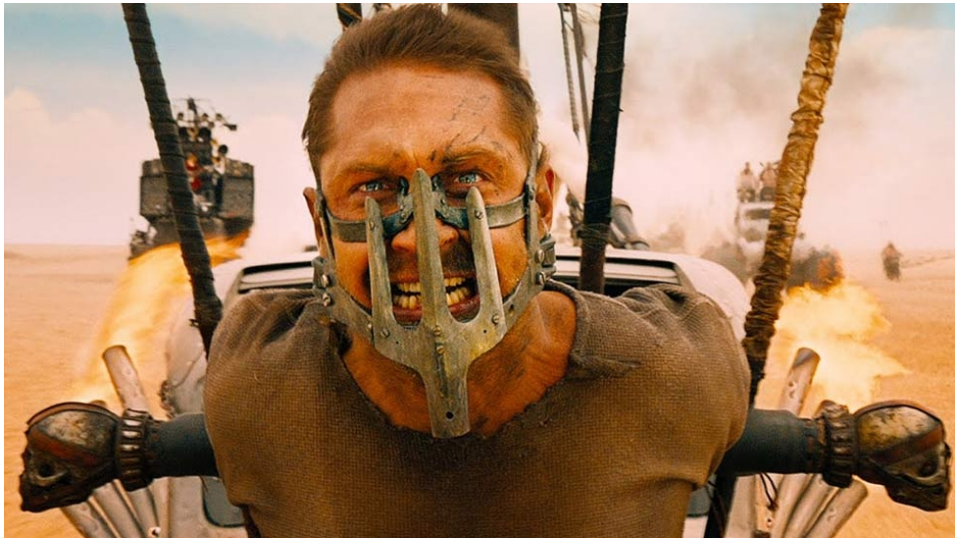


yannis@smaragd.org

Smart Contract Static Analysis: Decompilation and Gas Vulnerabilities

Yannis Smaragdakis (UAthens, Dedaub)

joint work with

Neville Grech (UAthens/UMalta, Dedaub)
Michael Kong (USydney)
Anton Jurisevic (USydney)
Lexi Brent (USydney)
Bernhard Scholz (USydney)

OOPSLA‘18 / ICSE'19

What Do I Do?

Static analysis research

- trying to create a model of all possible program behaviors
- mature framework for Java bytecode, less so for LLVM bitcode

This talk: two pieces of recent work, on Ethereum

- MadMax: detector for gas-related vulnerabilities
- Gigahorse: a decompiler for EVM bytecode (and more)

Secret Sauce: Declarative Specifications

All analyses specified declaratively, in the Datalog language

- i.e., logical rules (hundreds of them)

E.g.,

```
LoopBoundBy(loop, var) ←
  InductionVar(i, loop),
  !InductionVar(var, loop),
  Flows(var, condVar),
  Flows(i, condVar),
  LoopExitCond(condVar, loop).
```

Blockchain and Cryptocurrencies Background

Cryptocoins

Virtual money

- money = something that I believe has value because I believe that others believe has value
  - no inherent value, only ability to exchange
- usually this collective hallucination ("consensus") starts from a trusted authority
- in cryptocurrencies: decentralized consensus, possible without trusted authority

CRYPTOCURRENCY

Ethereum hits a fresh record high and is up over 13,000% in a year

- The price of ethereum hit an all-time high of $1,417.38 on Wednesday, according to CoinDesk
- The cryptocurrency's price is up around 60 percent in the last week
- Steven Nerayoff, a co-creator of ethereum, said it could "easily" double or triple this year

Arjun Kharpal | @ArjunKharpal

Published 3:16 AM ET Wed, 10 Jan 2018 | Updated 9:56 AM ET Wed, 10 Jan 2018

CNBC

$14,216.94 ▲4.39%

Today's Open	$13,619.03	Change	▲$597.91
Today's High	$14,307.53	Market Cap	$0.239T
Today's Low	$13,401.24	Supply	16,804,188

Smart Contract Static Analysis

Cryptography for our Purposes

Two main functions:

- unforgeable signatures, identification
- publication of boxes with locks that only I can open
  - an infinite number of boxes, of all possible sizes, can fit other boxes inside

“Have” = “Know”

