Verification Tools in Practice

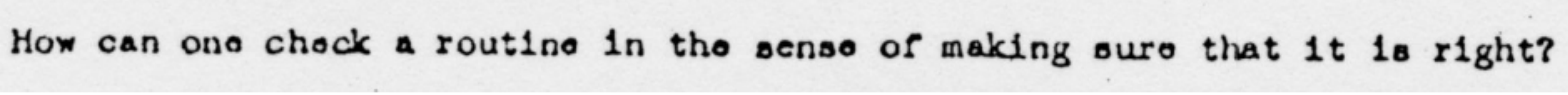
Testing, Quality Assurance, and Maintenance
Winter 2018

Prof. Arie Gurfinkel

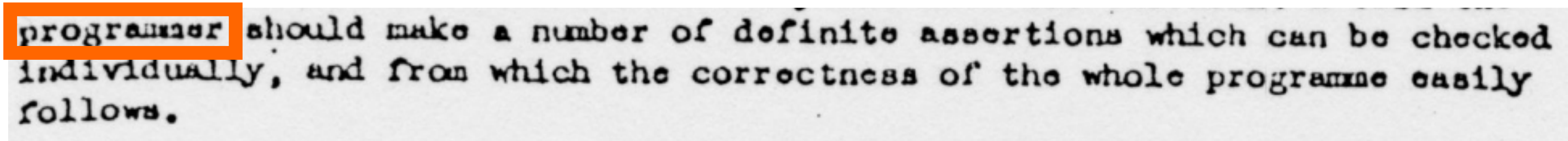
UNIVERSITY OF
WATERLOO

Turing, 1949

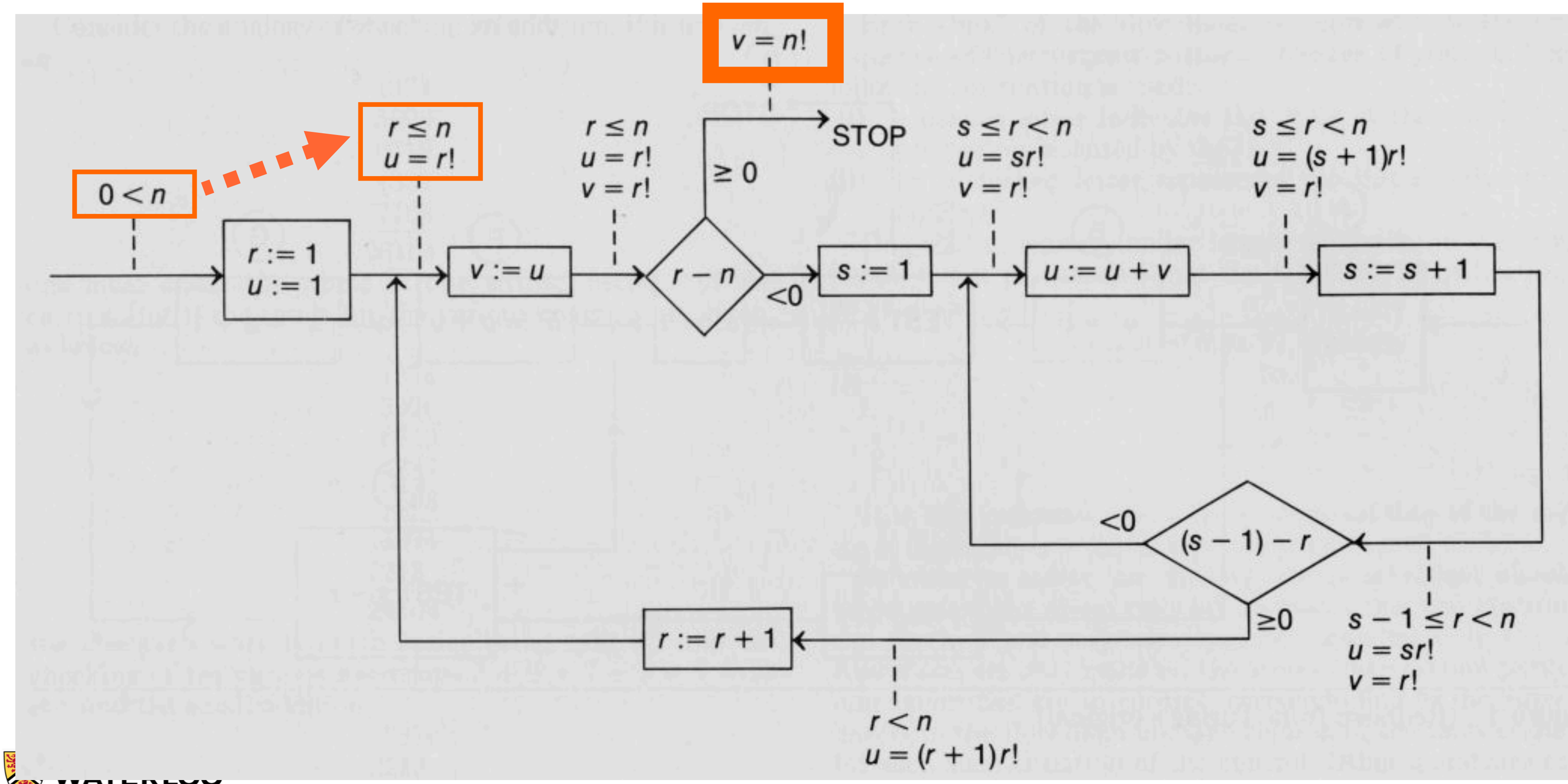
Alan M. Turing. “Checking a large routine”, 1949



How can one check a routine in the sense of making sure that it is right?



programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows.



