SRL
SOFTWARE RELIABILITY LAB

ETH zürich

SECURIFY

Not Your Grandma’s Smart Contract Verification

Florian Buenzli

Dana Drachsler-Cohen

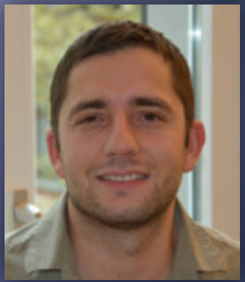
Andrei Dan

Arthur Gervais

Quentin Hibon

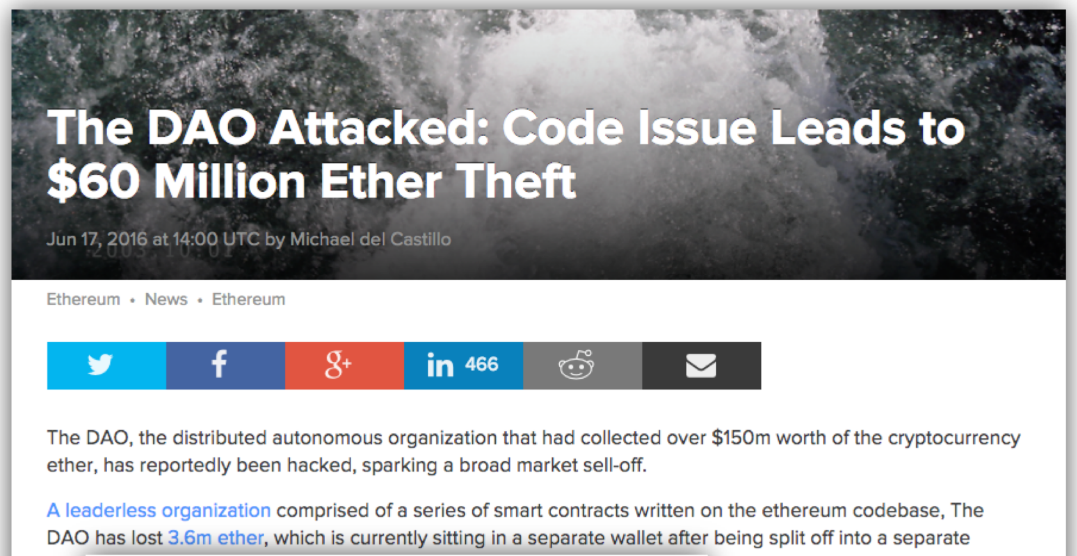
Hubert Ritzdorf

Petar Tsankov


Martin Vechev

http://blockchainsecurity.ethz.ch

Smart Contract *Security Bugs* in the News



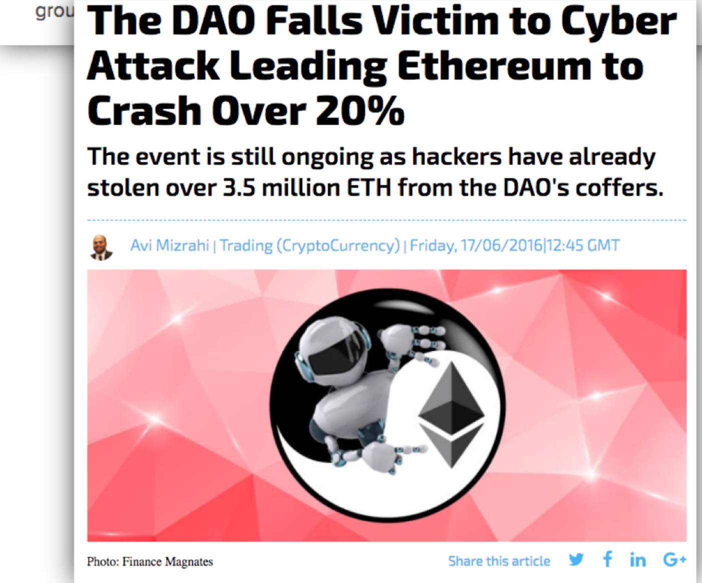
The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

Jun 17, 2016 at 14:00 UTC by Michael del Castillo

Ethereum • News • Ethereum

The DAO, the distributed autonomous organization that had collected over $150m worth of the cryptocurrency ether, has reportedly been hacked, sparking a broad market sell-off.

A leaderless organization comprised of a series of smart contracts written on the ethereum codebase, The DAO has lost 3.6m ether, which is currently sitting in a separate wallet after being split off into a separate grou



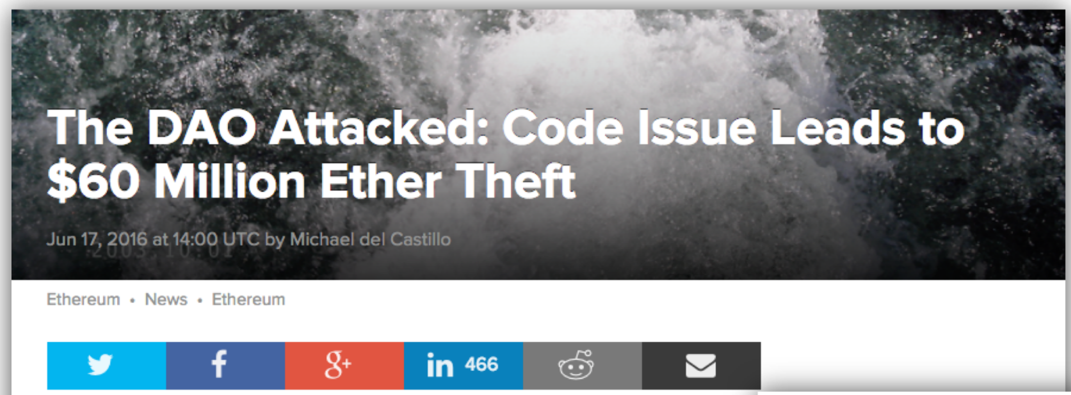
The DAO Falls Victim to Cyber Attack Leading Ethereum to Crash Over 20%

The event is still ongoing as hackers have already stolen over 3.5 million ETH from the DAO's coffers.