Blockchain

A decentralized ledger of transactions

- maintained by untrusted peers

Continuously expanding chain of blocks

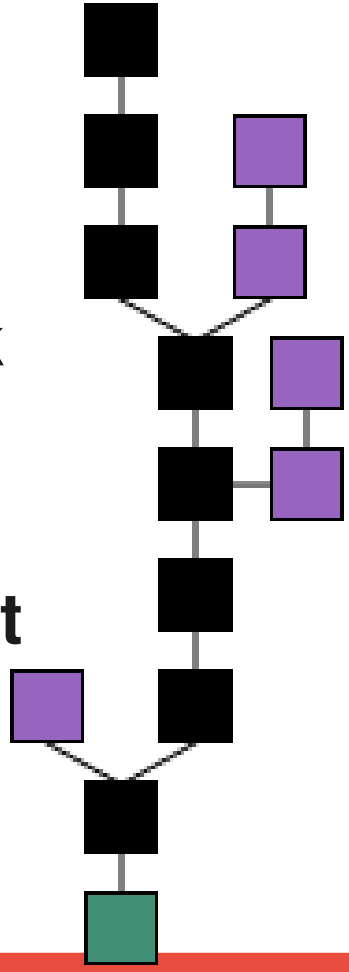
- longest chain is accepted as valid

Peers collect transactions, try to form new block

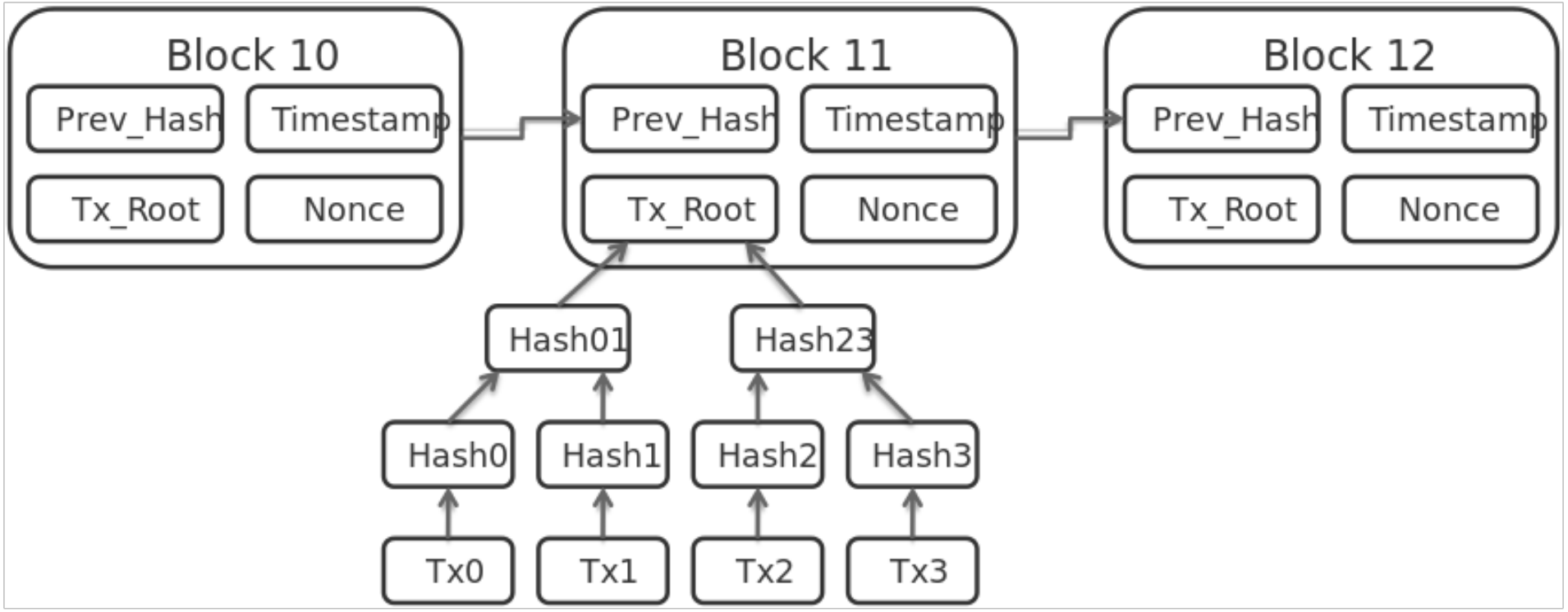
- by mining: solving a crypto-puzzle (proof of work)
- reward for solver (“miner”)

Peers accept the block if transactions consistent

Blocks sign previous ones



Example Structure



Ethereum Blockchain

Main novelty: smart contracts

- complete programs, persistently on the blockchain
- accounts managed by smart contracts
- can call into them, starts a transaction

Gas: fee paid for running them

- translated in Ether (the Ethereum currency)
- bounded/hard coded

Security Threats

Digital currency Ethereum is cratering because of a $50 million hack

Rob Price

Jun. 17, 2016, 10:34 AM 30,040

FACEBOOK LINKEDIN TWITTER EMAIL PRINT

The value of the digital currency Ethereum has dropped dramatically amid an apparent huge attack targeting an organisation with huge holdings of the currency.