VerifyThis! Verification Competition

http://www.pm.inf.ethz.ch/research/verifythis.html

“Hackathon”-format

Given a few days and a few problems, create “provably” correct solutions

Solutions are judges by a committee to decide that they meet “provability” criteria.

Dafny is the most commonly used tools in the competition

Great way to learn about other tools, current research, and challenging problems

Microsoft Visual Studio Products

Code Contracts

- https://marketplace.visualstudio.com/items?itemName=RiSEResearchinSoftwareEngineering.CodeContractsforNET
- https://github.com/Microsoft/CodeContracts

- statically and dynamically checked method pre- and post-conditions

IntelliTest

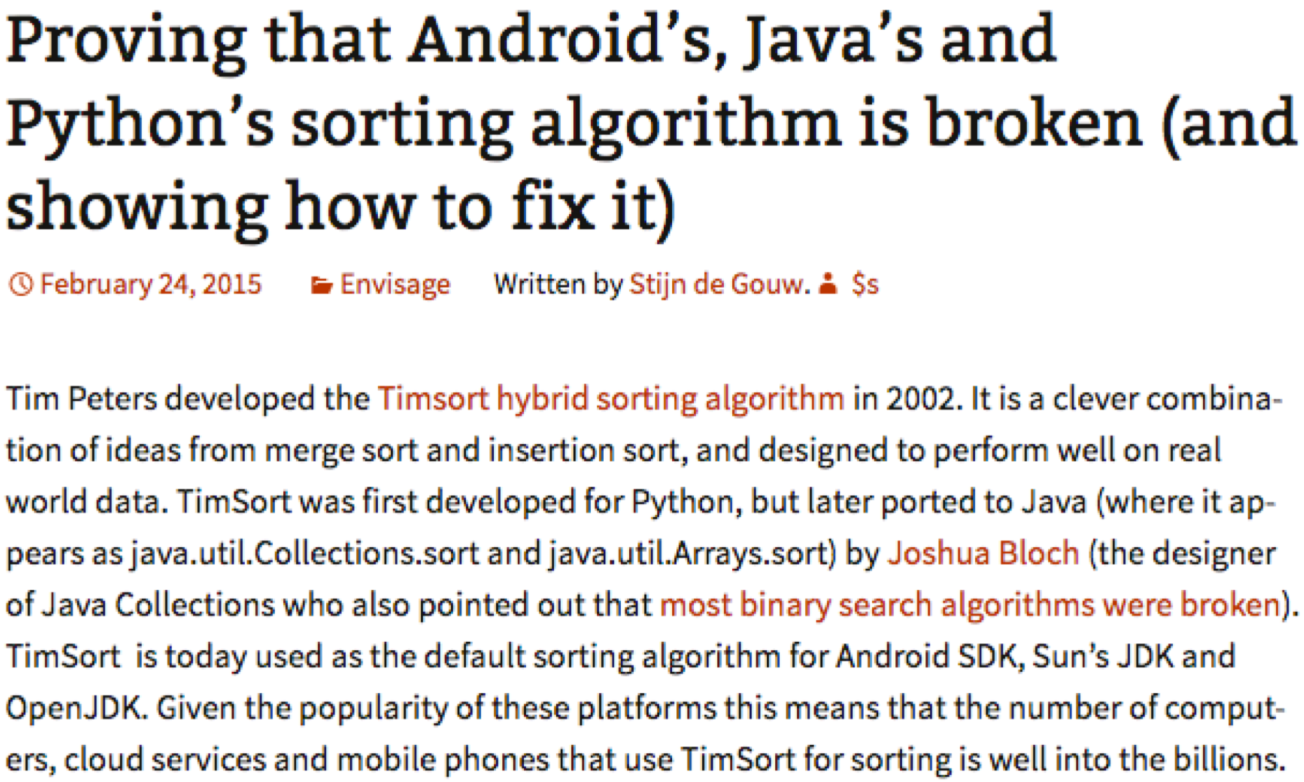
- https://www.visualstudio.com/en-us/docs/test/developer-testing/intellitest-manual/introduction

- automated test generation by dynamic symbolic execution

Auto-active Verification Tools

Formal verification of code targeting developers

- Why3: http://why3.lri.fr
  - Functional PL approach: WhyML, VCGen, supports many SMT solvers
- VeriFast: https://github.com/verifast/verifast
  - concurrent and low-level memory, works directly on C language
- OpenJML: http://www.openjml.org/
  - standard for annotations for Java
  - based on ESC-Java by Rustan Leino (precursor of Dafny)
- KeY: https://www.key-project.org/
  - Java, based on Symbolic Execution, found bugs in TimSort
- Frama-C: https://frama-c.com/
  - static analysis and auto-active verification for C
  - standard for annotations
  - Automation with static analysis, verification with Why3



Proving that Android's, Java's and Python's sorting algorithm is broken (and showing how to fix it)

February 24, 2015   Envisage   Written by Stijn de Gouw. $s

Tim Peters developed the Timsort hybrid sorting algorithm in 2002. It is a clever combination of ideas from merge sort and insertion sort, and designed to perform well on real world data. TimSort was first developed for Python, but later ported to Java (where it appears as java.util.Collections.sort and java.util.Arrays.sort) by Joshua Bloch (the designer of Java Collections who also pointed out that most binary search algorithms were broken). TimSort is today used as the default sorting algorithm for Android SDK, Sun's JDK and OpenJDK. Given the popularity of these platforms this means that the number of computers, cloud services and mobile phones that use TimSort for sorting is well into the billions.