Avi Mizrahi | Trading (CryptoCurrency) | Friday, 17/06/2016|12:45 GMT

Photo: Finance Magnates

Smart Contract *Security Bugs* in the News



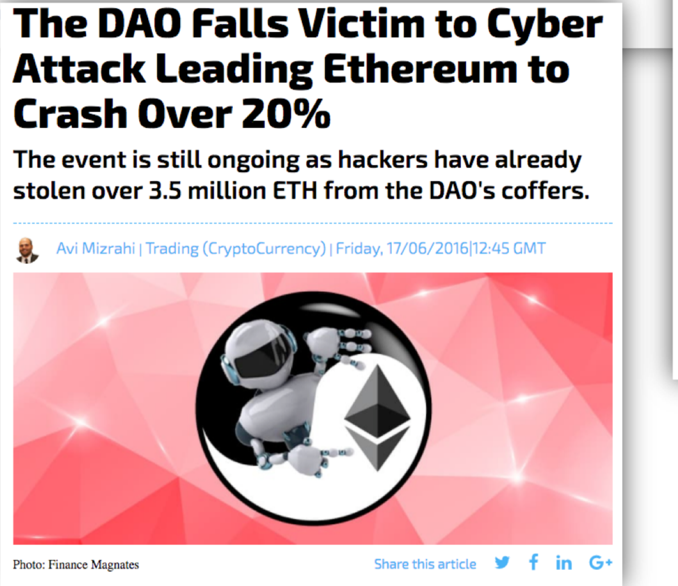
The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

Jun 17, 2016 at 14:00 UTC by Michael del Castillo

Ethereum • News • Ethereum

The DAO, the distributed autonomous organization that had collected over $150r ether, has reportedly been hacked, sparking a broad market sell-off.

A leaderless organization comprised of a series of smart contracts written on the DAO has lost 3.6m ether, which is currently sitting in a separate wallet after being grou



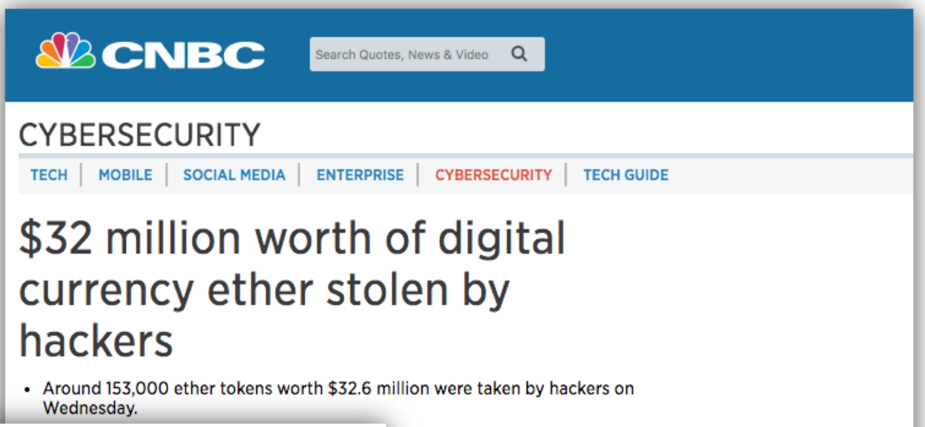
The DAO Falls Victim to Cyber Attack Leading Ethereum to Crash Over 20%

The event is still ongoing as hackers have already stolen over 3.5 million ETH from the DAO's coffers.

Avi Mizrahi | Trading (CryptoCurrency) | Friday, 17/06/2016|12:45 GMT

Photo: Finance Magnates

Share this article



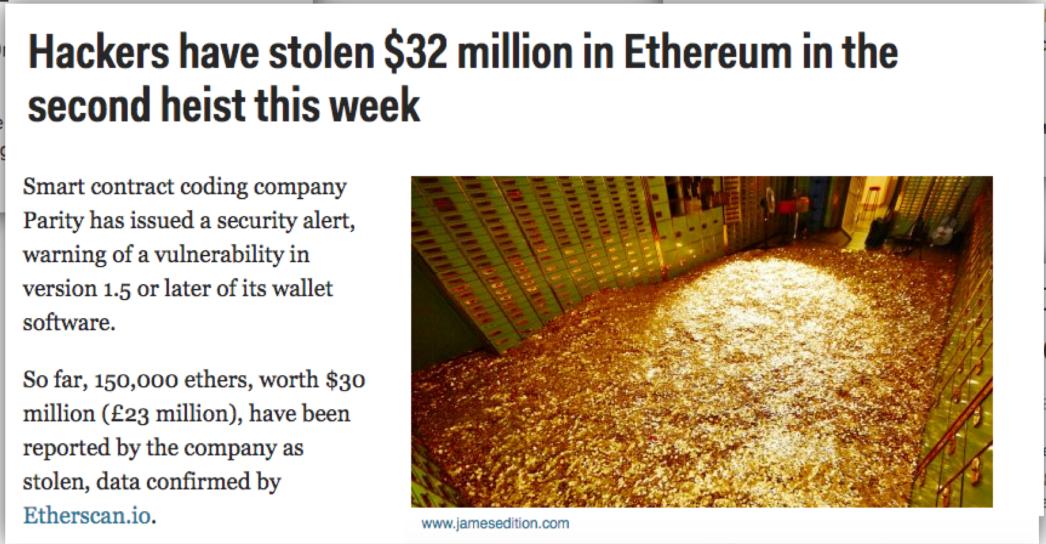
CYBERSECURITY

TECH | MOBILE | SOCIAL MEDIA | ENTERPRISE | CYBERSECURITY | TECH GUIDE

$32 million worth of digital currency ether stolen by hackers

- Around 153,000 ether tokens worth $32.6 million were taken by hackers on Wednesday.
- llet was exploited by hackers.
- day where $7 million worth of ether

