Finding the Greedy, Prodigal, and Suicidal Contracts at Scale

Paper By Nikolić, Kolluri, Sergey, Saxena, and Hobor
Presented by Thomas Quig

Outline

- Brief overview of Ethereum
- Smart contracts applied within the Ethereum Blockchain.
- What makes a contract Greedy? Prodigal? Suicidal?
- Trace Vulnerabilities
    - Formal Properties & Definitions
    - Examples
- MAIAN - Symbolic Analysis of Ethereum Bytecode
- Evaluation of Contracts
    - Results
- Discussion Questions

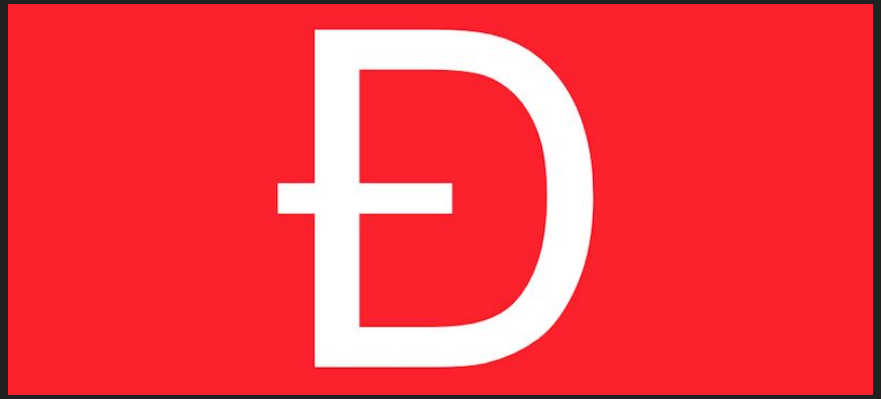
Ethereum

- Open source blockchain with Smart Contract functionality.
- Components
  - Smart Contracts
    - Computer programs designed for **automatic control** of pseudo legal events/transactions.
    - Vending Machines : Money + Request → Thing
  - Ether
    - The actual currency associated with the Ethereum Blockchain
  - Accounts
    - Two types of accounts, Normal Accounts and Smart Contracts.
  - Gas
    - The “in-game currency” used to submit/run Smart Contracts in the blockchain
- Current Value
  - As of 1613901999, 1 ETH = 1,940.85 (That's Epoch Time 21/2/21)

Smart Contracts (In the context of the Ethereum Blockchain)

- Made up of **bytecode**
- Run on the EVM
- Any modifying functions costs Ether (gas) to run
- Contracts are run by Miners, output is agreed upon
- **Immutable**

The DAO Bug - Summary



- DAO
  - Distributed Autonomous Organization
  - Governance Network
- Split Function
  - Split contracts between majority and minority
  - Had recursive call
- Re-Entry Bug
  - Took funds from the Split before checking the Split's Balance.
  - Could do this repeatedly
- Recursive Execution
  - Criminal managed to call DAO within itself

The DAO Bug - Effects

- 3.6 Million Ether Stolen
  - At the time was $150 Million
  - Currently Valued at 7 BILLION ETH ($55,000,000 ETC)
- Hard Fork *(Can the Solidity language and/or EVM be modified in a scalable way?)*
  - Community Decided to **split into ETH and ETC**
- Security Considered More