The price per unit dropped to $15 from record highs of $21.50 in hours, with millions of units of the digital currency worth as much as $50 million stolen at post-theft valuations.

At a pre-theft valuation, it works out as a staggering $79.6 million.

Martin Hunter/Getty Images

Security

Parity's $280m Ethereum wallet freeze was no accident: It was a HACK, claims angry upstart

And we have evidence to prove it, says biz stiffed out of $1m

By Iain Thomson in San Francisco 10 Nov 2017 at 22:40 78 SHARE ▼

DAO Hack

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

DAO Hack

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

contract Attack {
  ... function() { dao.withdraw(10); } ...
}
```

Gigahorse Decompiler

Go to http://contract-library.com

EVM Bytecode Decompilation is Hard!

- Ethereum vs. JVM/CIL bytecode
  - No data structures, objects, methods or types
  - Stack depth can be different under different control flow paths
  - All control-flow edges (jumps) are variables, not constants
  - All functions of a contract are fused in one (jumps transfer control)

Decompilation: Stratification Points

Original Bytecode

Whole program 3 address IR + CFG

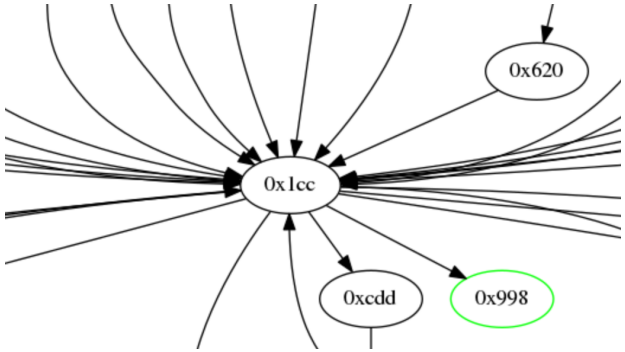
3 address IR + function bounds + local CFGs

Functional 3-address code

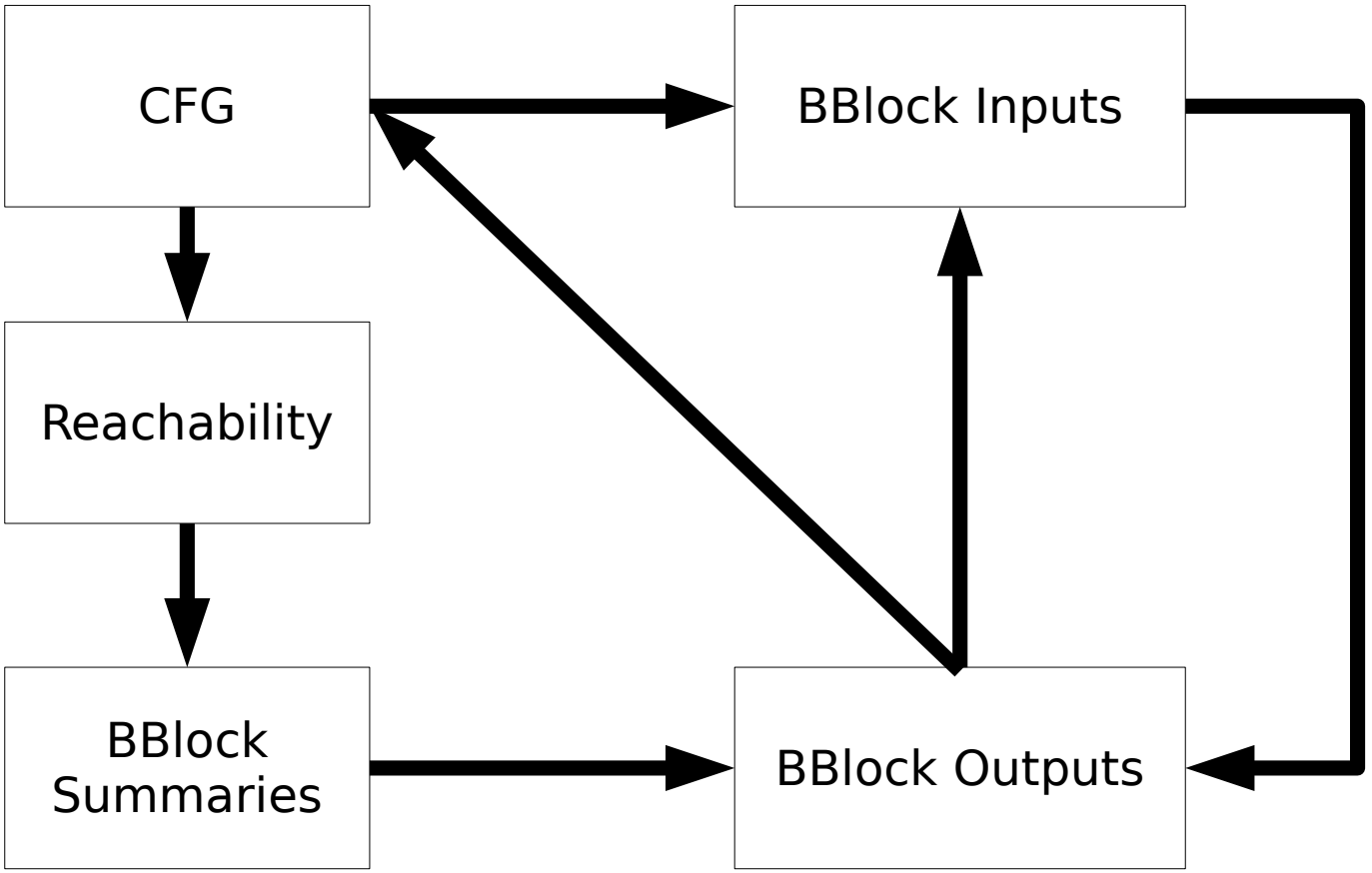
1. Whole contract ctx & flow sensitive analysis

2. Function extraction algorithms

3. Function argument inference with flow sensitive analysis



Large-Scale Recursion

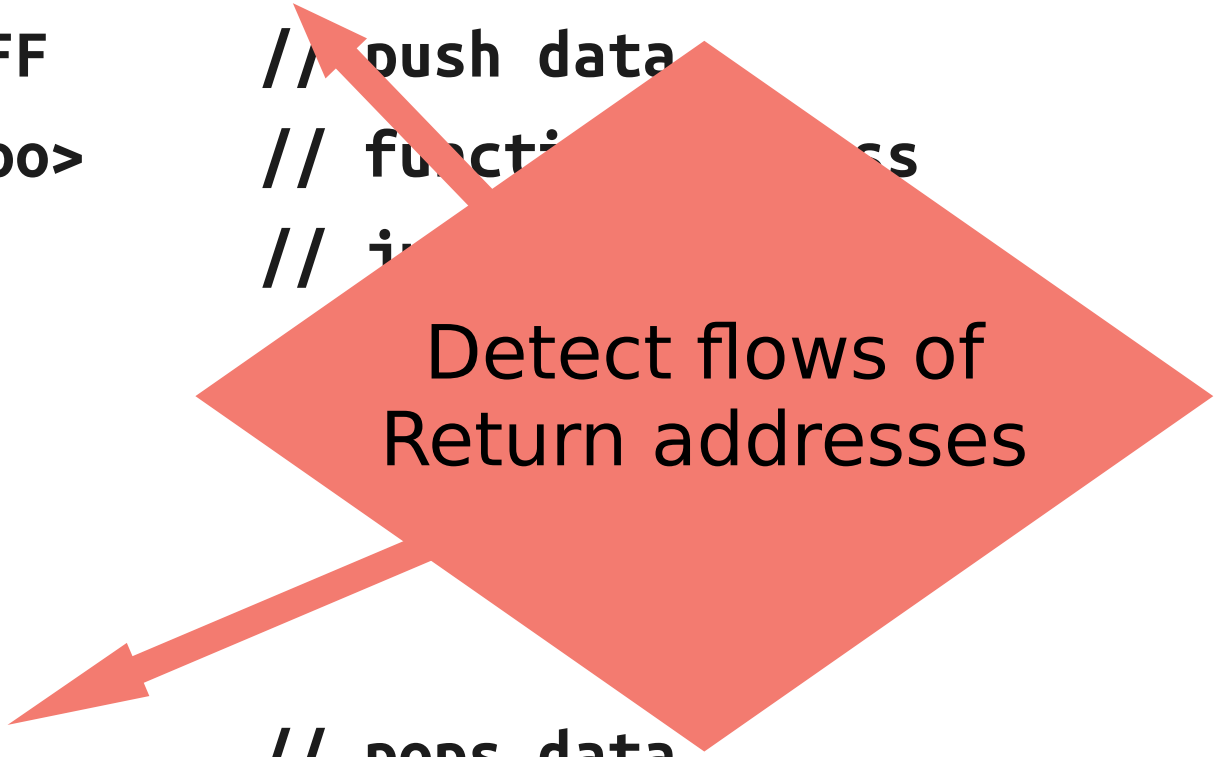


Heuristics: Functions That Return