ET Thu, 20 July 2017

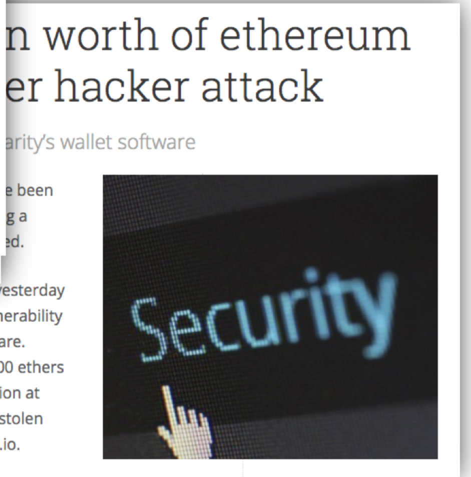


Hackers have stolen $32 million in Ethereum in the second heist this week

Smart contract coding company Parity has issued a security alert, warning of a vulnerability in version 1.5 or later of its wallet software.

So far, 150,000 ethers, worth $30 million (£23 million), have been reported by the company as stolen, data confirmed by Etherscan.io.

www.jamesedition.com



n worth of ethereum er hacker attack

arity's wallet software

e been g a ed.

Smart contract coding company Parity yesterday issued a security alert, warning of a vulnerability in version 1.5 or later of its wallet software. According to the company, so far 150,000 ethers have been stolen, worth nearly $35 million at current price levels. The amount of the stolen ether has been confirmed by Etherscan.io.

Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.transfer(amount);
  }
}
```

Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.transfer(amo
  }
}
```

Only owner can send ether

Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.transfer(amo
  }
}
```

Any user may change the wallet's owner

Only owner can send ether

Unprivileged write to storage

Any user may change the wallet's owner

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.transfer(amo
  }
}
```

Only owner can send ether

An attacker used a similar bug to ***steal $30M*** in July

Unprivileged write to storage, again

MultiWallet Contract

```
uint[256] m_owners;

function initMultiowned(address[] _owners)
only_uninitialized {
  m_numOwners = _owners.length + 1;
  m_owners[1] = uint(msg.sender);
  …
}
```

Unprivileged write to storage, again

MultiWallet Contract

```
uint[256] m_owners;

function initMultiowned(address[] _owners)
only_uninitialized {
  m_numOwners = _owners.length + 1
  m_owners[1] = uint(msg.sender);
  …
}
```

Can only be called once
Relies on a correct deployment

Unprivileged write to storage, again

No check inside the function

MultiWallet Contract

```
uint[256] m_owners;

function initMultiowned(address[] _owners)
only_uninitialized {
  m_numOwners = _owners.length + 1
  m_owners[1] = uint(msg.sender);
  …
}
```

Can only be called once
Relies on a correct deployment

Unprivileged write to storage, again

MultiWallet Contract

```
uint[256] m_owners;

function initMultiowned(address[] _owners)
only_uninitialized {
  m_numOwners = _owners.length + 1
  m_owners[1] = uint(msg.sender);
  …
}
```

No check inside the function

Can only be called once
Relies on a correct deployment

$152 million currently frozen as a result

More Security Bugs…

Unexpected ether flows

More Security Bugs…

Unexpected ether flows

Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*

More Security Bugs…

- Unexpected ether flows
- Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*
- Use of unsafe inputs (e.g., reflection, hashing, …)

More Security Bugs…

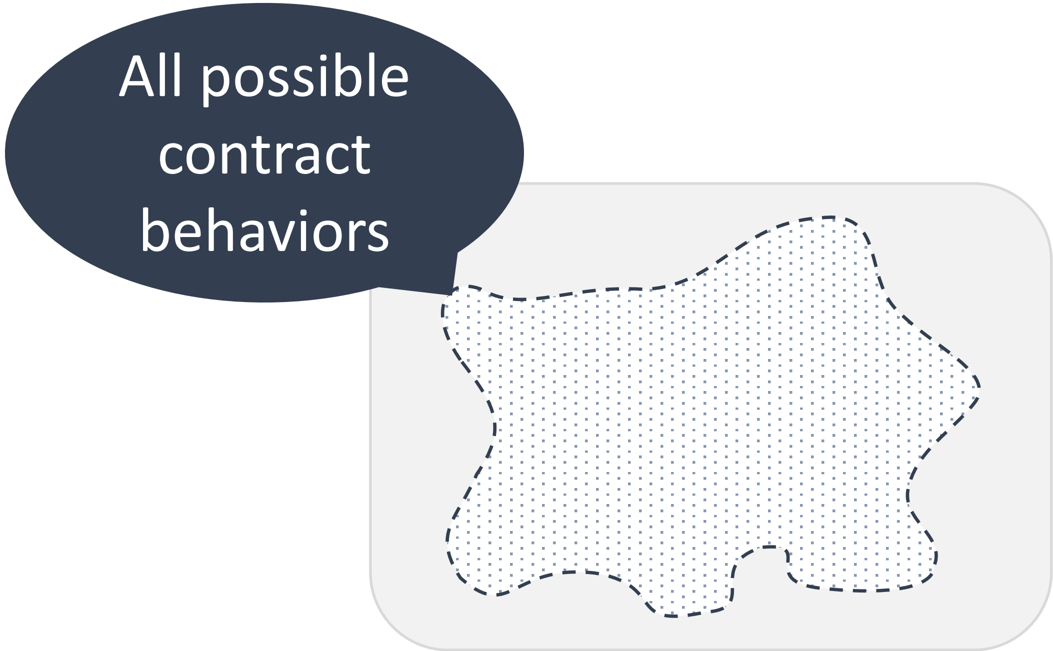
- Unexpected ether flows
- Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*
- Use of unsafe inputs (e.g., reflection, hashing, …)
- Reentrant method calls *(e.g., DAO bug)*