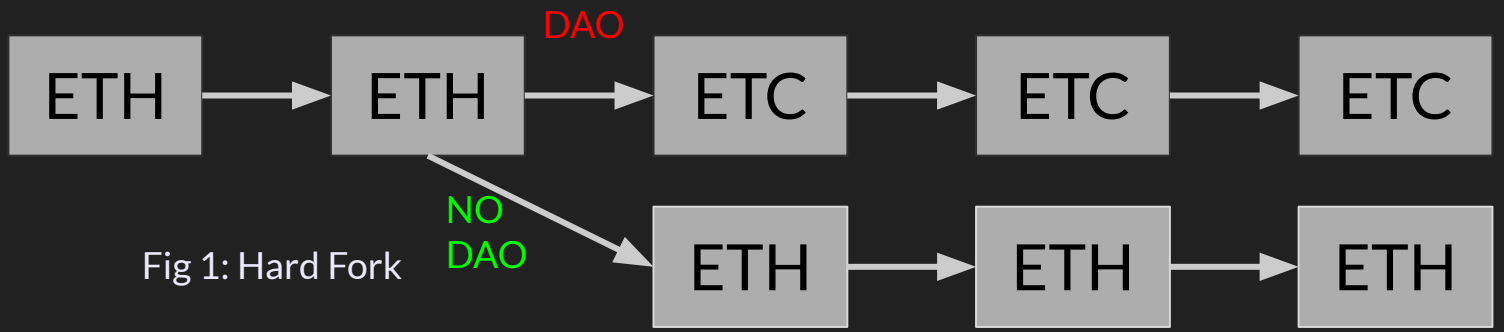
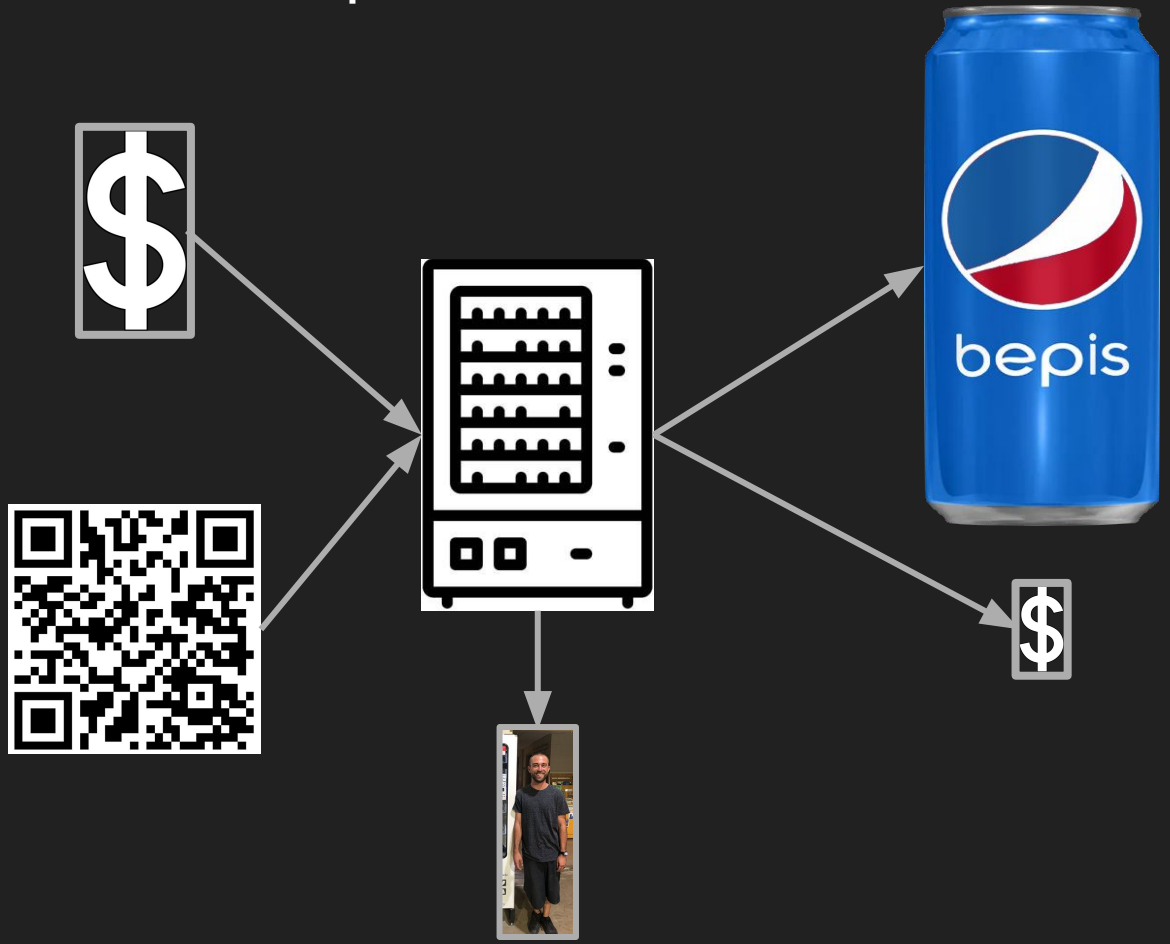


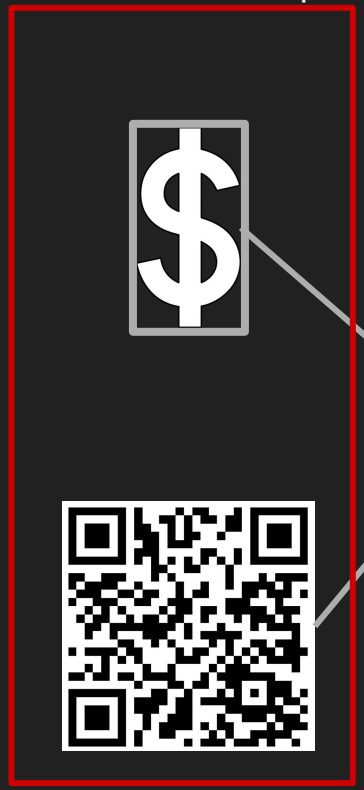
Fig 1: Hard Fork

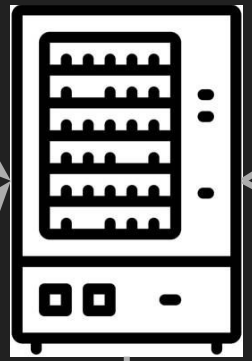
What does DAO have to do with this?