```
          PUSH4 <return>   // return address
          PUSH4 0xFF       // push data
          PUSH4 <foo>      // function address
          JUMP             // jumps to 'foo'
return:   JUMPDEST
          ...
          ...
foo:      JUMPDEST
          POP              // pops data
          JUMP             // jumps to 'return'
```

Heuristics: Functions That Return

```
          PUSH4 <return>   // return address
          PUSH4 0xFF       // push data
          PUSH4 <foo>      // functi      ss
          JUMP             // j
return:   JUMPDEST
          ...
          ...
foo:      JUMPDEST
          POP              // pops data
          JUMP             // jumps to 'return'
```



Heuristics: Finding More Functions

```
i = 1.
do {
  InFunction_i(block, block) ←  FunctionEntry_i-1(block).
  InFunction_i(next, func) ←
    InFunction_i(block, func),BlockEdge(block, next),
    !FunctionCall_i-1(block, next), !Function_Exit(block).

  FunctionCall_i(prev, block), FunctionEntry_i(block) ←
    InFunction_i(block, f1), InFunction_i(block, f2), f1 != f2,
    BlockEdge(prev, block), !FunctionExit(prev),
    !InFunction_i(prev, f1), !InFunction_i(prev, f2).

  i = i + 1.
} until fixpoint(FunctionEntry)
```

Output IR After Function Arg Inference

```
private 0xa3b (va1, va2, va3) → (int4, int16)
    f1 := CONST 0xa4b
    ret := CONST 0x3f
    v1, v2 := CALLPRIVATE(f1, ret, va2)
    r1 := SHA3(va2, va3)
    RETURNPRIVATE va1, r1, v1;
}
private 0xa4b(va1, va2) → (int4, int16)
...
}
```

Implementation

- A few (<5) KLoC of Datalog
- Decompiles 99.9% of entire Ethereum blockchain in 2 hours

MadMax: Gas-Focused Vulnerability Detection

What is MadMax? [OOPSLA'18]

Cutting-edge (exhaustive) *static analysis*

- Abstract Interpretation, CFA Flow Analysis, memory modeling

Performs analysis directly on the bytecode

- Source code only available for 0.34% of contracts (Etherscan)

Evaluated on the entire Ethereum blockchain

- Found $5B on vulnerable contracts (81% estimated precision)

Gas-focused vulnerabilities

Gas Focused Vulnerabilities

- **Gas is needed to execute contracts:**
  - Paid for by the account that calls the smart contract
  - Has monetary value - prevents wasting of resources
  - If not enough gas is budgeted, transaction is reverted
  - Possibly blocking forever due to lack of progress
- **Contract susceptible to DoS attacks if attacker can cause it to require unbounded gas**

Vulnerability 1: Unbounded Mass Ops

```
contract NaiveBank {
  struct Account {
    address addr;
    uint balance;
  }
  Account accounts[];
  function applyInterest() returns (uint) {
    for (uint i = 0; i < accounts.length; i++) {
      // apply 5 percent interest
      accounts[i].balance = accounts[i].balance * 105 / 100;
    }
    return accounts.length;
  }
  function openAccount() returns (uint) { … }
}
```

Vulnerability 2: Wallet Griefing

