Verification of (Ethereum) Smart Contracts

Lars Hupel
INNOQ Technology Lunch
2021-02-10

INNOQ

Conclusion

- Smart Contracts are code.

Conclusion

- Smart Contracts are code.
- Code has bugs.

Conclusion

- Smart Contracts are code.
- Code has bugs.
- Smart Contracts have bugs.

Conclusion

- Smart Contracts are code.
- Code has bugs.
- Smart Contracts have bugs.

The end.

Conclusion

- Smart Contracts are code.
- Code has bugs.
- Smart Contracts have bugs.

The end?

We are barely beginning to understand smart contracts, their engineering and their implications.

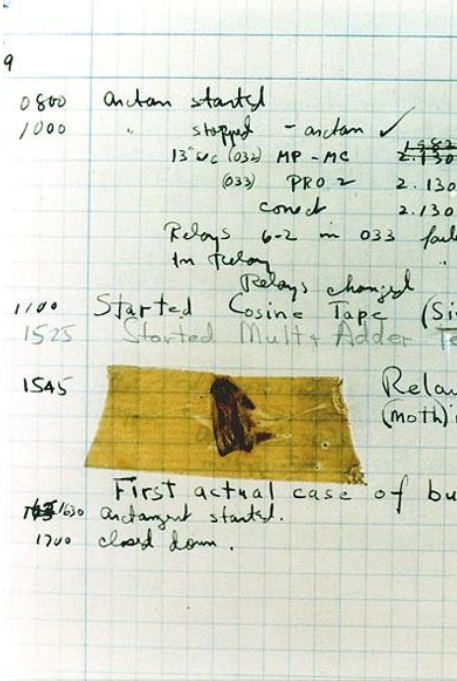
Smart Contracts

Smart Contracts

- digitized contracts with conditions and protocols
- fully automated
- immutable
- decentralized
- without intermediary

Bugs

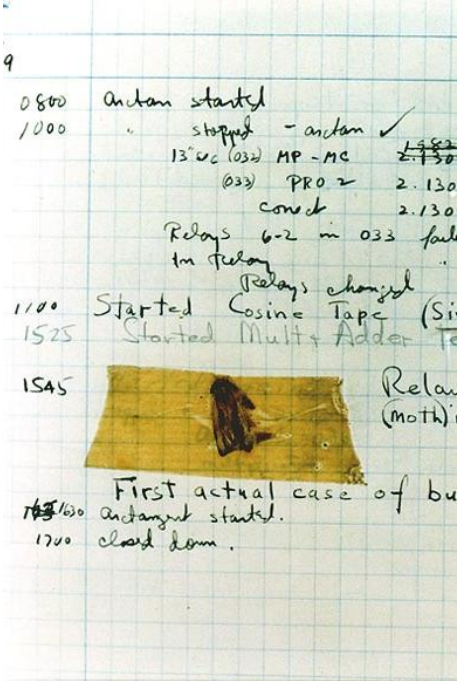
Classic bug mitigation techniques don't work on (public) blockchains:



Bugs

Classic bug mitigation techniques don't work on (public) blockchains:

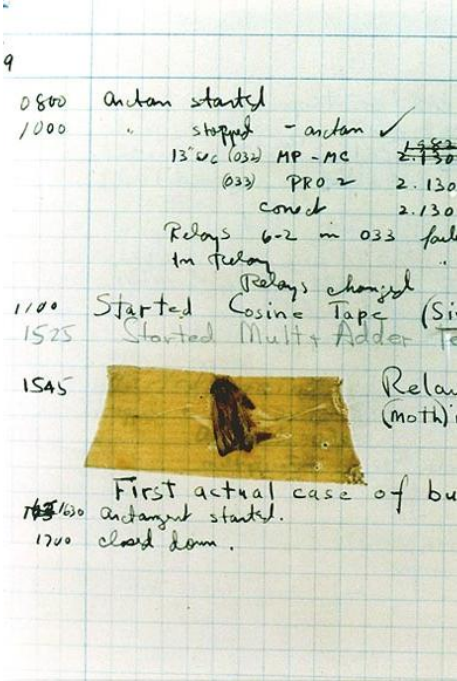
- updating the program



Bugs

Classic bug mitigation techniques don't work on (public) blockchains:

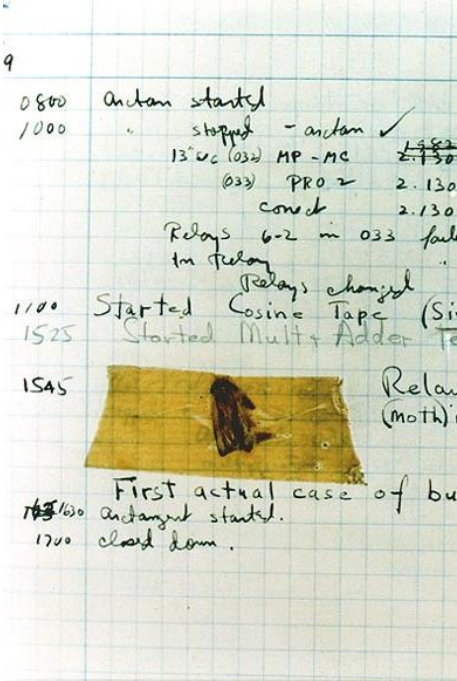
- updating the program
- rolling back the database



Bugs

Classic bug mitigation techniques don't work on (public) blockchains:

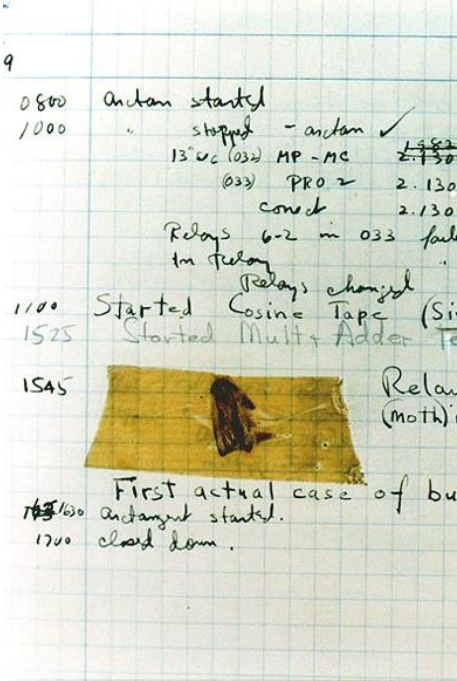
- updating the program
- rolling back the database
- putting the system in lockdown



Bugs

Classic bug mitigation techniques don't work on (public) blockchains:

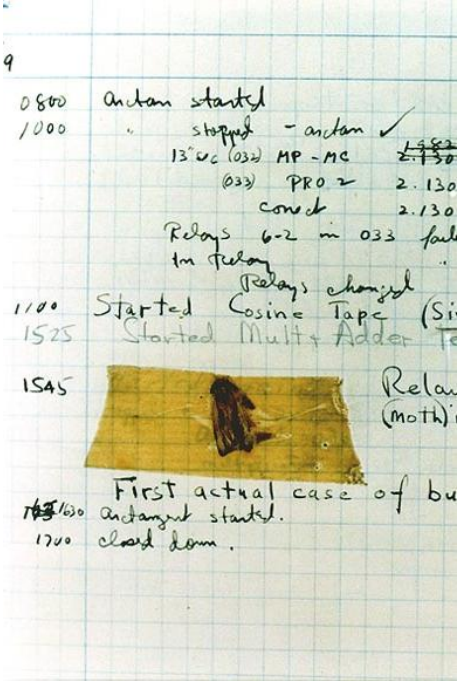
- updating the program
- rolling back the database
- putting the system in lockdown
- sending apology letters



Bugs

Classic bug mitigation techniques don't work on (public) blockchains:

- updating the program
- rolling back the database
- putting the system in lockdown
- sending apology letters
- …



Bugs happen

The DAO (2016)

- *Decentralized Autonomous Organization*
- kind of a digital VC
- idea: investors vote on proposals to spend money
- reentrancy bug exploited to “steal” $ 50 M