http://envisage-project.eu/proving-android-java-and-python-sorting-algorithm-is-broken-and-how-to-fix-it/

General Purpose Theorem Provers

Logician view of program verification (and not just program)

ACL2: http://www.cs.utexas.edu/users/moore/acl2/

PVS: http://pvs.csl.sri.com/

HOL: https://hol-theorem-prover.org/

HOL-light: http://www.cl.cam.ac.uk/~jrh13/hol-light/

Isabelle: http://isabelle.in.tum.de/

Coq: https://coq.inria.fr/

- https://softwarefoundations.cis.upenn.edu/plf-current/Hoare.html
- Programs are extracted from proofs of theorems

Lean: https://leanprover.github.io/

Intermediate Languages for Verification

Build your own auto-active verifier for your favorite language (python, etherium, javascript, …)

BoogiePL: https://github.com/boogie-org/boogie

- intermediate language of Dafny
- “similar” to our while language

WhyML

- Intermediate language of Why3

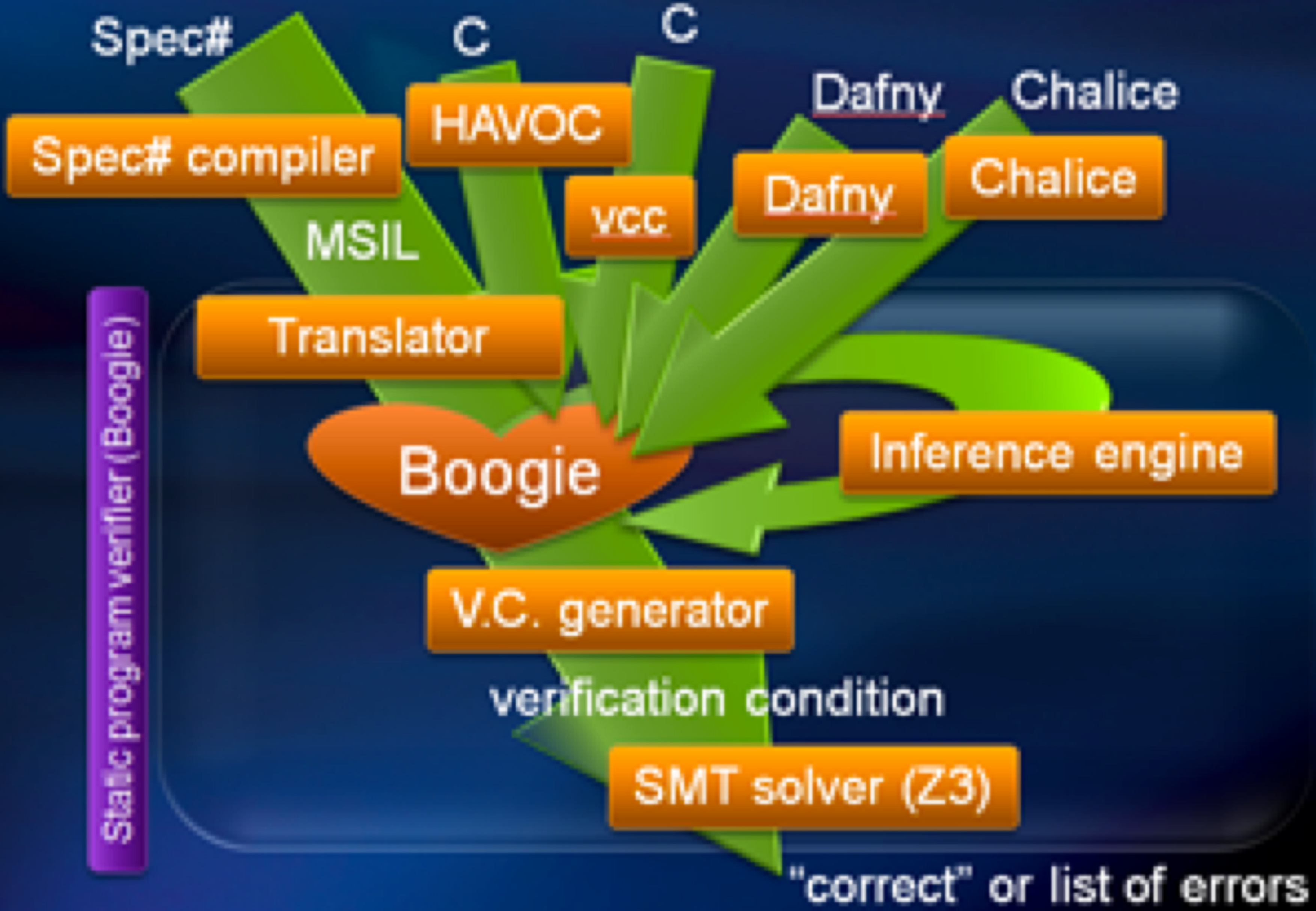
Viper

- http://www.pm.inf.ethz.ch/research/viper.html
- emphasis on concurrency and memory (inhale/exhale/implicity dyn. frames)

Constrained Horn Clauses

- a fragment of FOL sufficient for invariants
- used by some automatic tools

Boogie verifier architecture



SPARKPro

http://www.adacore.com/sparkpro/

Ada is an old language still used in security- and safety-critical domains

SPARKPro provides auto-active verification for Ada

Actively used by specialized companies

IronClad and InronFleet

https://github.com/Microsoft/Ironclad

Distributed state machine and distributed key-value dictionary implemented in Dafny