```
for (uint i = 0; i < investors.length; i++) {
  if (investors[i].invested < min_investment) {
    // Refund, and check for failure.
    // Looks benign but locks entire contract
    // if attacked by a griefing wallet.
    if (!(investors[i].addr.send(investors[i].dividendAmount))) {
        throw;
    }
    investors[i] = newInvestor;
  }
}
```

Vulnerability 3: Integer Overflow

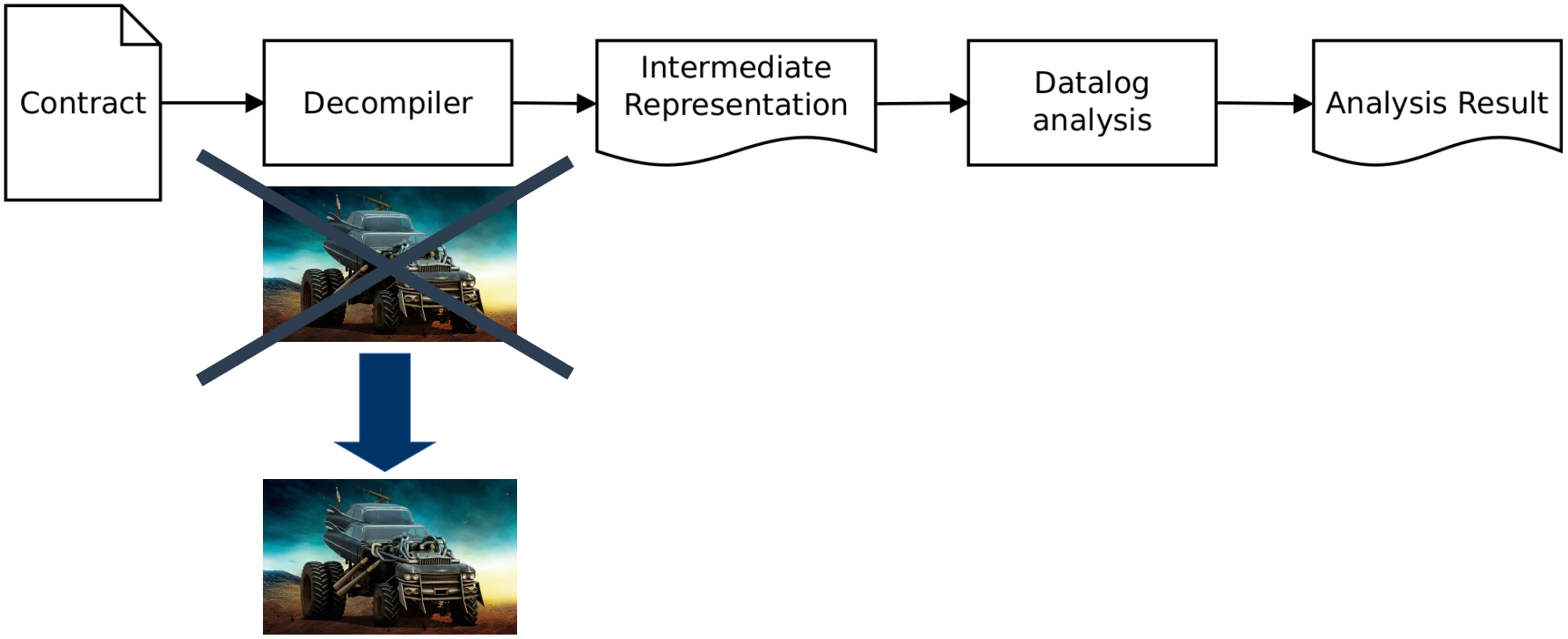
```
contract Overflow {
  Payee payees[];