More Security Bugs…

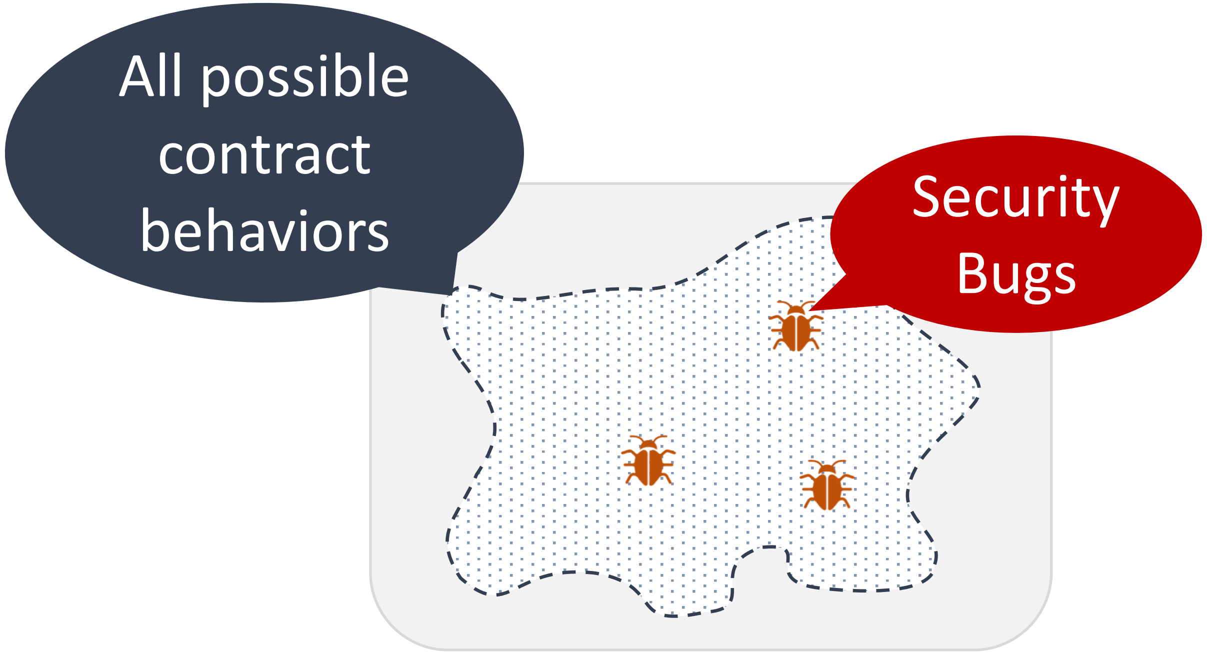
- Unexpected ether flows
- Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*
- Use of unsafe inputs (e.g., reflection, hashing, …)
- Reentrant method calls *(e.g., DAO bug)*
- Manipulating ether flows via transaction reordering