Verified key correctness properties and key liveness properties

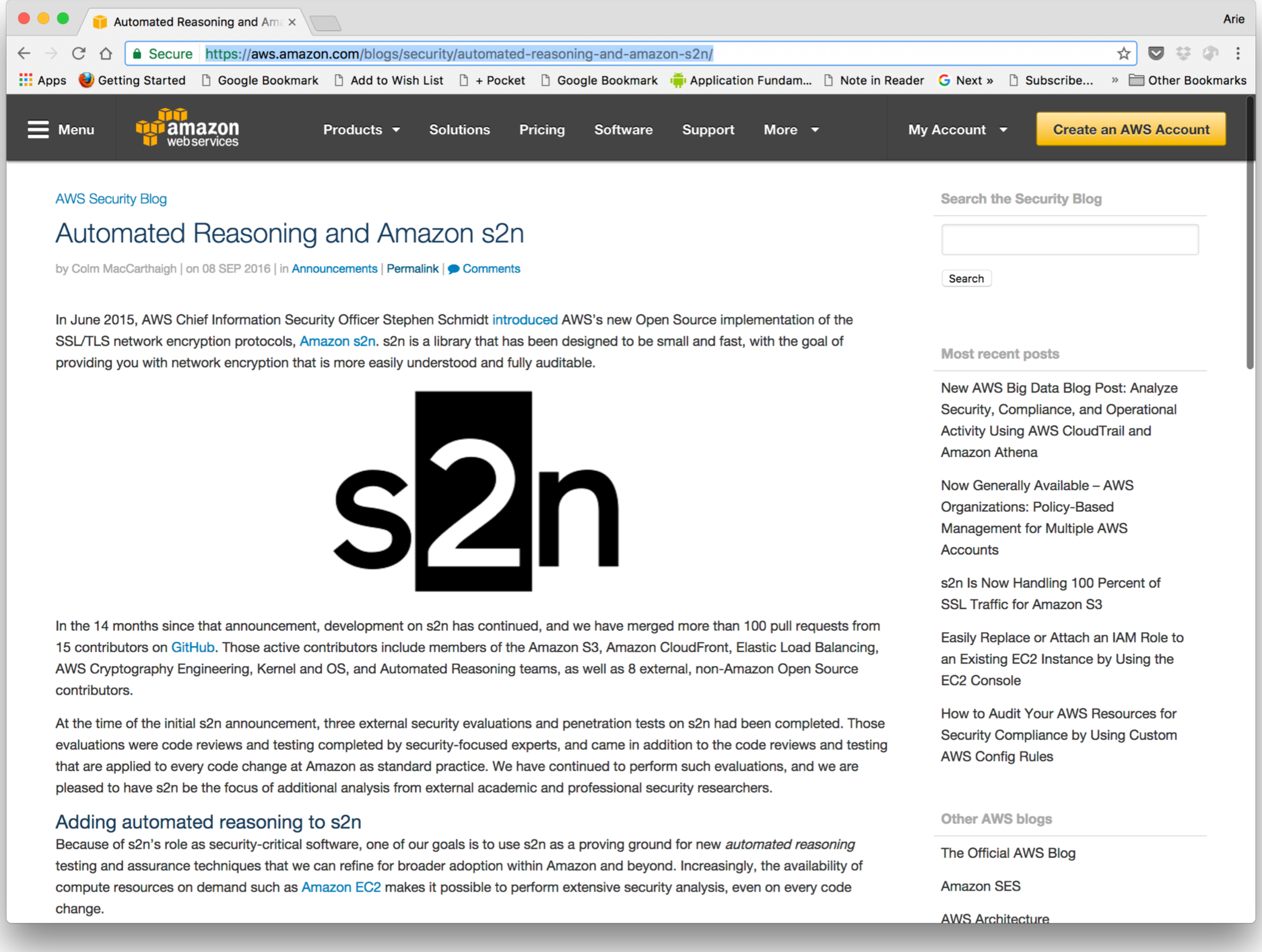
- e.g., distributed algorithm eventually converges to a result as long as enough participants are active

Open sourced and available on GitHub

Non-trivial system designed and verified by system engineers

- not researchers in FM trying to show off what their tools can do

Amazon S2N



AWS Security Blog

Automated Reasoning and Amazon s2n

by Colm MacCarthaigh | on 08 SEP 2016 | in Announcements | Permalink | Comments

In June 2015, AWS Chief Information Security Officer Stephen Schmidt introduced AWS's new Open Source implementation of the SSL/TLS network encryption protocols, Amazon s2n. s2n is a library that has been designed to be small and fast, with the goal of providing you with network encryption that is more easily understood and fully auditable.

In the 14 months since that announcement, development on s2n has continued, and we have merged more than 100 pull requests from 15 contributors on GitHub. Those active contributors include members of the Amazon S3, Amazon CloudFront, Elastic Load Balancing, AWS Cryptography Engineering, Kernel and OS, and Automated Reasoning teams, as well as 8 external, non-Amazon Open Source contributors.

At the time of the initial s2n announcement, three external security evaluations and penetration tests on s2n had been completed. Those evaluations were code reviews and testing completed by security-focused experts, and came in addition to the code reviews and testing that are applied to every code change at Amazon as standard practice. We have continued to perform such evaluations, and we are pleased to have s2n be the focus of additional analysis from external academic and professional security researchers.