  function goOverAll() {
    for (var i = 0; i < payees.length; i++) {
    ...
    }
  }
  ...
}
```

uint8

Higher level analyses

Overview of MadMax



Higher Level Analyses

Structured loop reconstruction:

- Induction Variables & Loop Exit Conditions

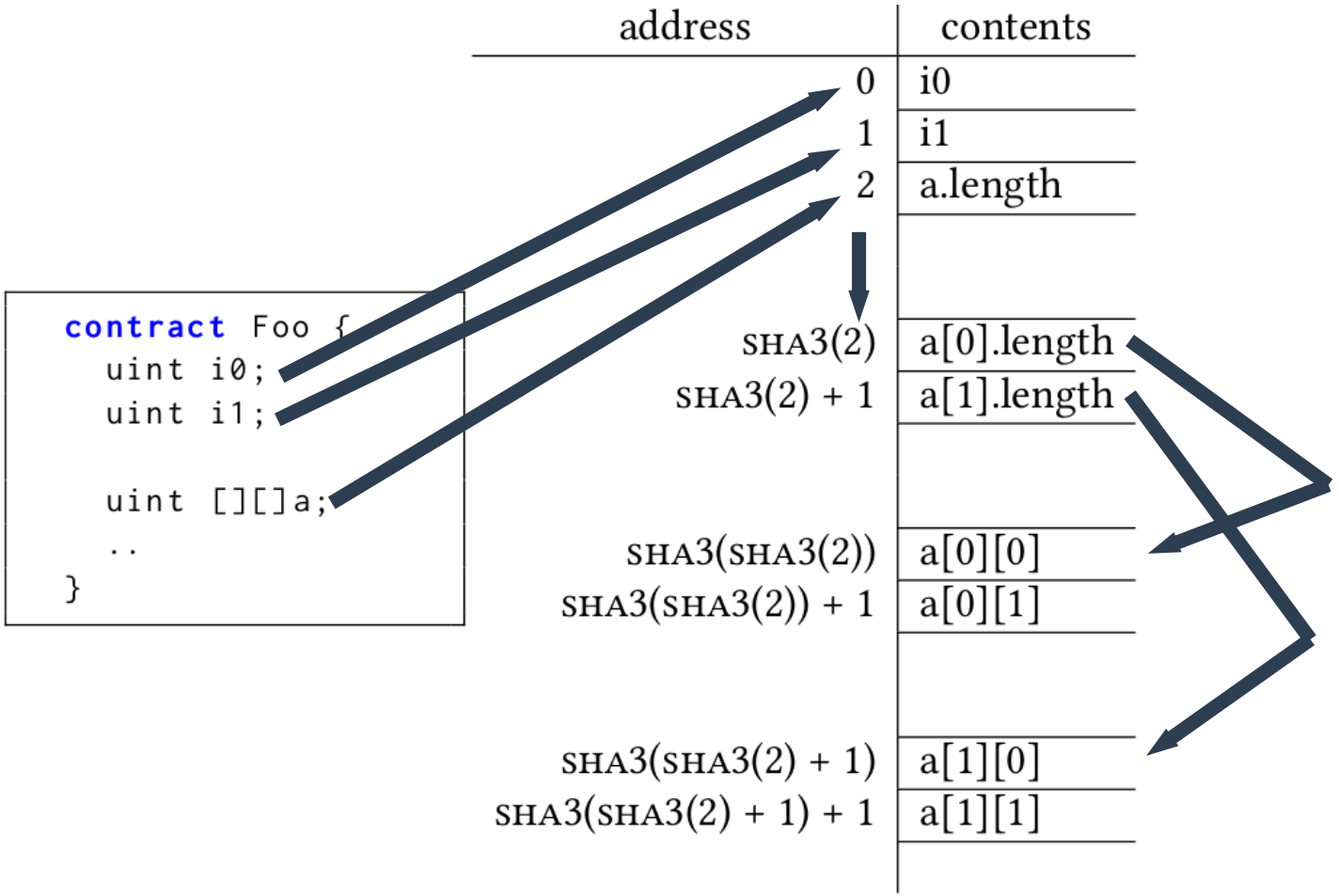
Alias Analyses

High level data structure semantic analysis

Cool concepts such as:

- `IncreasedStorageOnPublicFunction`
- `PossiblyResumableLoop`

Modeling Storage & Data Structures



Example top-level query

