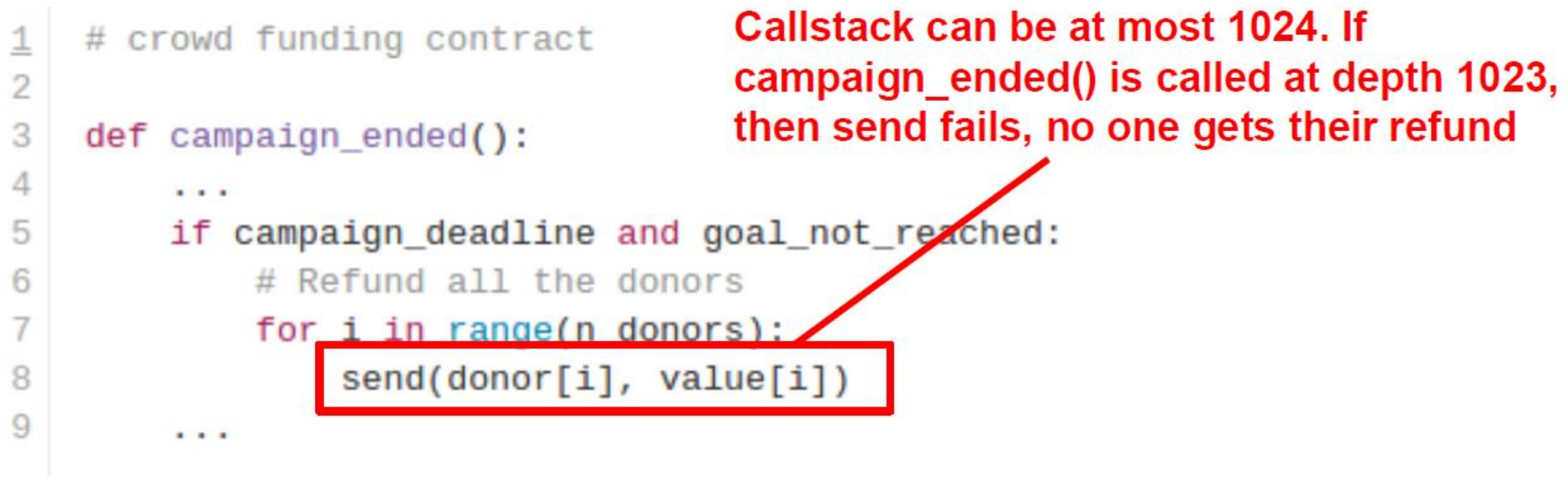


```
# crowd funding contract

def campaign_ended():
    ...
    if campaign_deadline and goal_not_reached:
        # Refund all the donors
        for i in range(n_donors):
            send(donor[i], value[i])
    ...
```

Sends *all* remaining gas to donor[i]
If any donor[i] is a contract that causes an exception, no one gets their refund

- Same contract can fail via obscure VM rules
  - Maximum stack depth is 1024
  - Low-level calls return true or false only
  - Unless checked, can lead to contract bricking



```
# crowd funding contract

def campaign_ended():
    ...
    if campaign_deadline and goal_not_reached:
        # Refund all the donors
        for i in range(n_donors):
            send(donor[i], value[i])
    ...
```

Callstack can be at most 1024. If campaign_ended() is called at depth 1023, then send fails, no one gets their refund

Remediation

- Use `address.transfer()`
  - Throws exception on failure
  - Forwards only 2,300 gas making it safe against re-entrancy (more later)
- Avoid low level call `address.send()`
  - Returns `false` on failure
  - Call forwards only 2,300 gas making it safe against re-entrancy
  - Only use in rare cases that you want to handle failure condition within your contract (versus reverting call)
- Avoid low level call `address.call.value().gas()()`
  - Returns `false` on failure
  - Forwards all available gas by default, not safe against re-entrancy
  - Only use when you need to control how much gas to forward when sending ether or to call a function of another contract

- Check for call failure if using low-level call
  - e.g. Recipient runs out of gas processing transfer
  - EVM call stack full (past 1024) on executing contract
  - http://hackingdistributed.com/2016/06/16/scanning-live-ethereum-contracts-for-bugs

```
if (gameHasEnded && !( prizePaidOut ) ) {
  winner.send(1000); // send a prize to the winner
  prizePaidOut = True;
}
```

```
if (gameHasEnded && !( prizePaidOut ) ) {
  if (winner.send(1000))
    prizePaidOut = True;
  else
    revert("Failure to send.  Undo call.");
}
```

- Have recipient withdraw money (and pay gas to do so)

```
if (gameHasEnded && !( prizePaidOut ) ) {
  accounts[winner] += 1000
  prizePaidOut = True;
}
...
function withdraw(amount) {
  require(accounts[msg.sender] >= amount);
  if (msg.sender.send(amount))
    accounts[msg.sender] -= amount;
  else
    revert("Failure to send.  Undo call.");
}
```