Bugs happen

Parity Wallet (2017)

- online wallet for Ether and other tokens
- bug exploited to “steal” $ 30 M

Bugs happen

Parity Wallet (2017)

- online wallet for Ether and other tokens
- bug exploited to “steal” $ 30 M
- another bug led to complete loss of $ 360 M

Ethereum's smart contracts are full of holes

Blockchain-powered computer programs promise to revolutionize the digital economy, but new research suggests they're far from secure.

by **Mike Orcutt** Mar 1, 2018

Computer programs that run on blockchains are shaking up the financial system. But much of the hype around what are called smart contracts is just that. It's a brand-new field. Technologists are just beginning to figure out how to design them so they can be relied on not to lose people's money, and—as a new survey of Ethereum smart contracts illustrates—security researchers are only now coming to terms with what a smart-contract vulnerability even looks like.

Preventing bugs

- bugs may have catastrophic effects
- we can't mitigate bugs

Preventing bugs

- bugs may have catastrophic effects
- we can't mitigate bugs
- we need to prevent bugs from happening

A new problem?

Finance is already strongly dependent on algorithms.

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.

Errant trades from the Knight Capital Group began hitting the New York Stock Exchange almost as soon as the opening bell rang on Wednesday. Brendan McDermid/Reuters

1 of 4

4:01 p.m. | Updated

$10 million a minute.

That's about how much the trading problem that set off turmoil on the stock market on Wednesday morning is already costing the trading firm.

The Knight Capital Group announced on Thursday that it lost $440 million when it sold all the stocks it accidentally bought Wednesday morning because a computer glitch.

'My savings are missing': technical glitch reduces Barclays customers' cash to zero

share

Save 7

One reader's Isa – worth several thousand pounds – was shown as £0 on the Barclays banking app CREDIT: AFP

Follow

By Dominic Webb

21 FEBRUARY 2019 • 7:06PM

Barclays' online banking customers are reporting that their savings accounts are being displayed as having a balance of zero.

Software Engineering

Development workflow

Development workflow

1. develop contract with Solidity

Development workflow

1. develop contract with Solidity
2. deploy with Remix/Metamask (or other tools)

Development workflow

1. develop contract with Solidity
2. deploy with Remix/Metamask (or other tools)
3. ???

Testing

Truffle: Framework covering the entire development lifecycle

- testing (Solidity and JavaScript)
- deployment
- migrations

Testing

Truffle: Framework covering the entire development lifecycle

- testing (Solidity and JavaScript)
- deployment
- migrations

The end.

Testing

Truffle: Framework covering the entire development lifecycle

- testing (Solidity and JavaScript)
- deployment
- migrations

The end?

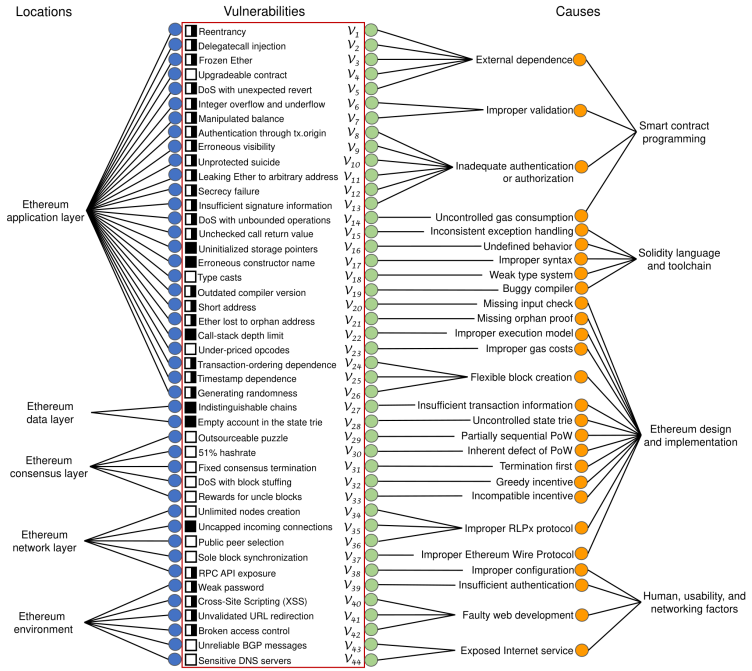
Testing is not enough

> *Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence.*

Edsger W. Dijkstra

How can we test for vulnerabilities?