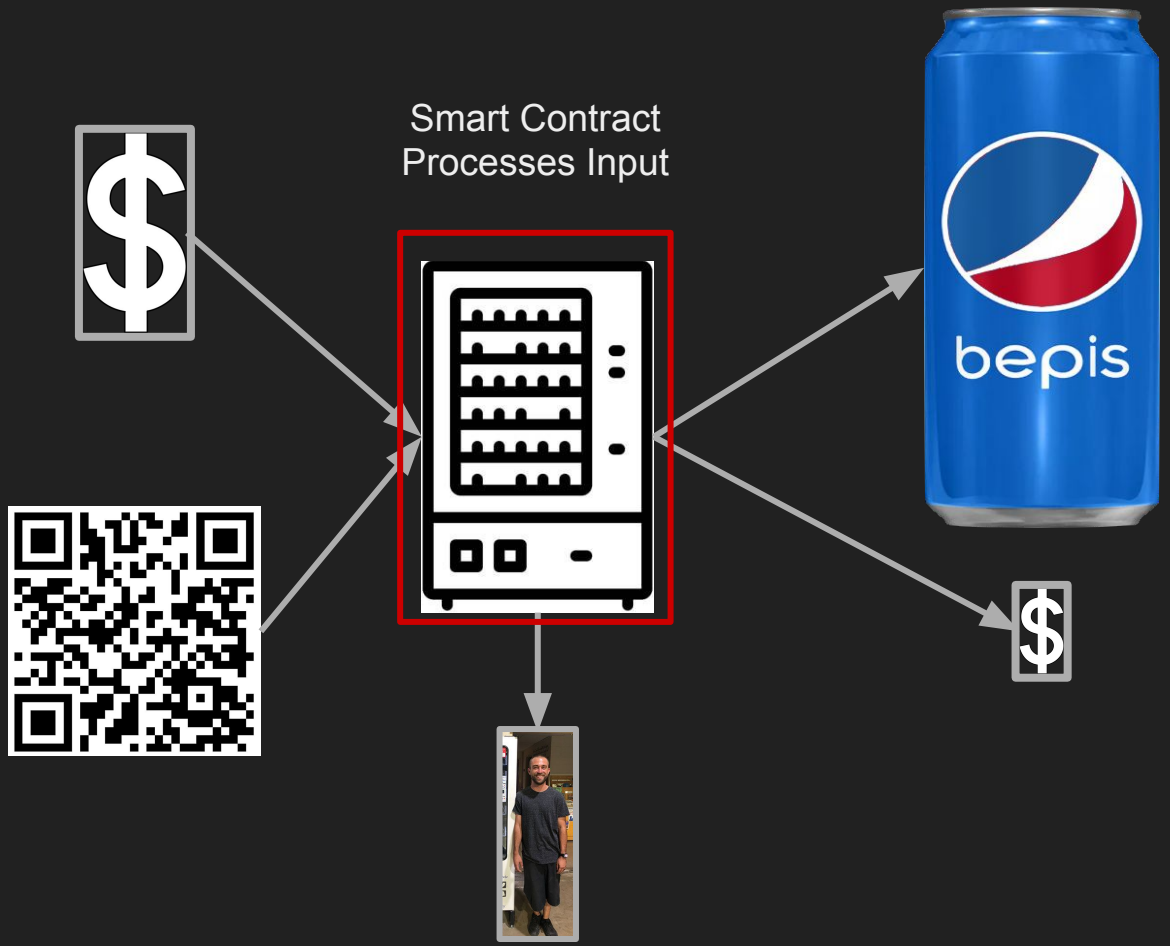
Vending Machine Example



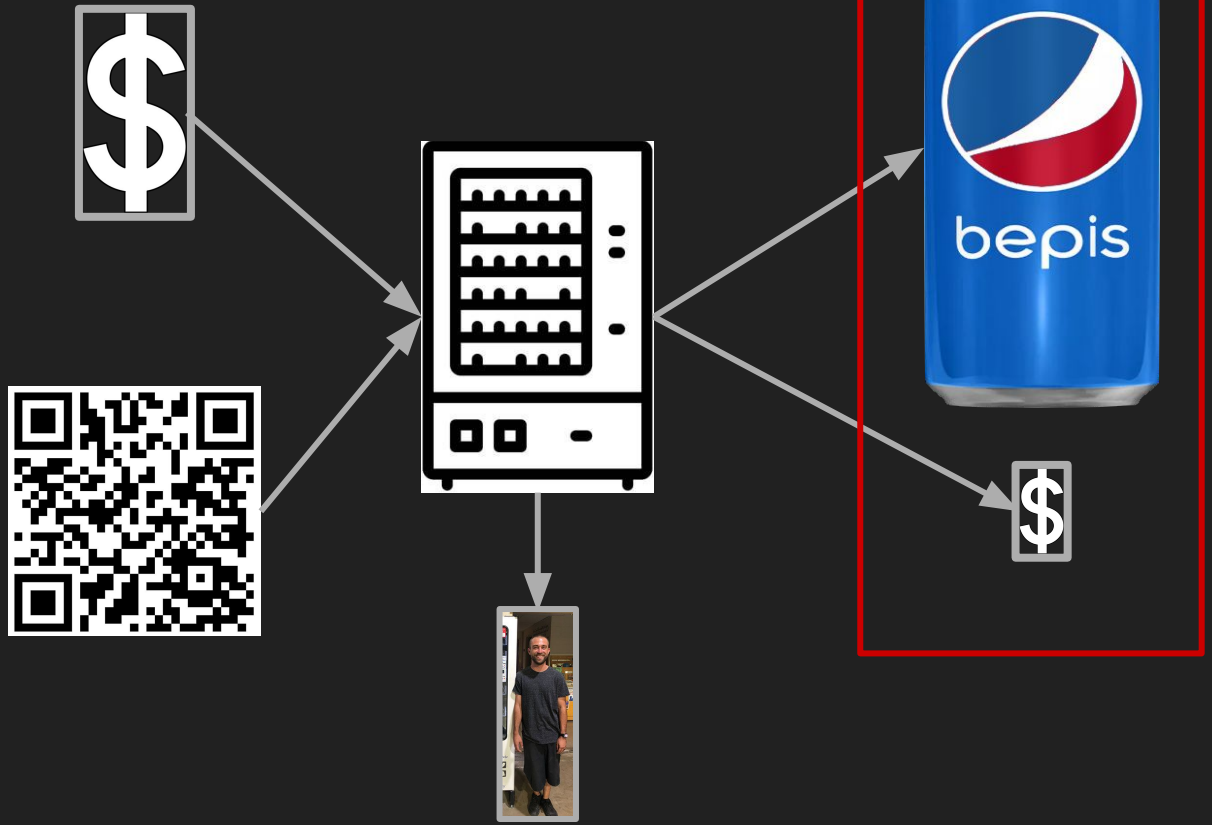
Ether + Some Input

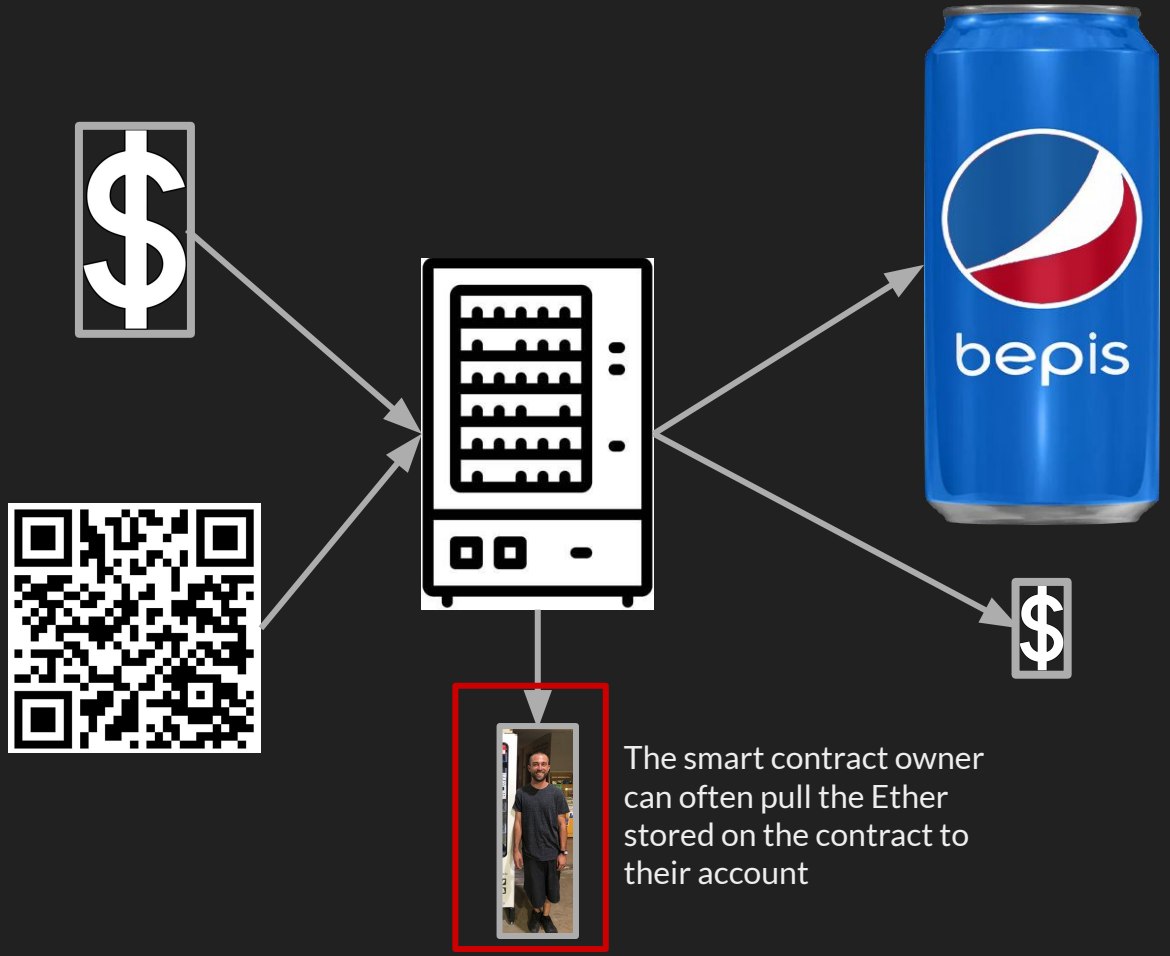





Smart Contract
Processes Input
bepis

