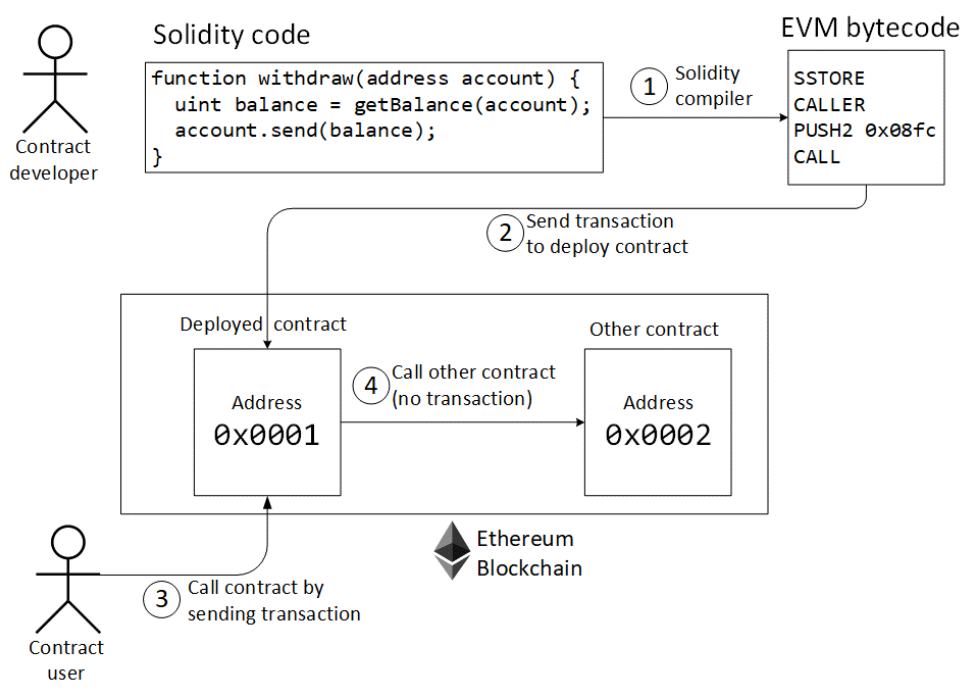
Smart Contract Vulnerabilities: Vulnerable Does Not Imply Exploited

Daniel Perez and Benjamin Livshits, Imperial College London

USENIX Security'21, August 2021

Ethereum Smart Contracts

- Programs deployed on the Ethereum blockchain
- Usually written in Solidity, compiled into EVM bytecode
- Can transfer money to other addresses (including contracts)
- Each instruction execution consumes gas



Smart Contracts: What could go wrong?

TheDAO hack (2016)

- TheDAO raised ***~$150M in ICO***
- Soon after, it got hacked ***~$50M***
- Price of Ether halved
- Ethereum community decided to hard-fork
- Attacker used a ***re-entrancy*** vulnerability

Parity Wallet bug (2017)

- Parity wallet library was used to manage multisig wallet contracts
- Parity ***wallet*** has been ***removed*** due to a "bug"
- ***Dependent contracts*** became unable to send funds
- Around ***$280M frozen***

Common vulnerabilities / bugs

- ***Reentrancy*** can allow an attacker to drain funds
- ***Unhandled exceptions***: can result in lost funds
- ***Dependency on destructed contract***: can result in locked funds
- ***Transaction order dependency***: can allow an attacker to manipulate prices
- ***Integer overflow*** can result in locked fund
- ***Unrestricted action*** can result in an attacker stealing or locking funds

Smart Contract analysis tools

- Usually static analysis and/or symbolic execution
- Work either on Solidity or on the EVM bytecode
- Check for known vulnerabilities/patterns

Securify web interface
https://securify.chainsecurity.com

Smart Contract exploitation

According to analysis tools

- Thousands of vulnerable contracts
- Hundreds of millions USD at risk

In the real-world

- TheDAO and Parity wallet bug were impressive
- Otherwise, a few attacks here and there