Automated Security Analysis

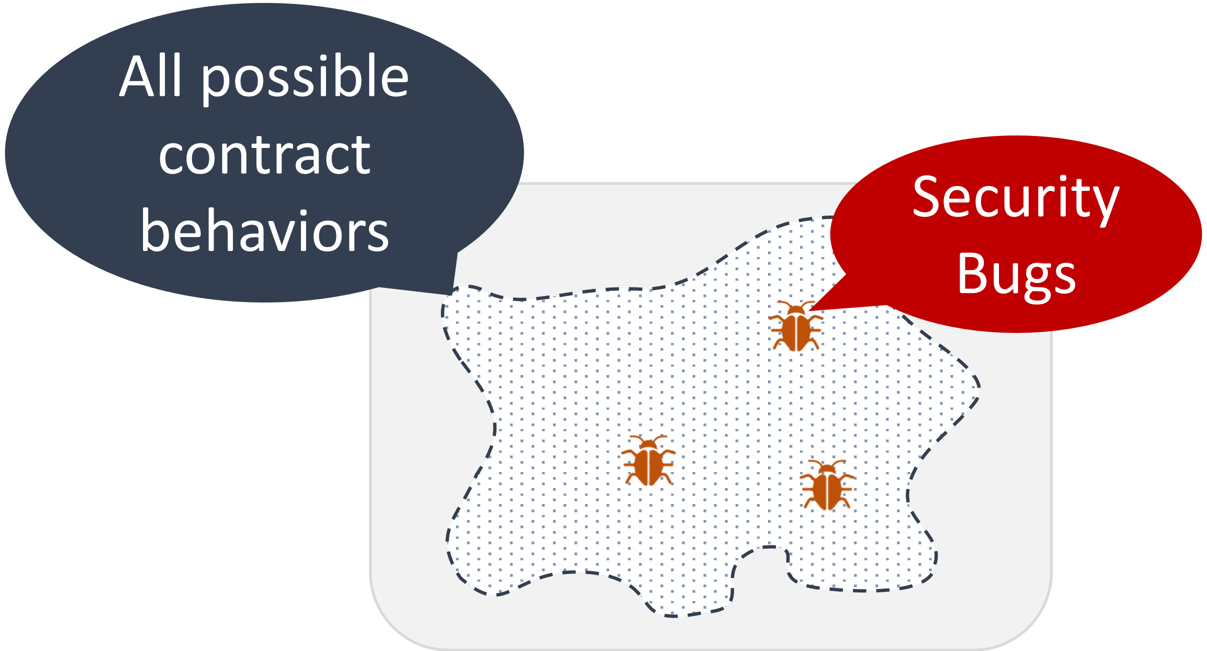
Automated Security Analysis Approaches



Automated Security Analysis Approaches

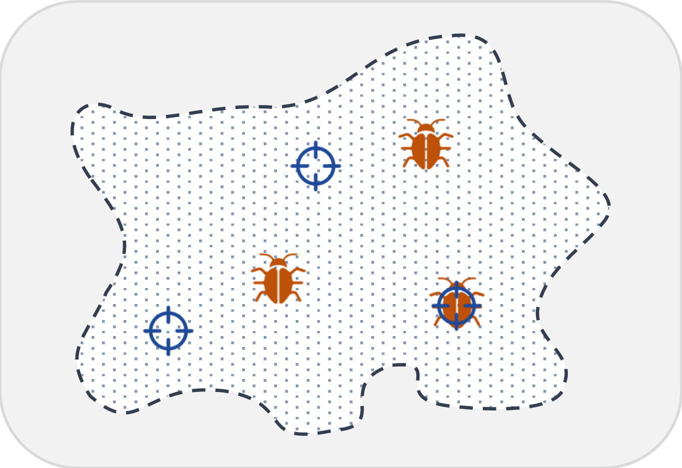


Automated Security Analysis Approaches



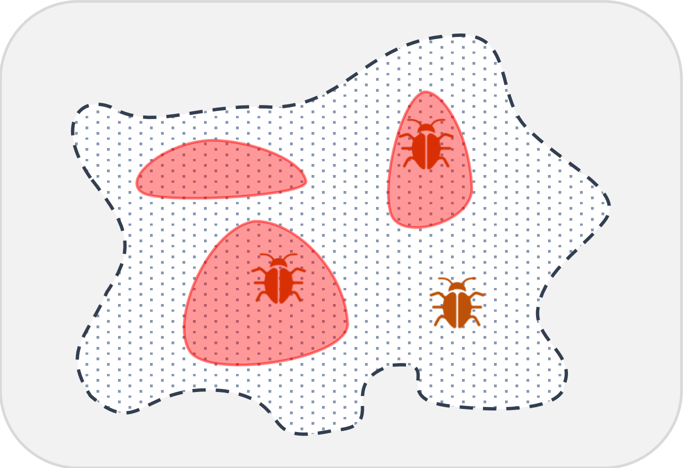
Problem: Cannot enumerate all possible contract behaviors…