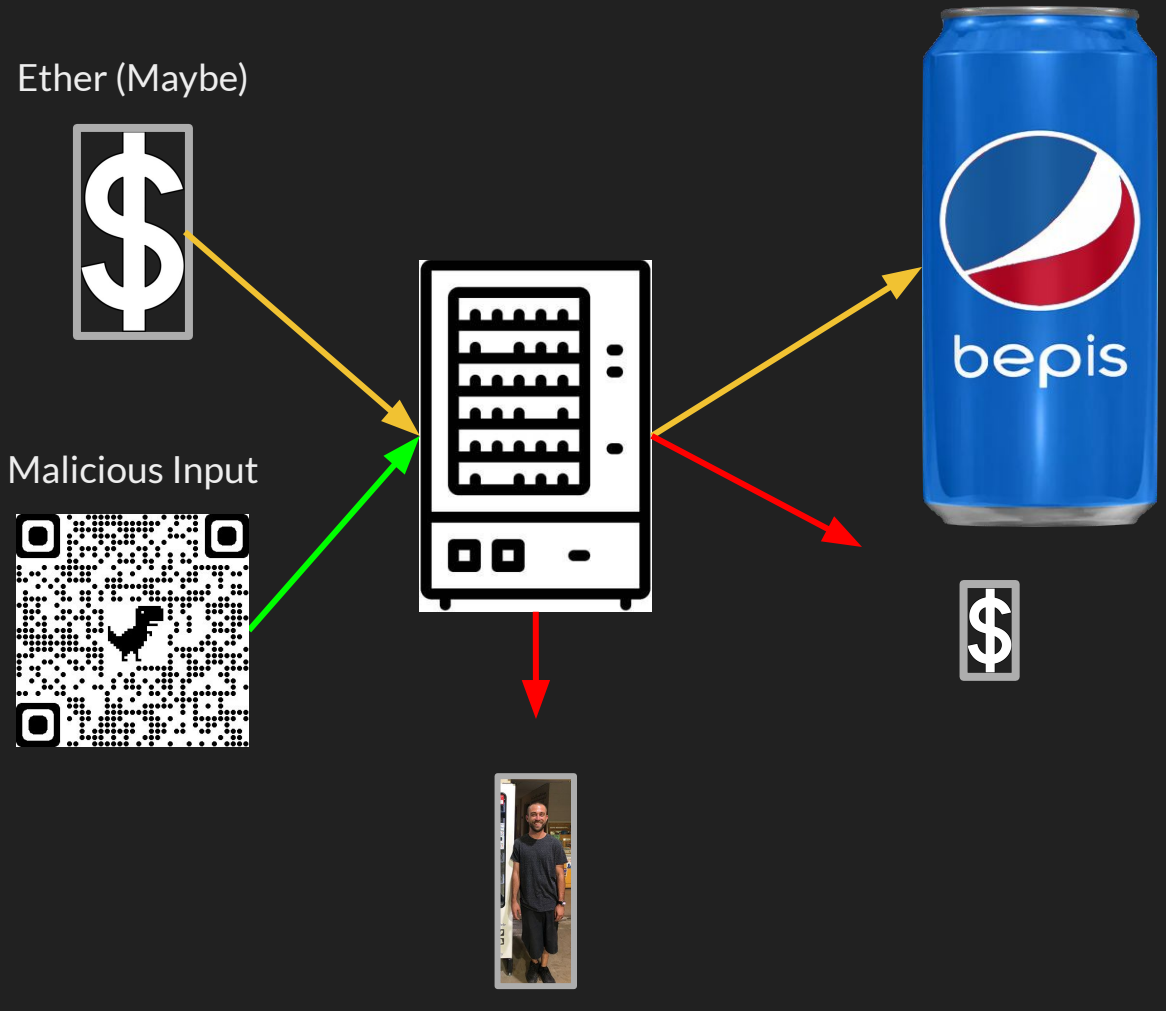
Returns an output
and/or some Ether

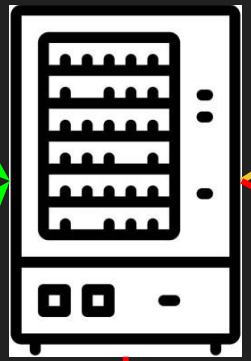




The smart contract owner can often pull the Ether stored on the contract to their account

Greedy Contracts

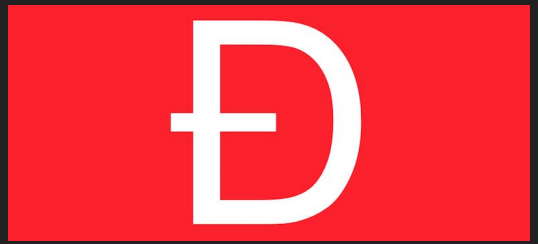

Ether (Maybe)
Malicious Input
bepis



No funds can be transferred.

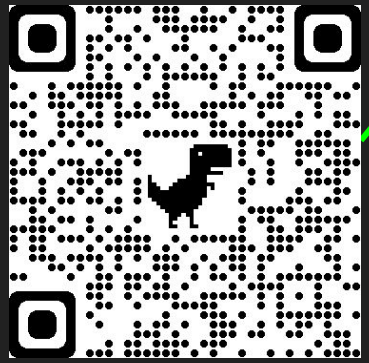
The owner can no longer pull ether out of the contract.