⇒ We analyze the Ethereum blockchain to look for ***actual exploitation***

Dataset

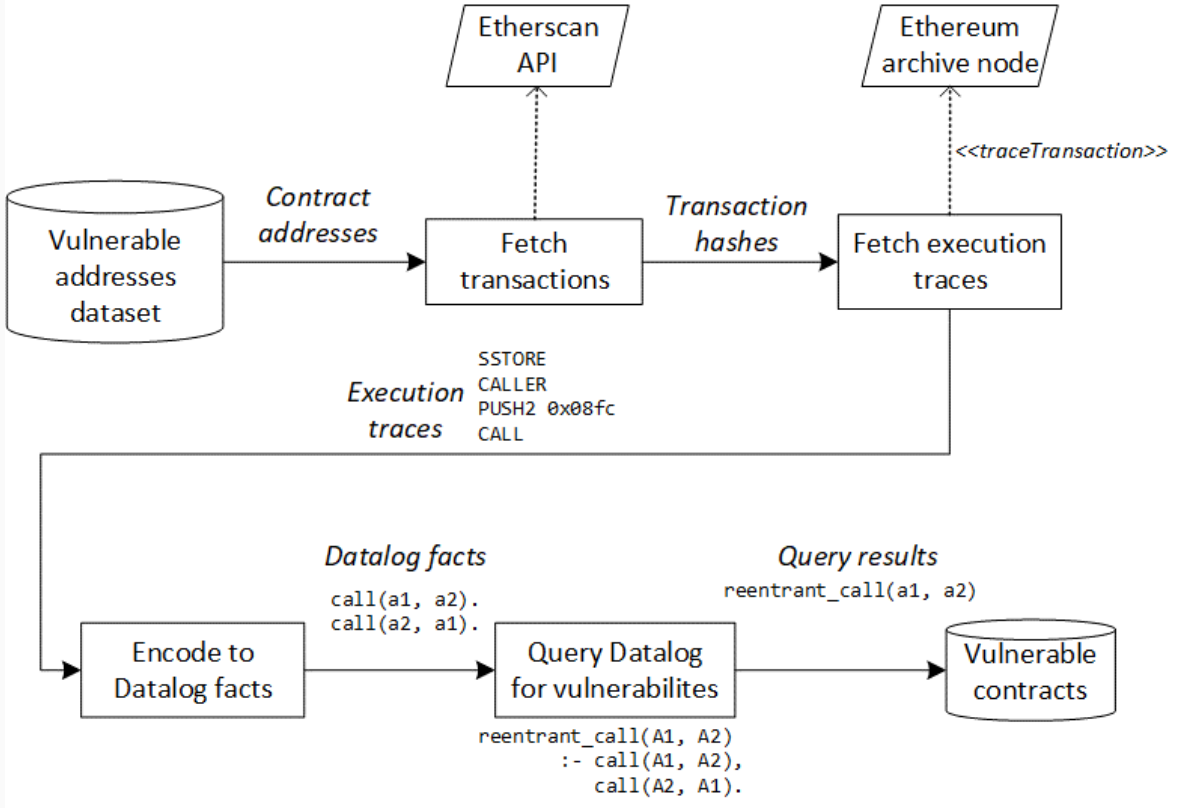
- Data received from the paper authors
- Received replies from 5 out of 8 of the authors we contacted
- Ether at stake computed at time of the report
- Total of around ***3M ETH at stake***

Name	Contracts analyzed	Vulnerabilities found	Ether at stake
Oyente	19,366	7,527	1,289,177
Zeus	1,120	855	729,376
Maian	NA	2,691	14.13
Securify	29,694	9,185	719,567
MadMax	91,800	6,039	1,114,692

Dataset overview

Detection overview

1. Retrieve all transactions
2. Retrieve execution traces for all transactions
3. Encode execution traces to Datalog
4. Query Datalog for vulnerabilities



Checking for re-entrancy

1. Record all calls

2. Check for mutually recursive calls between contracts

```
call_flow(A, B) :- call(A, B).
call_flow(A, B) :- call(A, C), call_flow(C, B).
reentrant_call(A, B) :- call_flow(A, B),
                        call_flow(B, A),
                        A != B.
```

Datalog rules

```
function payMe(address account) public {
  uint amount = getAmount(account);
  // XXX: vulnerable
  if (!account.send(amount))
    throw;
  balance[account] -= amount;
}
```

Vulnerable contract

```
function () {
  victim.payMe(owner);
}
```

Attacker contract

Checking for unhandled exceptions

1. Record all call results
   - Top value on the stack after a call
2. Check if the return value is used in a condition (**JUMPI**)

```
influences_condition(A) :- used_in_condition(A).
influences_condition(A) :-
  depends(B, A), used_in_condition(B).

unhandled_exception(A) :-
  failed_call(A), ~influences_condition(A).
```

Datalog rules

```
// allows user to withdraw funds
function withdraw(address account) public {
  uint amount = getAmount(account);
  balance[account] -= amount;
  account.send(amount); // could silently fail
}
```

Problematic contract

Checking for integer overflow

1. Infer variable types from bytecode
   - **SIGNEXTEND** means the variable is signed
   - **AND n 0xff** means it is a **uint8**
2. Compare typed and untyped results

The approach can lead to some false-positives

```
function overflow(uint fee) {
  uint amount = 100;
  amount -= fee;
  msg.sender.send(amount);
}
```