Security Analysis Approaches



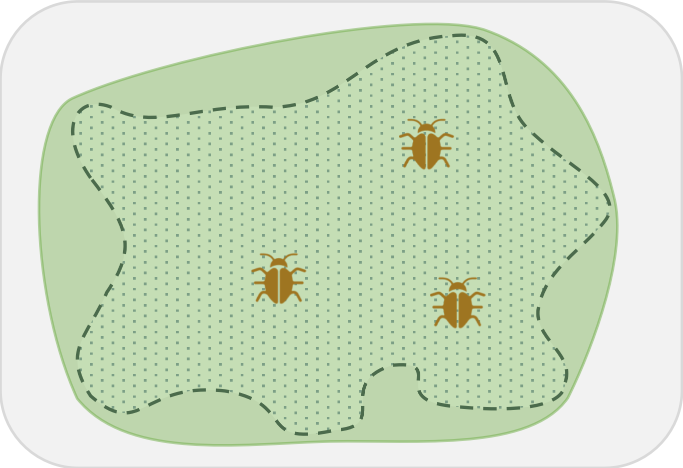
Testing

Report true bugs
Can miss bugs



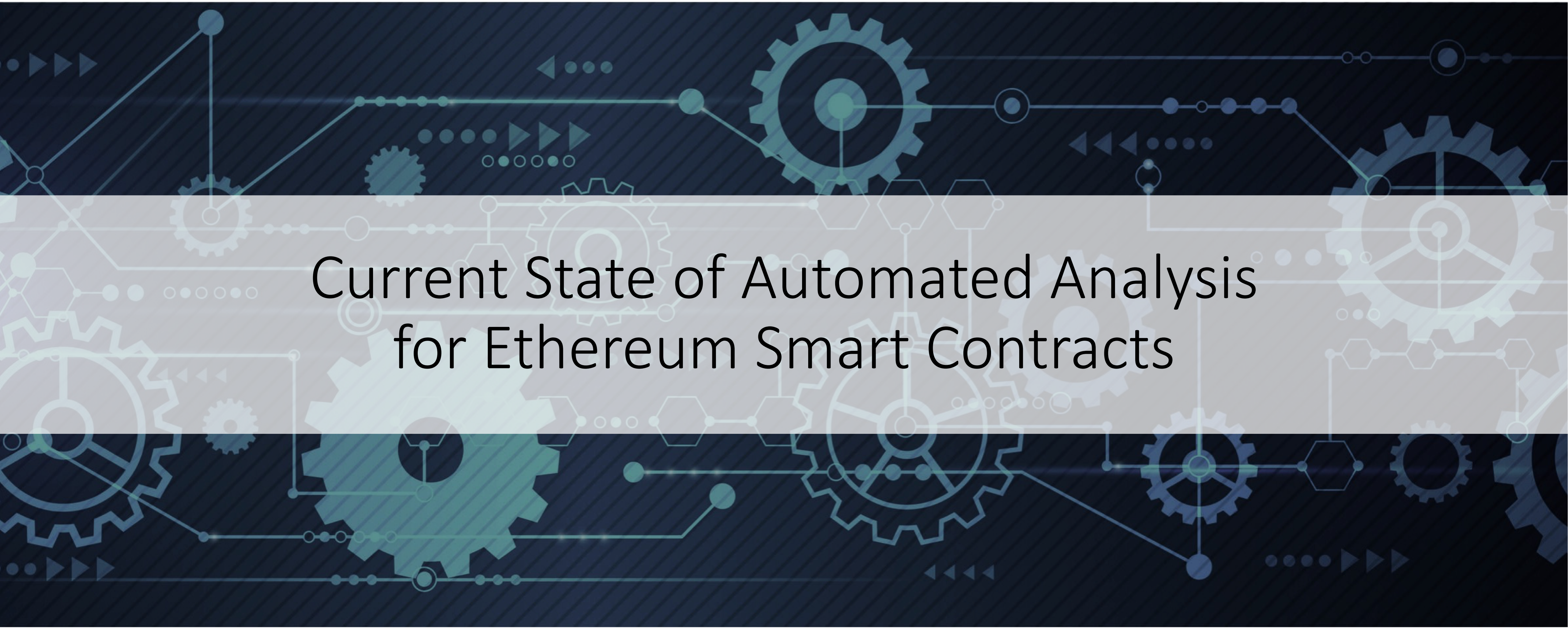
Dynamic (symbolic) analysis

Report true bugs
Can miss bugs



Automated verification

Can report false alarms
No missed bugs



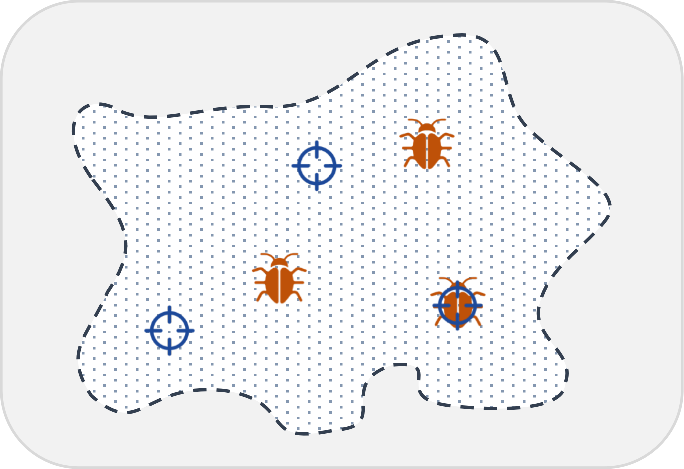
Current State of Automated Analysis for Ethereum Smart Contracts