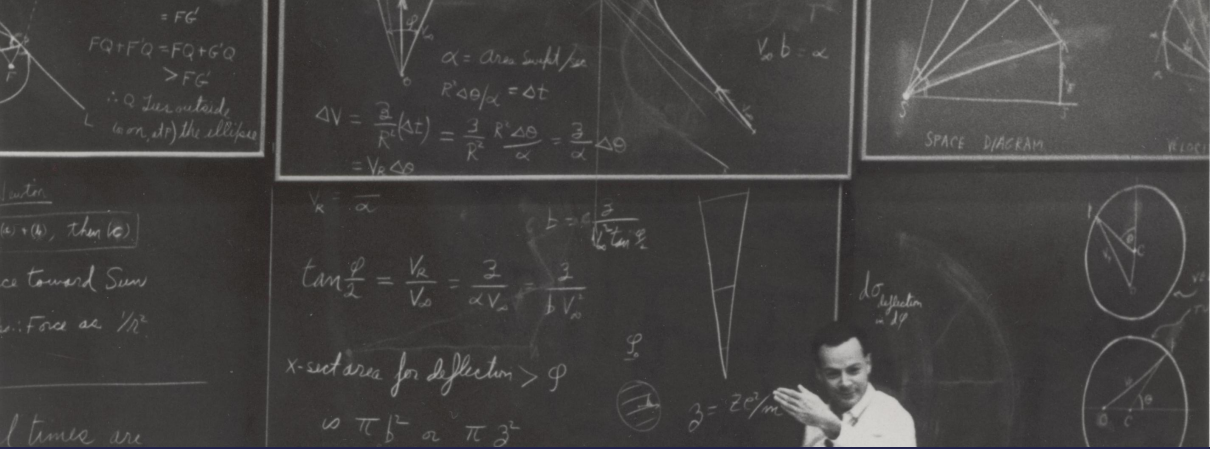
What even are our vulnerabilities?


Locations
Vulnerabilities
Causes
Ethereum application layer
Ethereum data layer
Ethereum consensus layer
Ethereum network layer
Ethereum environment
Reentrancy V_1
Delegatecall injection V_2
Frozen Ether V_3
Upgradeable contract V_4
DoS with unexpected revert V_5
Integer overflow and underflow V_6
Manipulated balance V_7
Authentication through tx.origin V_8
Erroneous visibility V_9
Unprotected suicide V_{10}
Leaking Ether to arbitrary address V_{11}
Secrecy failure V_{12}
Insufficient signature information V_{13}
DoS with unbounded operations V_{14}
Unchecked call return value V_{15}
Uninitialized storage pointers V_{16}
Erroneous constructor name V_{17}
Type casts V_{18}
Outdated compiler version V_{19}
Short address V_{20}
Ether lost to orphan address V_{21}
Call-stack depth limit V_{22}
Under-priced opcodes V_{23}
Transaction-ordering dependence V_{24}
Timestamp dependence V_{25}
Generating randomness V_{26}
Indistinguishable chains V_{27}
Empty account in the state trie V_{28}
Outsourceable puzzle V_{29}
51% hashrate V_{30}
Fixed consensus termination V_{31}
DoS with block stuffing V_{32}
Rewards for uncle blocks V_{33}
Unlimited nodes creation V_{34}
Uncapped incoming connections V_{35}
Public peer selection V_{36}
Sole block synchronization V_{37}
RPC API exposure V_{38}
Weak password V_{39}
Cross-Site Scripting (XSS) V_{40}
Unvalidated URL redirection V_{41}
Broken access control V_{42}
Unreliable BGP messages V_{43}
Sensitive DNS servers V_{44}
External dependence
Improper validation
Inadequate authentication or authorization
Uncontrolled gas consumption
Inconsistent exception handling
Undefined behavior
Improper syntax
Weak type system
Buggy compiler
Missing input check
Missing orphan proof
Improper execution model
Improper gas costs
Flexible block creation
Insufficient transaction information
Uncontrolled state trie
Partially sequential PoW
Inherent defect of PoW
Termination first
Greedy incentive
Incompatible incentive
Improper RLPx protocol
Improper Ethereum Wire Protocol
Improper configuration
Insufficient authentication
Faulty web development
Exposed Internet service
Smart contract programming
Solidity language and toolchain
Ethereum design and implementation
Human, usability, and networking factors