Adding automated reasoning to s2n

Because of s2n's role as security-critical software, one of our goals is to use s2n as a proving ground for new *automated reasoning* testing and assurance techniques that we can refine for broader adoption within Amazon and beyond. Increasingly, the availability of compute resources on demand such as Amazon EC2 makes it possible to perform extensive security analysis, even on every code change.

Search the Security Blog

Search

Most recent posts

- New AWS Big Data Blog Post: Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena
- Now Generally Available – AWS Organizations: Policy-Based Management for Multiple AWS Accounts
- s2n Is Now Handling 100 Percent of SSL Traffic for Amazon S3
- Easily Replace or Attach an IAM Role to an Existing EC2 Instance by Using the EC2 Console
- How to Audit Your AWS Resources for Security Compliance by Using Custom AWS Config Rules

Other AWS blogs

- The Official AWS Blog
- Amazon SES
- AWS Architecture

https://aws.amazon.com/blogs/security/automated-reasoning-and-amazon-s2n/

UNIVERSITY OF WATERLOO

Symbolic Execution Tools

KLEE: https://klee.github.io/

- EXE-style symbolic execution for LLVM virtual machine

S2E: http://s2e.systems/

- EXE-style symbolic execution at Virtual Machine Level

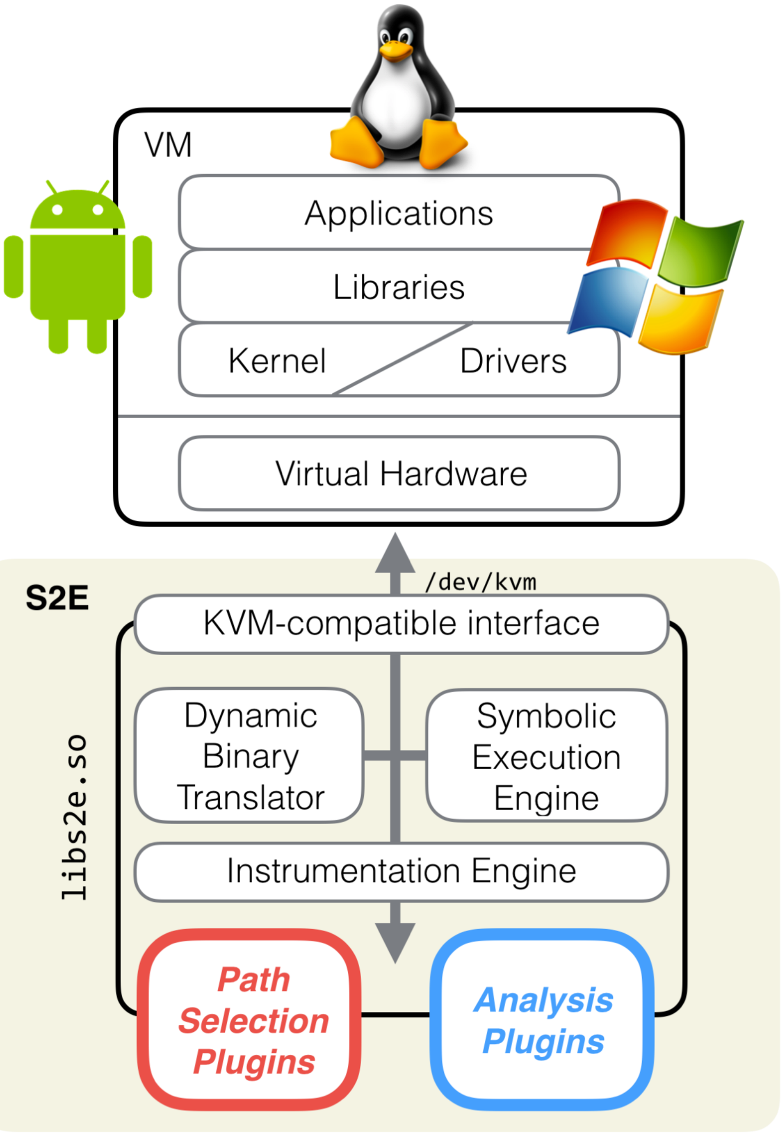
Angr: http://angr.io/

- A framework for symbolic execution and analysis of binaries

Mayhem: https://forallsecure.com/blog/2016/02/09/unleashing-mayhem/

- Automated exploit generation using symbolic execution

S2E: In-vivo Analysis of Software Systems



Automatic Verification

Automatically verify specific properties of programs

- minimize user guidance
- fixed, not very deep properties

Microsoft Static Driver Verifier

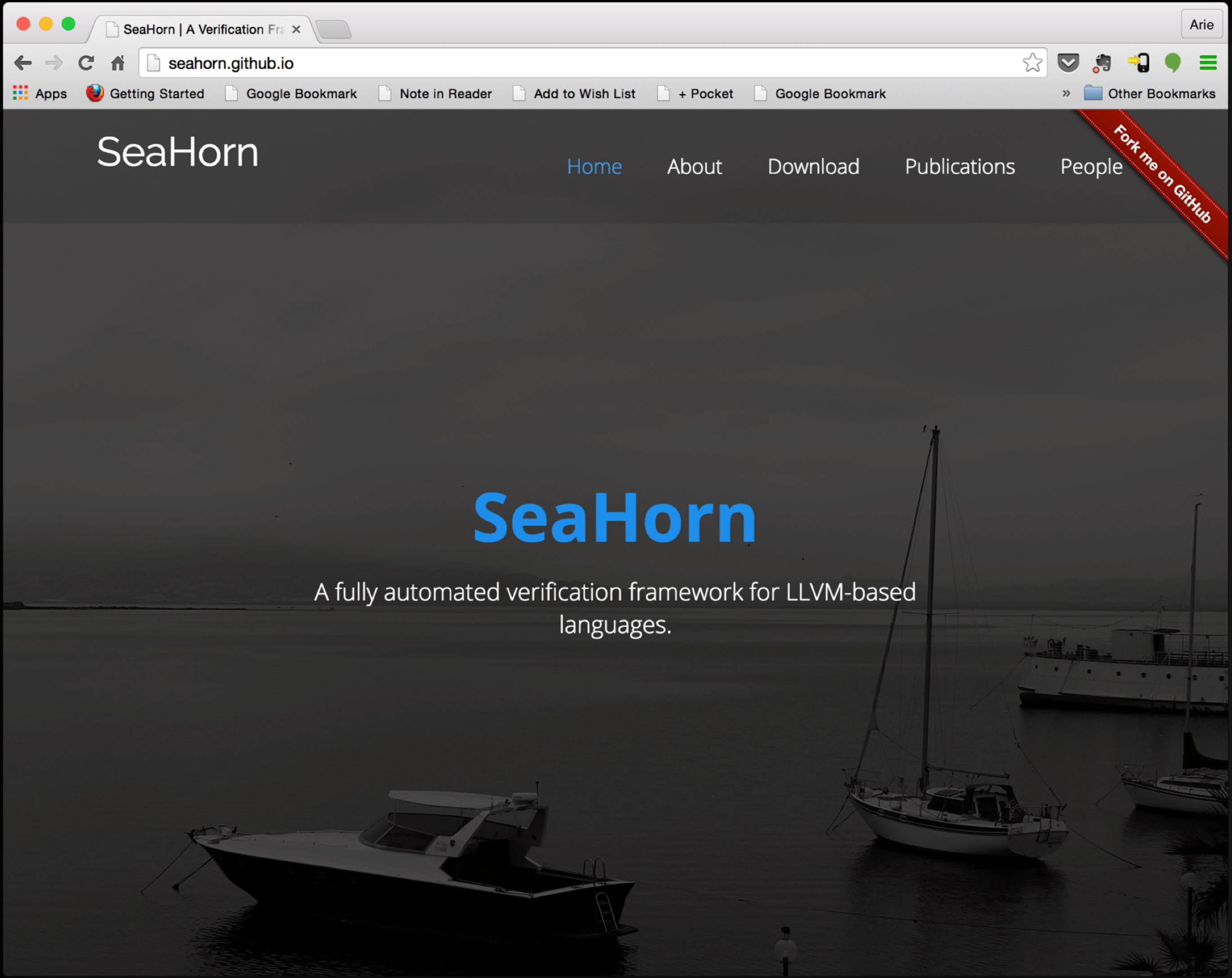
- https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/static-driver-verifier
- based on Model Checking
- checks for correct API usage rules: e.g., always lock-before-unlock

Linux Driver Verification: https://linuxtesting.org/ldv

- Implementation of MS SDV for Linux using OSS tools

Facebook Infer: http://fbinfer.com/

- Automatically prove correct memory handling (e.g., absence of null dereferencing) in C and Java programs



SeaHorn | A Verification Fra

seahorn.github.io

Apps · Getting Started · Google Bookmark · Note in Reader · Add to Wish List · + Pocket · Google Bookmark · Other Bookmarks

SeaHorn

Home · About · Download · Publications · People

Fork me on GitHub

SeaHorn

A fully automated verification framework for LLVM-based languages.