Security Analysis Approaches

TRUFFLE

Populus

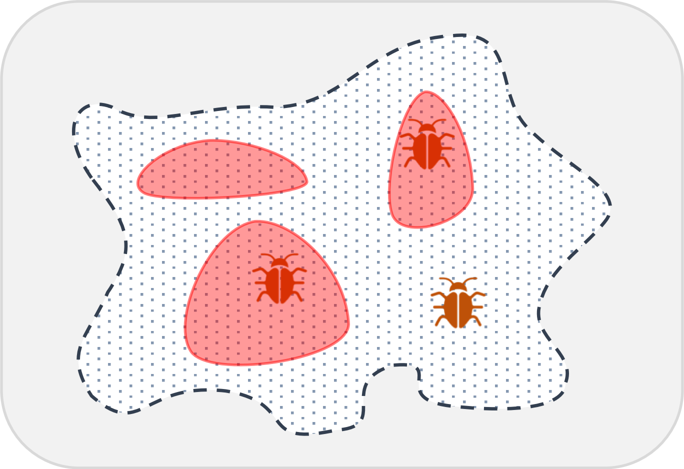
Oyente



Testing

Report true bugs

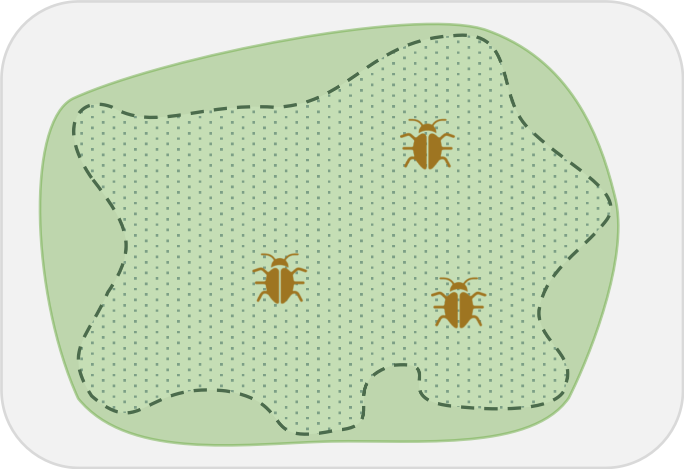
Can miss bugs



Dynamic (symbolic) analysis

Report true bugs

Can miss bugs



Automated verification

Can report false alarms

No missed bugs

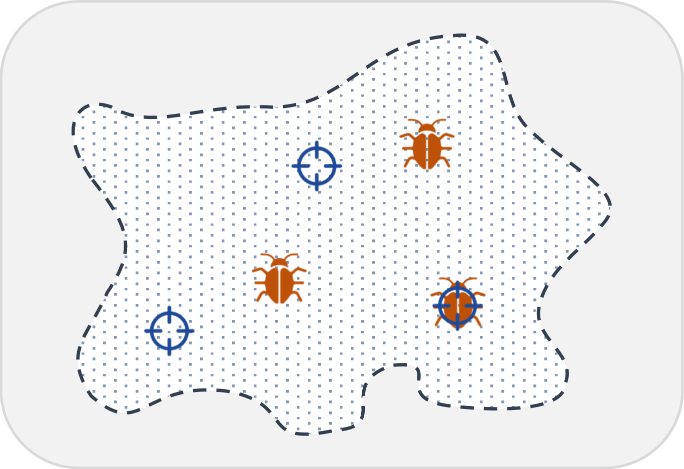
Security Analysis Approaches

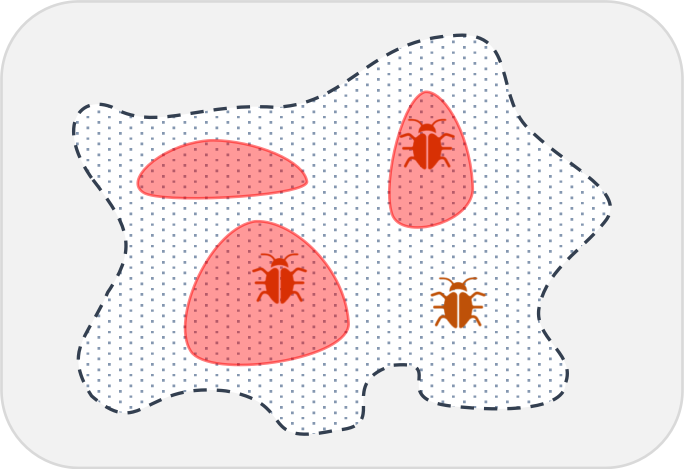
TRUFFLE

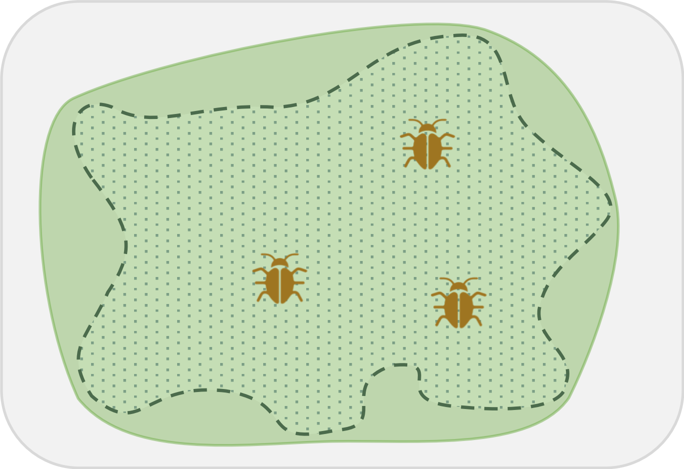
Populus

Oyente

SECURIFY







Testing

Dynamic (symbolic) analysis

Automated verification

Report true bugs
Can miss bugs

Report true bugs
Can miss bugs

Can report false alarms
No missed bugs

SECURIFY

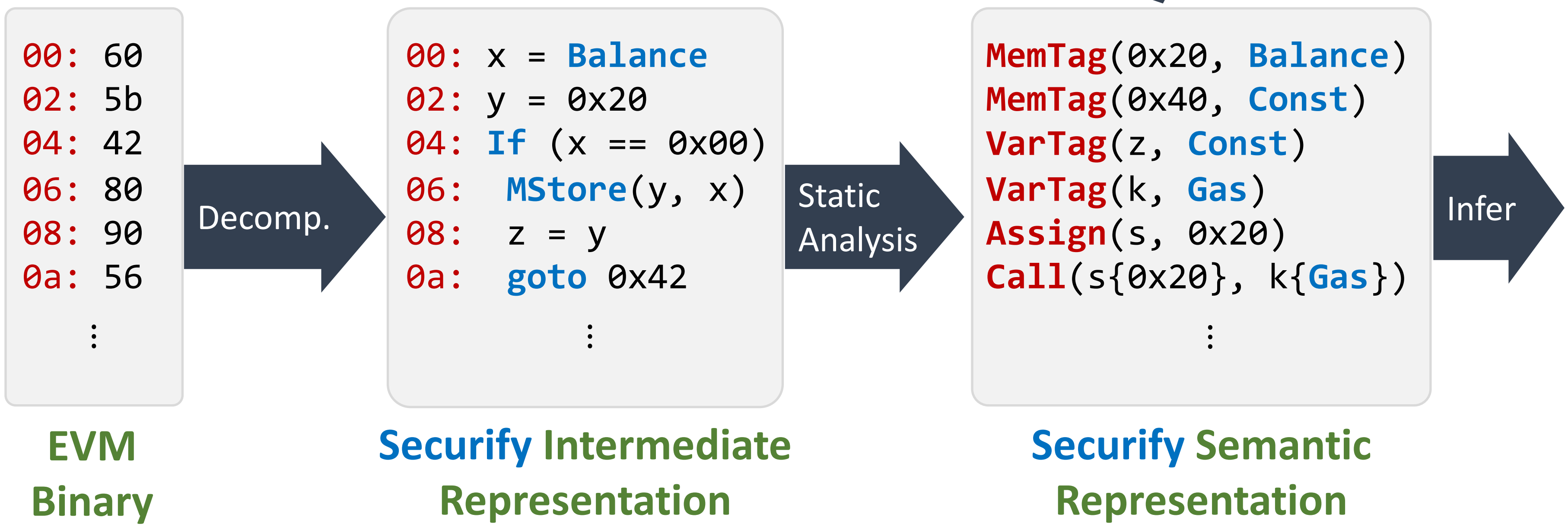
www.securify.ch

Fully ***automated***, one-click, ***formal verification*** system for Ethereum smart contracts

Securify: Under the Hood

Captures key semantic facts about the contract

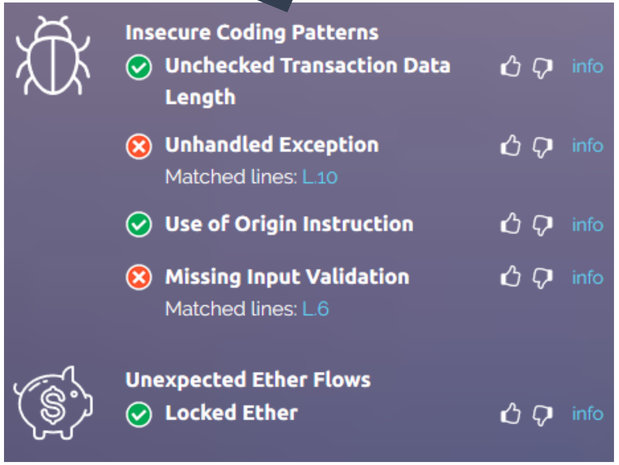
Security patterns expressed in a ***designated security language***



```
00: 60
02: 5b
04: 42
06: 80
08: 90
0a: 56
      ⋮
```

```
00: x = Balance
02: y = 0x20
04: If (x == 0x00)
06:  MStore(y, x)
08:   z = y
0a:  goto 0x42
         ⋮
```

```
MemTag(0x20, Balance)
MemTag(0x40, Const)
VarTag(z, Const)
VarTag(k, Gas)
Assign(s, 0x20)
Call(s{0x20}, k{Gas})
          ⋮
```