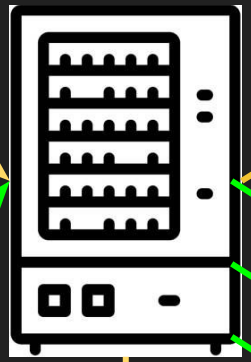
Prodigal Contracts

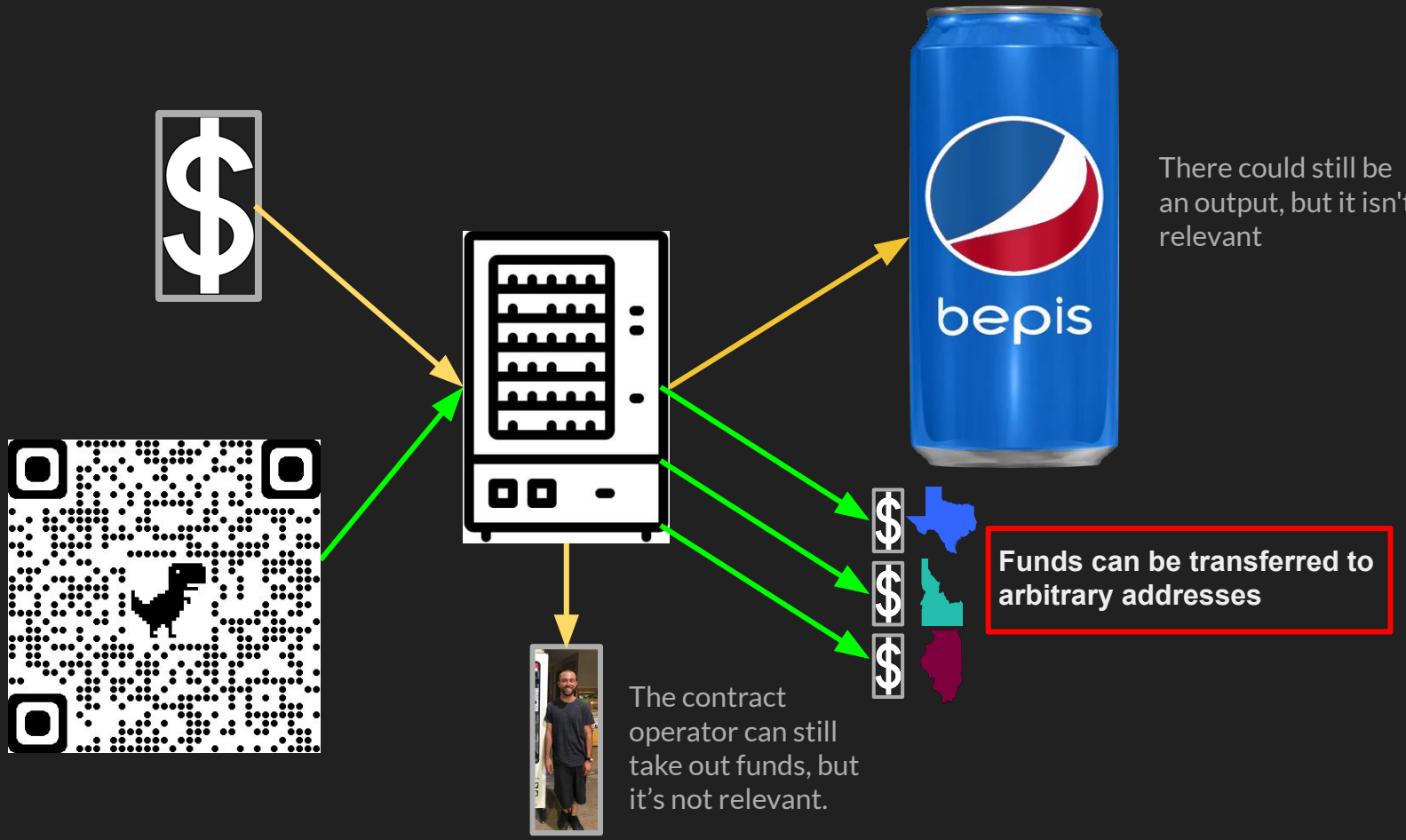


Ether (Maybe)

Malicious Input

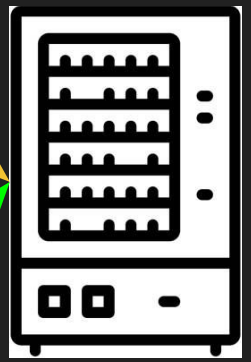




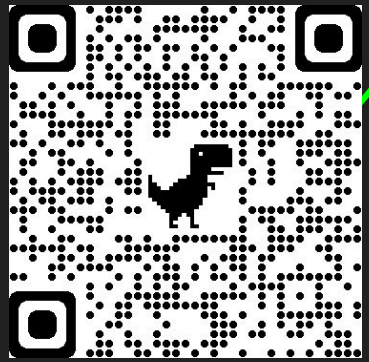

bepis
There could still be an output, but it isn't relevant
Funds can be transferred to arbitrary addresses
The contract operator can still take out funds, but it's not relevant.

Suicidal Contracts

Ether (Maybe)



Malicious Input



Caveats of vending machine example

- Vending machine is a black box
- Bugs can appear in both
  - Can't fix bugs in ethereum
- Can’t complain to support in Ethereum
- Attacks on vending machines not scalable
- Physical Limitations

Parity Bug

Questions?

Discussion Question

Are there any examples of attackers using an analysis of this kind to attack vulnerable contracts en masse?

Trace Vulnerabilities

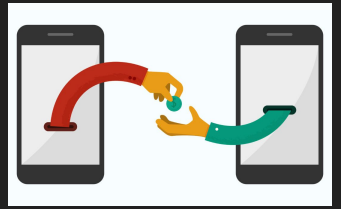
One… Two… Three… Four… now your contract is no more

Trace Vulnerabilities

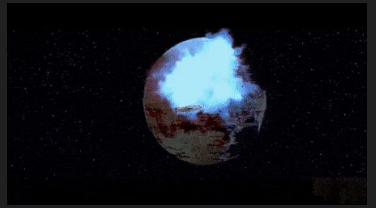
- Execution trace
  - Sequential involcations of a contract
- Two main Categories of Bugs (excluding catastrophically buggy code)
  - Safey
  - Liveliness
- Immutable Code
  - These bugs CANNOT go away unless the contract is killed.
- EVM takes in **code** and **gas**, can update **storage** and **balance**
- Involcation depth
- σ(C), AKA the blockchain state
- Invalid State -> “throw”

Important Solidity Functions To Know

send, call, transfer



suicide, selfdestruct



stop, return

Safety Violations

“During executions, nothing bad happens”

Safety Violations

- Prodigal Contracts
  - Definition 3.1 (Prodigal contracts). *A contract C at blockchain state σ(C) is called prodigal if an arbitrary account A can send a zero-Ether trace to C, which when executed results in transfer of Ether from the C to A*
    - CALLVALUE = 0
  - Detection
    - Reach CALL instruction with recipient being the account who called it (A), afterwards reach normal stopping instruction. If it throws, the attack is reverted and thus worthless
    - Reach Suicide instruction with recipient being the **caller**
- Suicidal Contracts
  - Definition 3.2 (Suicidal contracts). *A contract C at blockchain state σ(C) is called suicidal if an arbitrary account can send a trace to C, which when executed, kills the contract.*
  - Detection
    - SUICIDE Instruction is final bytecode

Liveness Violations

“Something good must eventually happen… right”?

Liveliness Violations

- Definition 3.3 (Greedy contracts). *A contract C at blockchain state σ(C) with a non-zero balance is called k-greedy if execution of any trace with invocation depth k for C and sent by any account, does not result in transfer of Ether from C (to any account).*

- Detection
  - Find execution traces up to *k* involcations that do not reach any transferring instructions (CALL)

Questions?