Vulnerability causes

Cause	total	eliminated	avoidable	open
Programming	14	0	13	1
Solidity	5	2	2	1
Ethereum	18	4	5	9
Human/usability	7	0	5	2
total	44	6	25	13

Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu: “A Survey on Ethereum Systems Security: Vulnerabilities, Attacks and Defenses” (2019)



Formal Methods

Formal Methods?

> *'Formal Methods' refers to mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems.*

César Muñoz (NASA)

Code contracts

```
@Requires("x >= 0")
@Ensures("result >= 0")
static double sqrt(double x);
```

What is verification?

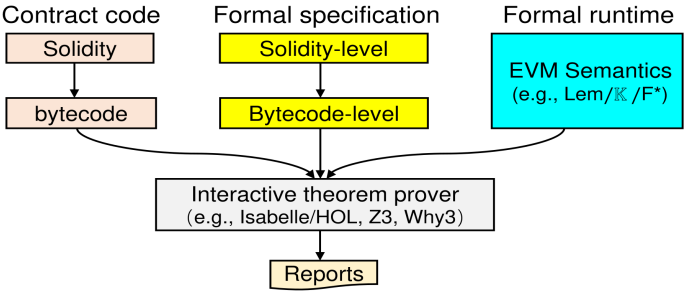
Verification consists of three components:

1. formal language for specification
2. formal semantics for implementation
3. static analysis

```
subsection "Preservation of semantics"

lemma ccomp_bigstep:
  "(c,s) ⇒ t ⟹ ccomp c ⊢ (0,s,stk) →* (size(ccomp c),t,stk)"
proof(induction arbitrary: stk rule: big_step_induct)
  case (Assign x a s)
  show ?case by (fastforce simp:fun_upd_def cong: if_cong)
next
  case (Seq c1 s1 s2 c2 s3)
  let ?cc1 = "ccomp c1"  let ?cc2 = "ccomp c2"
  have "?cc1 @ ?cc2 ⊢ (0,s1,stk) →* (size ?cc1,s2,stk)"
    using Seq.IH(1) by fastforce
  moreover
  have "?cc1 @ ?cc2 ⊢ (size ?cc1,s2,stk) →* (size(?cc1 @ ?cc2),s3,stk)"
    using Seq.IH(2) by fastforce
  ultimately show ?case by simp (blast intro: star_trans)
next
  case (WhileTrue b s1 c s2 s3)
  let ?cc = "ccomp c"
  let ?cb = "bcomp b False (size ?cc + 1)"
  let ?cw = "ccomp(WHILE b DO c)"
```

Verification in Ethereum



Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu: "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks and Defenses" (2019)

Full stack verification

Component	Availability
Specification of EVM	✔
Specification of Solidity	?
Verified Solidity compiler	✖
Verified EVM runtime	✔
Solidity VCG	✖
Formal business case	✖

Verification works

> *“The HackerGold (HKG) token is an ERC20 token written in Solidity [...] We found that the [...] implementation does not conform to the ERC20 standard.”*

Formally Verified Smart Contracts by Runtime Verification (2017–2019)

Verification works

> *The Vyper ERC20 token is shipped as part of the official Vyper language compiler distribution.*
> *Vyper, currently being developed by the Ethereum Foundation, is an experimental smart contract language with syntax similar to Python, designed with the goal of being a simpler and more secure alternative to Solidity.*
> *The Vyper ERC20 token was successfully verified against the ERC20-EVM specification, implying its full conformance to the ERC20 standard.*

Formally Verified Smart Contracts by Runtime Verification (2017–2019)