Insecure Coding Patterns

✓ Unchecked Transaction Data Length — info

✗ Unhandled Exception — info
Matched lines: L.10

✓ Use of Origin Instruction — info

✗ Missing Input Validation — info
Matched lines: L.6

Unexpected Ether Flows

✓ Locked Ether — info

EVM Binary

Securify Intermediate Representation

Securify Semantic Representation

Securify Report

Securify: Under the Hood

```
00: 60
02: 5b
04: 42
06: 80
08: 90
0a: 56
      ⋮
```

EVM Binary

Decomp.

```
00: x = Balance
02: y = 0x20
04: If (x == 0x00)
06:  MStore(y, x)
08:  z = y
0a:  goto 0x42
          ⋮
```

Securify Intermediate Representation

Static Analysis

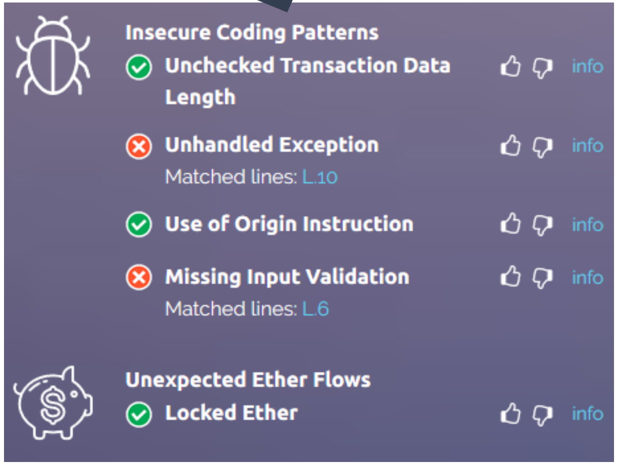
Captures key semantic facts about the contract

```
MemTag(0x20, Balance)
MemTag(0x40, Const)
VarTag(z, Const)
VarTag(k, Gas)
Assign(s, 0x20)
Call(s{0x20}, k{Gas})
          ⋮
```

Securify Semantic Representation

Infer

Security patterns expressed in a ***designated security language***



Insecure Coding Patterns

✓ **Unchecked Transaction Data Length**

✗ **Unhandled Exception**
Matched lines: L10

✓ **Use of Origin Instruction**

✗ **Missing Input Validation**
Matched lines: L6

Unexpected Ether Flows

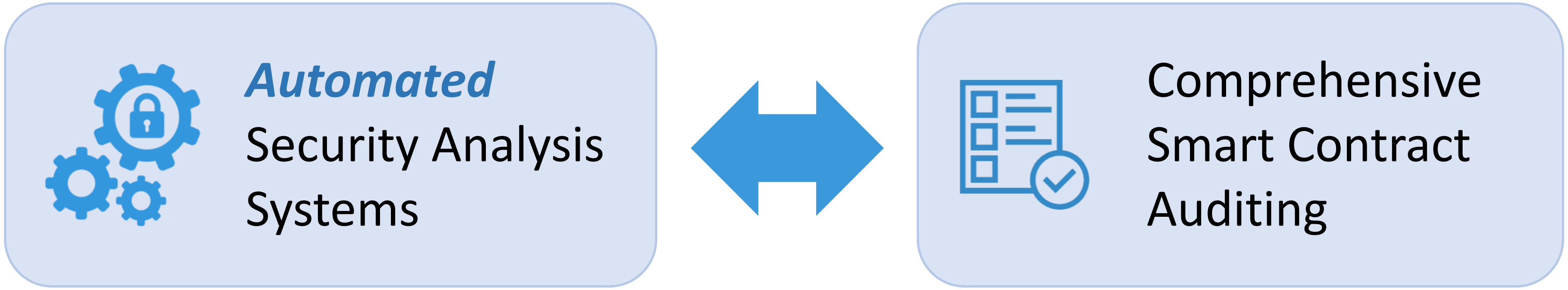
✓ **Locked Ether**

Securify Report

Fully automated, easily extensible

ChainSecurity

Swiss-based startup that provides intelligent security solutions for ***blockchains*** and ***smart contracts***



Automated Security Analysis Systems

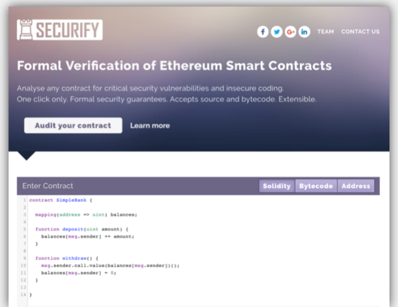
Comprehensive Smart Contract Auditing

https://chainsecurity.com

Summary

Research

- Fully automated
- Strong guarantees
- Extensible



Formal Verification of Ethereum Smart Contracts

Analyse any contract for critical security vulnerabilities and insecure coding.
One click only. Formal security guarantees. Accepts source and bytecode. Extensible.

```
contract SimpleBank {

  mapping(address => uint) balances;

  function deposit(uint amount) {
    balances[msg.sender] += amount;
  }

  function withdraw() {
    msg.sender.call.value(balances[msg.sender])();
    balances[msg.sender] = 0;
  }

}
```

https://www.securify.ch

Product

Get in touch with our team of ***security*** / ***blockchain*** / ***program analysis*** experts

https://chainsecurity.com

contact@chainsecurity.com

@chain_security

Demo

https://vimeo.com/user73702627/securify