Contract vulnerable to integer overflow

Detection results

Vulnerability	Vulnerable contracts	Total Ether at Stake	Contracts exploited	Exploited Ether	% of Exploited Ether
Re-entrancy	4,337	1,518,067	116	6,076	**0.40%**
Unhandled Exception	11,427	419,418	264	271.9	**0.07%**
Destructed contract	7,285	1,416,086	0	0	**0.00%**
Transaction order	1,881	302,679	54	297.2	**0.01%**
Integer overflow	2,492	602,980	62	1,842	**0.31%**
Unrestricted action	5,163	580,927	42	0	**0.00%**
Total	23,327	3,124,433	463	8,487	**0.27%**

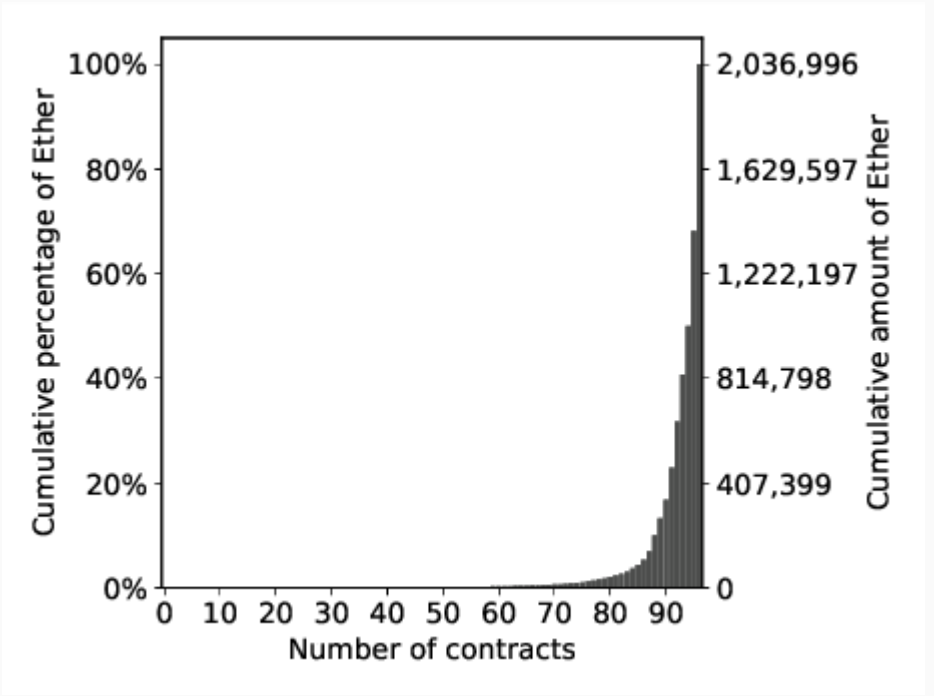
In total, ***at most 0.27%*** of the 3M Ether at stake has been ***exploited***

Takeaways

- **Re-entrancy** seems to be the ***most dangerous*** issue. Good target for static analysis. Better prevention at the EVM level could be helpful.
- **Unhandled exceptions** are ***handled well enough*** by current tooling.
- **Dependency on destructed contract** is mostly a ***non-issue***.
- **Transactions order dependency** does ***not*** seem to be ***used to steal ETH*** directly.
- **Integer overflow** is hard to prevent and consequences ***hard to predict***. Also good target for static analysis.

Contracts wealth distribution

- Combined value: ***~2 million ETH***
- Only ***~2,000*** out of 23,327 contracts ***held Ether***
- Less than 100 contracts had more than 10 Ether
- The ***top 6 contracts held 83%*** of the total Ether



Cumulative Ether held in contracts holding more than 10 ETH

Vulnerable but not exploitable

- Many cases where "vulnerable" contracts are not exploitable
- High-value contracts flagged vulnerable are typically not exploitable
- Usually limitation due to the nature of static analysis

```
function removeOwner(address owner) onlyWallet {
  isOwner[owner] = false;
  // Could in Theory run out of gas
  for (uint i=0; i<owners.length - 1; i++) {
    if (owners[i] == owner) {
      owners[i] = owners[owners.length - 1];
      break;
    }
  }
  // a bit more logic
}
```

Multisig wallet with >350K ETH at address

0x7da82C7AB4771ff031b66538D2fB9b0B047f6CF9

Summary

- Analyzed 23k contracts with ***3M Ether at risk***
- ***At most 0.27%*** of this Ether, less than 10k ETH, was ***exploited***
- Overall, ***high-value contracts*** seem to be ***secure***

Thank you

For questions, please find my contact info at https://daniel.perez.sh