Example

```
Theorem can_claim_back id b d st bc:
  (* (a) The backer b has donated d, so the contract holds
         that record in its state *)
  donated b d st →
  (* (b) The campaign has not been funded. *)
  ¬ funded (state st) →
  (* (c) Balance is small: not reached the goal. *)
  balance st < (get_goal (state st)) →
  (* (d) Block number exceeds the deadline. *)
  get_max_block (state st) < block_num bc →
  (* (conclusion) Backer b can get their donation back. *)
  ∃ (m : message),
    sender m == b ∧
    out (step_prot c st bc m) = Some (Msg d id b 0 ok_msg).
```

Figure 5. A backer can claim back her funds if the campaign fails.

Ilya Sergey, Amrit Kumar, Aquinas Hobor: "Scilla: a Smart Contract Intermediate-Level LAnguage" (2018)

Research

- *Workshop on Trusted Smart Contracts* (2017–)
- *Workshop Formal Methods for Blockchains* (2019–)

Legal Force & Effect

Code is Law

> *However, these more complex agreements, with greater engagement with real world goods and services, highlight the necessity of effective dispute resolution, as well as indicate a necessary interrelation with territorial legal systems.*

Code is Law

> *However, these more complex agreements, with greater engagement with real world goods and services, highlight the necessity of effective dispute resolution, as well as indicate a necessary interrelation with territorial legal systems.*
> *Contestation mechanisms are necessary to 'soften' the effects of self-executing 'smart contracts', and make transactions reversible, allowing the outcomes of dispute resolution to be enforced.*

Jake Goldenfein, Andrea Leiter: "Legal Engineering on the Blockchain: 'Smart Contracts' as Legal Conduct" (2018)

Legalese

- “Legalese” is (English) writing that is inscrutable to laypeople
- Are smart contracts an improvement?
- Does verification help?

Smart Contracts?

> *There are only two things you need to know about smart contracts:*
>
> 1. *They're not smart.*
> 2. *They're not contracts.*

Philip Wadler

INNOQ

Navigation

Die Blockchain

Eine Kette der Möglichkeiten

Q & A

INNOQ
www.innoq.com

Lars Hupel

lars.hupel@innoq.com

@larsr_h

LARS HUPEL

Senior Consultant
innoQ Deutschland GmbH

Lars is known as one of the founders of the Type-level initiative which is dedicated to providing principled, type-driven Scala libraries in a friendly, welcoming environment. A frequent conference speaker, they are active in the open source community, particularly in Scala.

Sources

- Treaty of Rome: https://en.wikipedia.org/wiki/File:Treaty_of_Rome.jpg, author unknown
- Lightning: https://pixabay.com/photos/lightning-storm-weather-sky-399853/
- Lady Justice: https://pixabay.com/photos/justice-statue-lady-justice-2060093/
- Feynman with blackboard: https://commons.wikimedia.org/wiki/File:HD.3A.053_(10481714045).jpg
- Interchange: https://unsplash.com/photos/nDfEFYiGrAY
- Mike Orcutt: "Ethereum's smart contracts are full of holes", https://www.technologyreview.com/s/610392/ethereums-smart-contracts-are-full-of-holes/
- Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu: "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks and Defenses", https://arxiv.org/pdf/1908.04507.pdf
- Jake Goldenfein, Andrea Leiter: "Legal Engineering on the Blockchain: 'Smart Contracts' as Legal Conduct", https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3176363
- Nathaniel Popper: "Knight Capital Says Trading Glitch Cost It $440 Million", https://dealbook.nytimes.com/2012/08/02/knight-capital-says-trading-mishap-cost-it-440-million/ (2012)
- Domic Webb: "'My savings are missing': technical glitch reduces Barclays customers' cash to zero", https://www.telegraph.co.uk/personal-banking/savings/savings-missing-technical-glitch-reduces-barclays-customers/ (2019)
- Ilya Sergey, Amrit Kumar, Aquinas Hobor: "Scilla: a Smart Contract Intermediate-Level LAnguage", https://arxiv.org/pdf/1801.00687.pdf
- Solidity Semantics: https://github.com/kframework/solidity-semantics