```
UnboundedMassOp(loop) ←
  IncreasedStorageOnPublicFunction(arrayId),
  ArrayIdToStorageIndex(arrayId, storeOffsetVar),
  Flows(storeOffsetVar, index),
  VarIndexesStorage(storeOrLoadStmt, index),
  InLoop(storeOrLoadStmt, loop),
  ArrayIterator(loop, arrayId),
  InductionVar(i, loop),
  Flows(i, index),
  !PossiblyResumableLoop(loop).
```

Experimental Evaluation

Results: Effectiveness

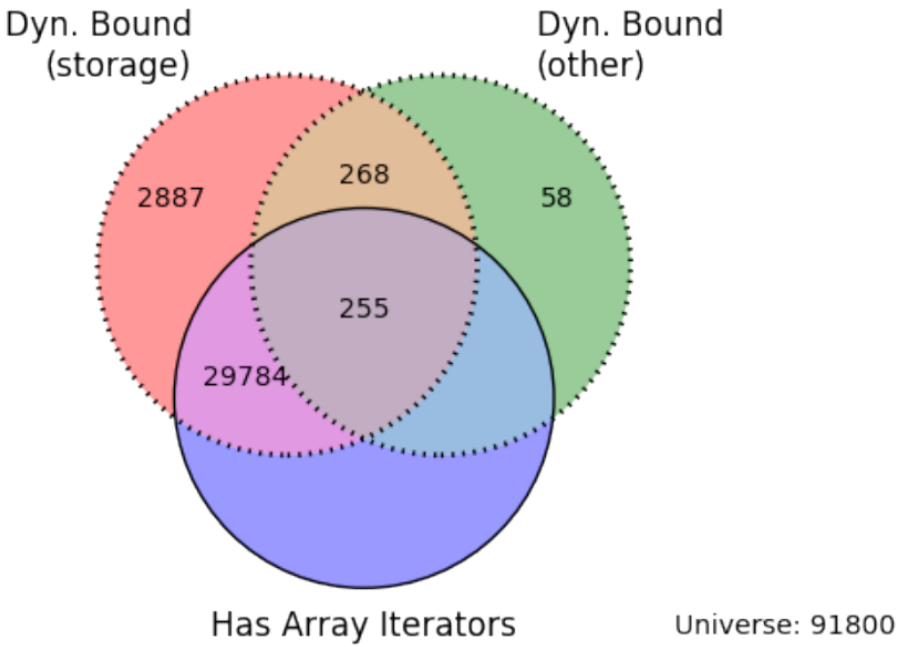
Analysed entire blockchain:

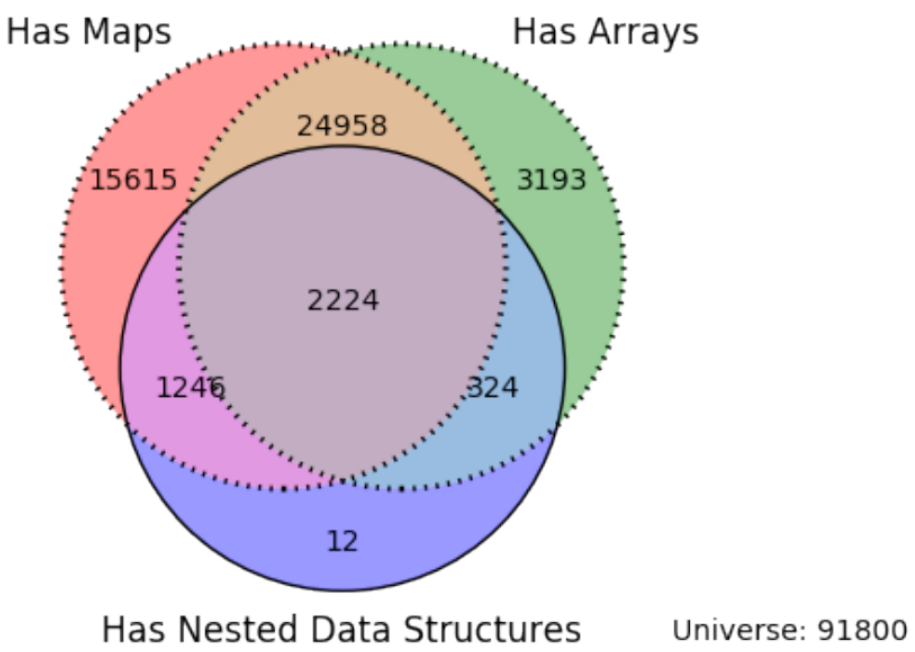
- 6.33M contracts (90k unique) in 10 hours
- 4.1% susceptible to unbounded iteration.
- 0.12% susceptible to wallet griefing.
- 1.2% susceptible to loop overflows.

Combined holding of 7.07 million ETH

81% estimated precision

Insights: Iteration and Data Structures





Reconstructing high level data structure semantics critical for low false positive rate.

Related work

Approach	Works	Soundy	Automated	Bytecode	General
Symbolic Execution	- Oyente by Luu et al. (2016) - Maian by Nikolic et al. (2018) - gasper by Chen et al. (2017) - Grossman et al. (2017)				
Formal Verification	- Proofs in Isabelle/HOL by Hirai (2017) & Amani et al. (2018) - Proofs in the K framework by Hildenbrandt et al. (2017) - Formalism of EVM in F* by Bhargavan et al. (2016)				
Abstract interpretation on Solidity	- Zeus by Kalra et al. (2018) - FSolidM by Mavridou and Laszka (2018)				
Abstract interpretation on EVM bytecode	MadMax (OOPSLA'18) (**Our Approach)**				

Conclusions

Datalog lends itself well to:

- Program analyzers (even flow-sensitive ones)
- High level decompilers

MadMax, a vulnerability detection tool:

- Scales to the entire Blockchain
- Interesting results, practical impact

Decompilation a very important step

- Current work focuses on this