http://seahorn.github.io

Smart Contracts: New Domain for Verification

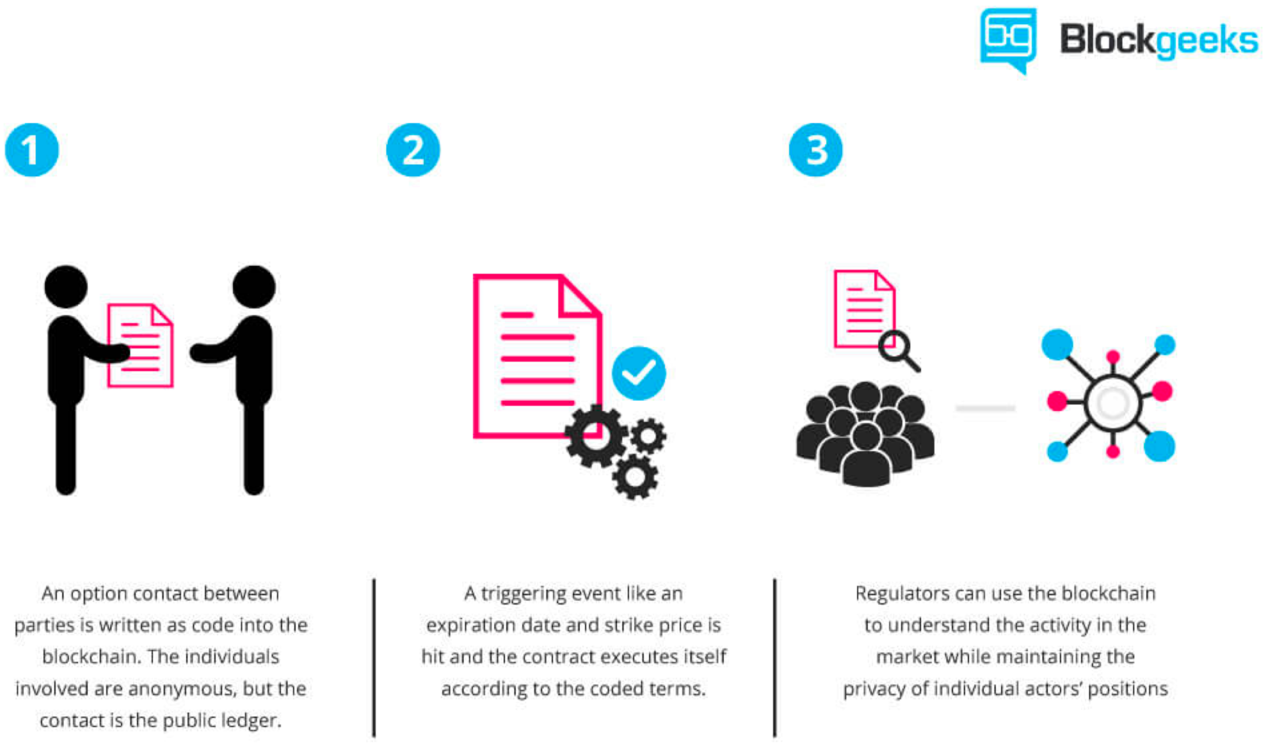
Use code to represent a contract

Contract is executed by code

Blockchain is used to ensure adherence to the contract in a distributed decentralized environment

What does a contract actually do?

https://blockgeeks.com/guides/smart-contracts/

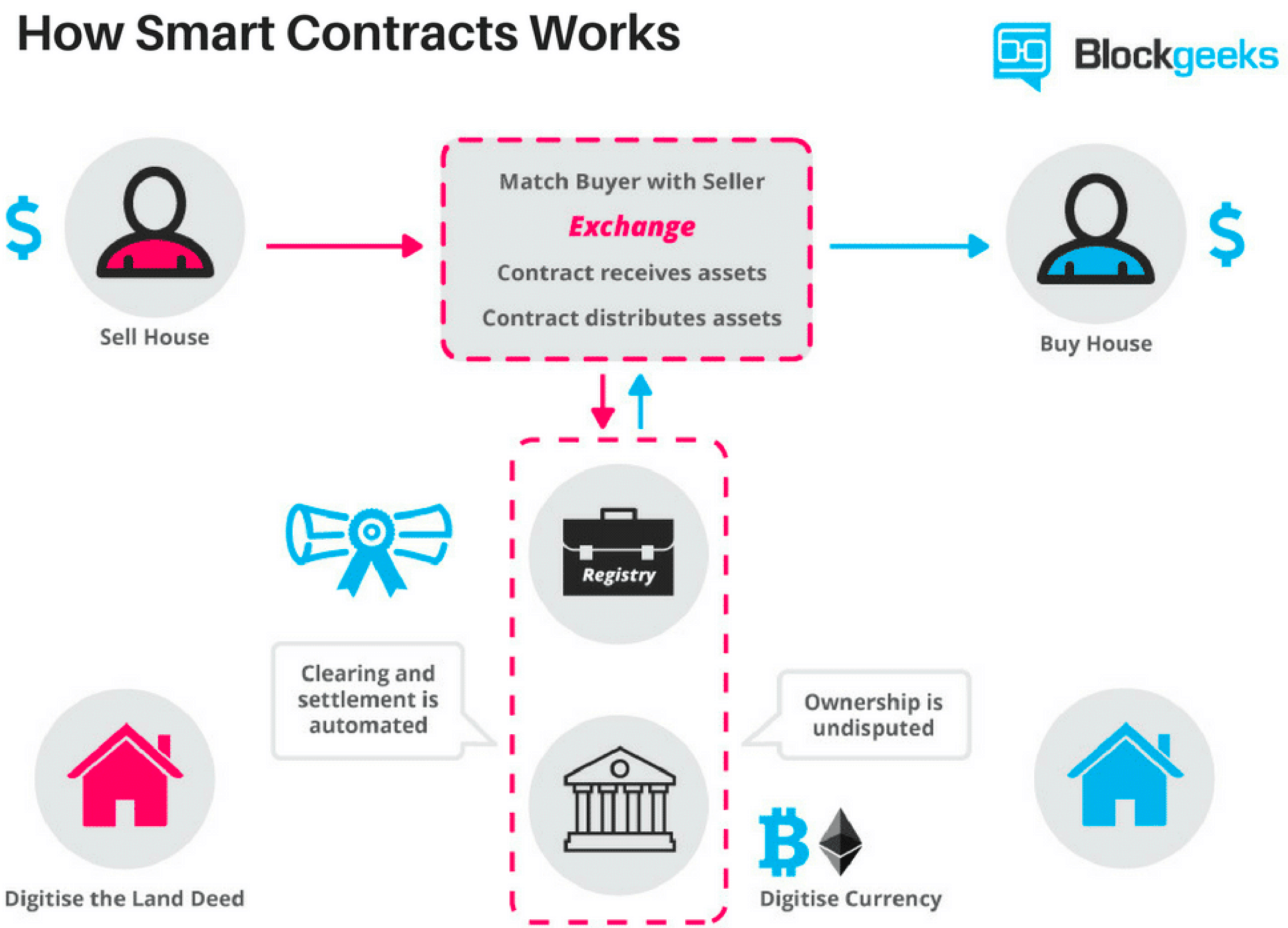


1. An option contact between parties is written as code into the blockchain. The individuals involved are anonymous, but the contact is the public ledger.