MAIAN

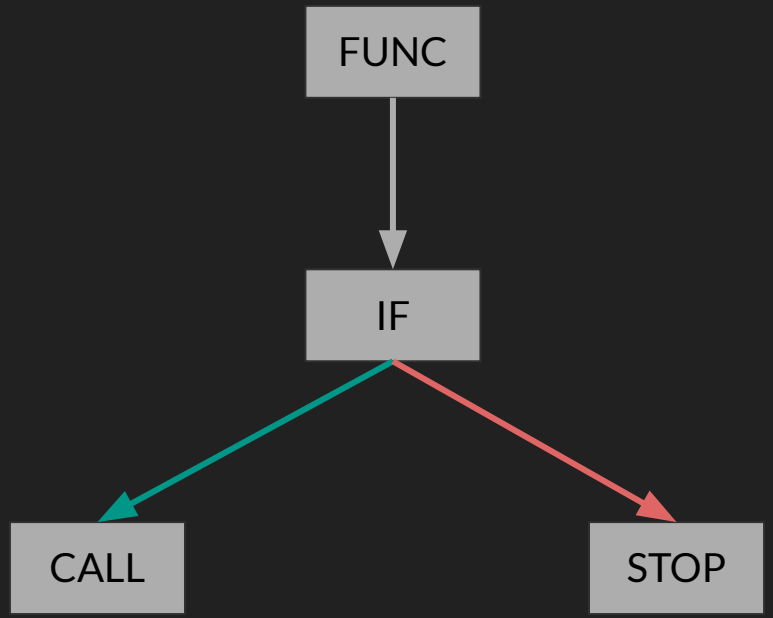
How Trace Vulnerabilities are found

MAIAN - Symbolic Analysis Overview

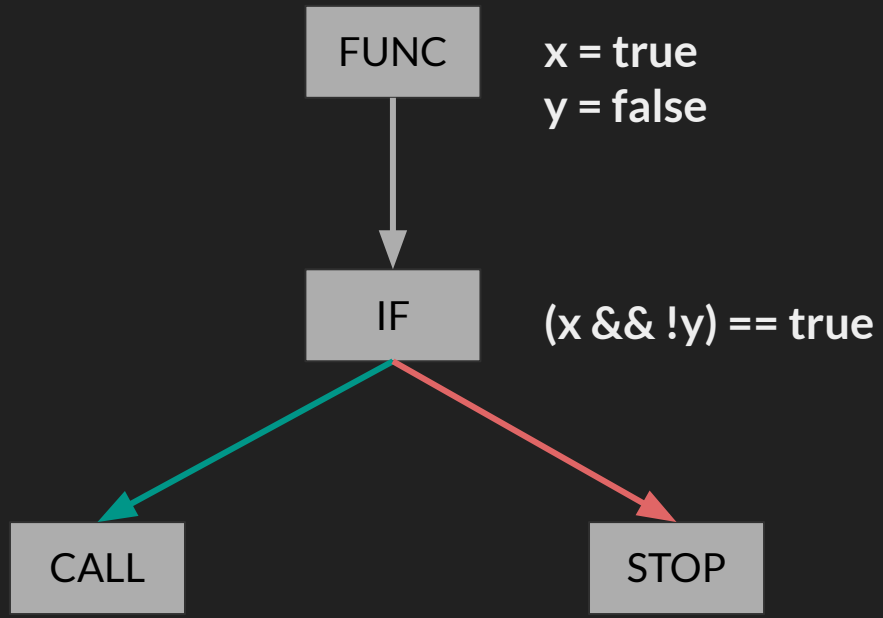
- Given contract bytecode and a start state, what end states can we get into?
  - Reason across **multiple contract calls and blocks**
- Symbolic Analysis and Execution
  - Run all options
    - Given a choice, fork the current execution to check both options
  - Limited by traditional symbolic execution
    - Path Explosion
    - Forking entire system.