2. A triggering event like an expiration date and strike price is hit and the contract executes itself according to the coded terms.

3. Regulators can use the blockchain to understand the activity in the market while maintaining the privacy of individual actors' positions

https://blockgeeks.com/guides/smart-contracts/

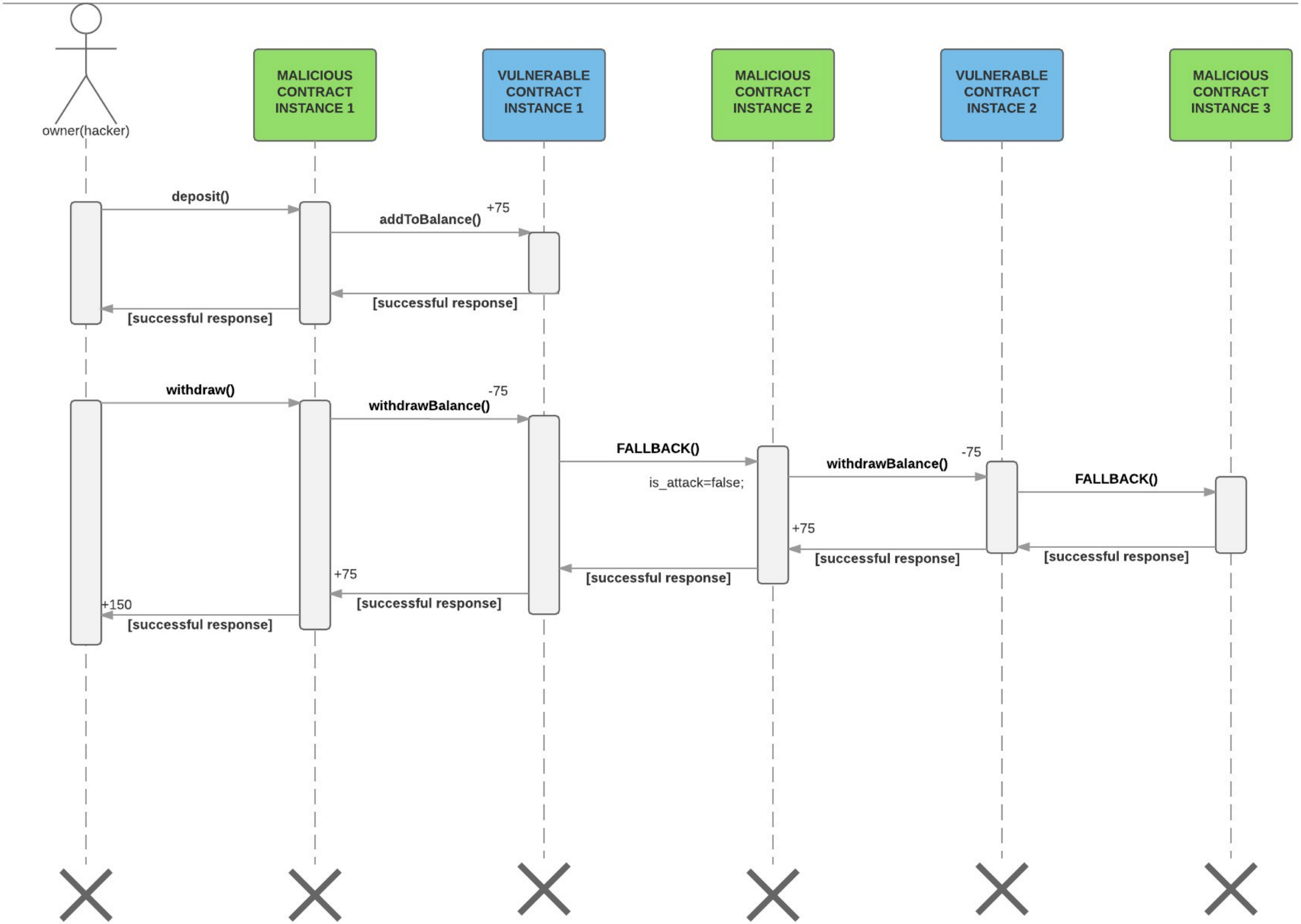


https://blockgeeks.com/guides/smart-contracts/

The DAO Attack

https://medium.com/@MyPaoG/explaining-the-dao-exploit-for-beginners-in-solidity-80ee84f0d470

Follow the rules, use the contract, withdraw all money, get rich in the process


owner(hacker)
MALICIOUS CONTRACT INSTANCE 1
VULNERABLE CONTRACT INSTANCE 1
MALICIOUS CONTRACT INSTANCE 2
VULNERABLE CONTRACT INSTACE 2
MALICIOUS CONTRACT INSTANCE 3
deposit()
+75
addToBalance()
[successful response]
[successful response]
withdraw()
-75
withdrawBalance()
FALLBACK()
-75
withdrawBalance()
FALLBACK()
is_attack=false;
+75
[successful response]
[successful response]
[successful response]
+75
[successful response]
+150
[successful response]

Symbolic Execution for Smart Contracts

MyThril

- https://github.com/ConsenSys/mythril
- Security analysis tool for Etherium Smart Contracts

Manticore: https://github.com/trailofbits/manticore

- Symbolic execution tool

Oyente: http://www.comp.nus.edu.sg/~loiluu/oyente.html

- An analysis tool for smart contracts
- https://oyente.melon.fund

Very active area of both research and development.

Is Verification Enough

Can verified software fail?

Do we need both testing and verification?