Execution Path Search

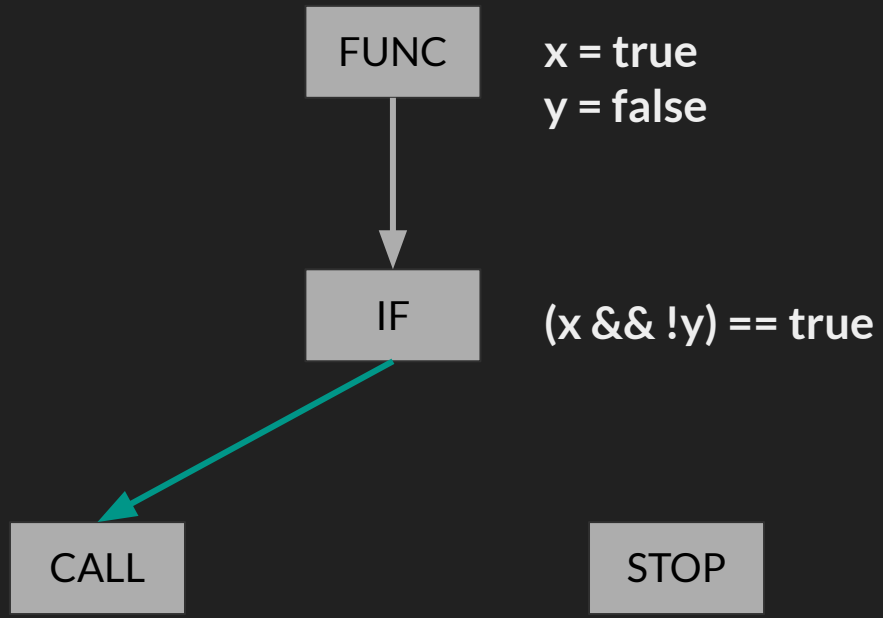
Find what **causes** the CALL



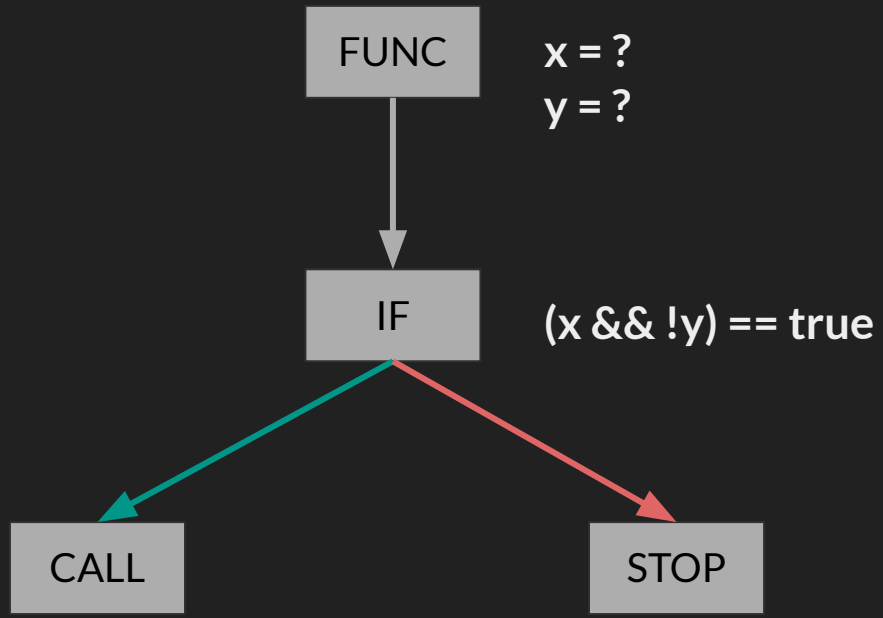
Concrete Evaluation



Search Space Limitation



Find what **causes** the CALL



MAIAN - Execution Path Search

- When at a concrete branch, evaluate it and take the result
  - Limits search space
- Depth First Search
- Memory Management
  - When memory needed, read from main blockchain and locally map
- External Calls
  - Marked as symbolic.
  - MAIAN is **inter-procedural**, not inter-contract.
- Datatype Evaluation
  - SMT Solver

MAIAN - Other Vulnerability Analysis

- Non-deterministic inputs
  - Cannot be evaluated concretely, must be symbolic
- Flagging Violations
  - If safety property violated, flag contract
    and path as buggy candidate. Look at it more later
- Bounding Path Search Space
  - Max Call Depths
  - Maximum Nodes
    - Loops
  - Max analysis time
- Pruning
  - Memoization

MAIAN - Attack Validation

Question: How can we be sure the attack works?

Answer: Actually simulate the attack on the entire blockchain

- MAIAN downloads the current version of the Blockchain
- Runs the attack on the target contract
- Observes the result
  - If it happens there, it would happen **in the wild.**

Questions?

Discussion Question

Do these trace vulnerabilities actually cover every possible case of such buggy contracts?

Evaluation - Scope

- 970,898 Smart Contracts
- 9,825 pieces of source code
- Started at block 4.8 Million

- Hardware
  - Linux Box
  - 64 Bit Ubuntu 16.04.3
  - 64 GB Ram, 40 CPU Intel Xeon E5-2680 2.8Ghz (most experiments run on 32 cores).

31,201

Greedy contracts flagged (1,524 distinct)

69% true positive rate

21529 Actually Greedy Contracts

1,504

Prodigal contracts flagged (438 distinct)

97% true positive rate

1,458 Actually Prodigal Contracts

```
1  bytes20 prev;
2  function tap(bytes20 nickname) {
3      prev = nickname;
4      if (prev != nickname) {
5        msg.sender.send(this.balance);
6      }
7  }
```

Leaked 5.0001 Ether!

1,495

Suicidal contracts flagged (403 Distinct)

~ 99% true positive rate

21529 Actually Greedy Contracts

Evaluation - Results

- Found parity attack successfully
- 97%, 69%, and 97% of prodigal, greedy, and suicidal contracts
- 4,905 Ether Extractable
  - 3.4 Million USD at publishing
  - 9.55 Million as of 2/20/2021

Questions?

Discussion Question

What is the next step for a contract that has been identified with a vulnerability? It can't be changed, contracts relying on it can't be changed, how to proceed?

Upgradeability

Conclusion

- Trace Vulnerabilities are critical vulnerabilities that could tear apart the ethereum blockchain if abused.
- An analysis of the entire ethereum blockchain revealed nearly 35,000 potentially vulnerable contracts.
- Symbolic Analysis and Execution is a strong attack vector in cryptosystems & smart contracts because of the impact.
- Mitigations: **Develop using Crytic** (I helped make it :D)

Discussion Questions

(5 or more people asked some form of these first three questions)

Does these vulnerabilities apply in other cryptosystems?

What is the responsible/correct next steps for disclosing this vulnerability?

What mitigation strategies existed at the time, today?

How does MAIAN Operate on libraries or large contracts?

What is the state of MAIAN today?

References

[1] https://medium.com/@ogucluturk/the-dao-hack-explained-unfortunate-take-off-of-smart-contracts-2bd8c8db3562

[2] https://medium.com/new-alchemy/a-short-history-of-smart-contract-hacks-on-ethereum-1a30020b5fd

[3] https://www.coindesk.com/dao-attacked-code-issue-leads-60-million-ether-theft

[4] https://medium.com/cybermiles/i-accidentally-killed-it-and-evaporated-300-million-6b975dc1f76b

[5] https://www.trustnodes.com/2017/11/07/exclusive-parity-hacker-claims-ethereum-newbie-interview