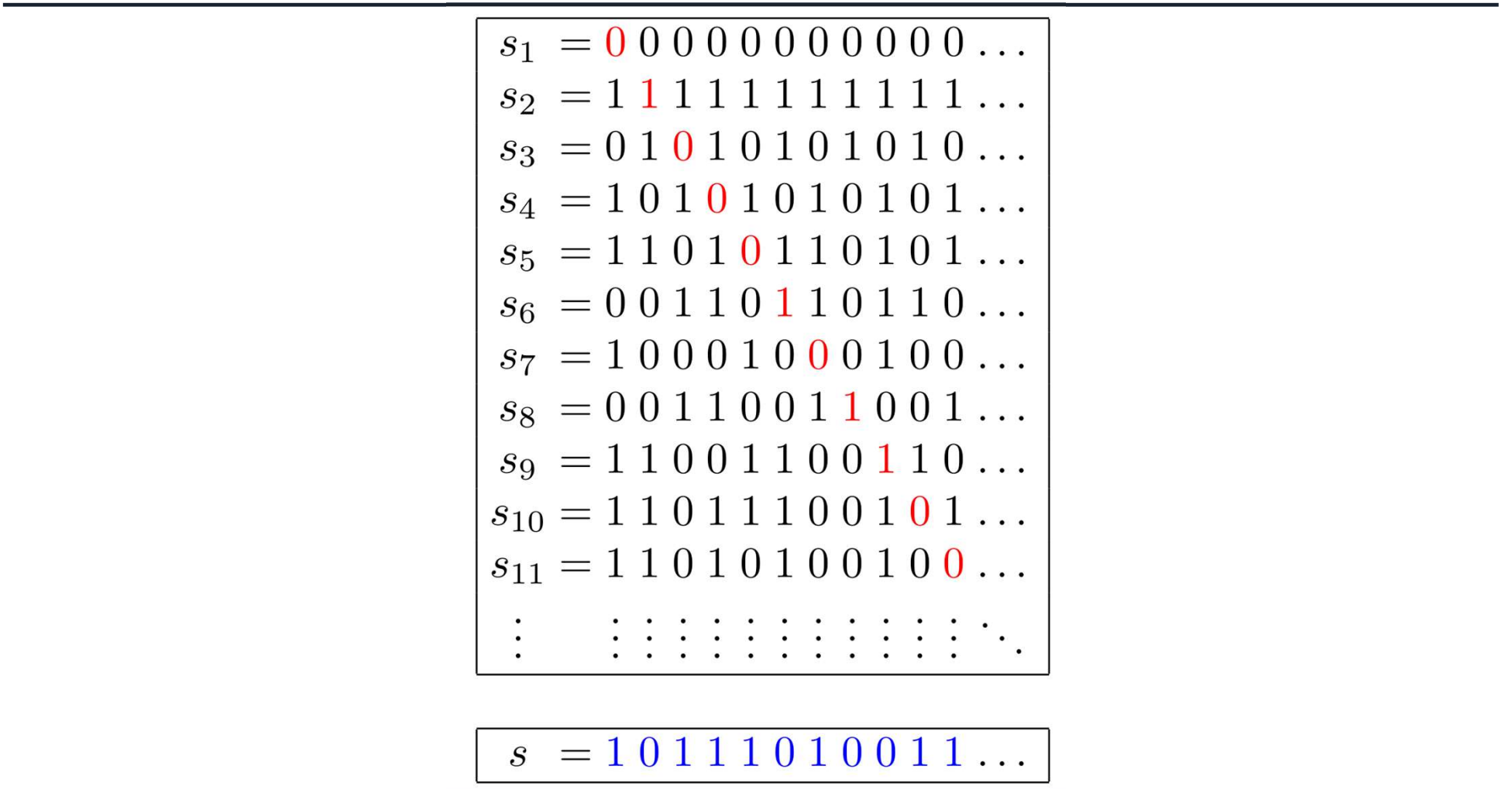
Escaping Cantor's Find-Fix Cycle

Falcon Darkstar Momot
Dartmouth College and Aiven.io

USENIX WOOT, 2025



https://commons.wikimedia.org/wiki/File:Diagonal_argument_01_svg.svg

Vulnerabilities: Apparently Dense

An analysis of how many undiscovered vulnerabilities remain in information systems

Jonathan M. Spring (2023), J. Computers and Security

“there will always be more vulnerabilities in a given piece of software”. 😭

Why? Why?!
Or: Woot!

Anecdotally:

Complexity is the enemy of security (but we love to create it)

What does that mean?

1. A constraint has been instantly violated
   --- Most of CVE probably lives here (WIP) ---
2. Two or more parts interact badly
3. The effects of multiple layers render the result incorrect
4. An external effect on one layer can violate a requirement
   --- Security teams mostly work above this line ---
5. The program does not meet specification
6. The specification is not decidable
   --- LangSec helps above this line ---
7. The specification does not match intent
8. The specifiers were not aware of all their requirements

Example 1

A constraint has been instantly violated.

Memory corruption bugs that are triggered by user input

This class is amenable to SAST detection

Example 2

Two or more parts interact badly

Response splitting attacks
TOCTOU (two independent parts interact in the time domain)

Occasionally can be addressed with SAST

Example 3

The effects of multiple layers render the result incorrect

Type confusion, e.g., the string “False” is rendered as != 0, therefore boolean 1
FPU imprecision renders an output incorrect
X.509 certificates with nulls in identifiers
Rendering over other parts of the viewport

Example 4

An external effect on one layer can violate a requirement

Glitching attacks and related analogue-domain interference

TOCTOU by a different application on the same filesystem

Example 5

The program does not meet specification

Missing authorization checks; “IDOR”

Corrupt documents cause DoS instead of graceful failure

LLMs spilling secrets

- Special case: the specification is structurally impossible to meet

Example 6

The specification is not decidable

Android master key

JSON with duplicate keys

Office document bugs where hashmaps are wrong

Example 7

The specification does not match intent

Authorization policies that do not match organization rules

Misapplication of anomaly detection

Arbitrary hard limits that cause DoS

Example 8

The specifiers were not aware of all of their requirements

Lack of a content security policy in HTTP
No fallback detection in old TLS
Most timing-derived info leaks
ZIP bombs
Rowhammer, &c.

1. A constraint has been instantly violated
   --- Most of CVE probably lives here (WIP) ---
2. Two or more parts interact badly
3. The effects of multiple layers render the result incorrect
4. An external effect on one layer can violate a requirement
   --- Security teams mostly work above this line ---
5. The program does not meet specification
6. The specification is not decidable
   --- LangSec helps above this line ---
7. The specification does not match intent
8. The specifiers were not aware of all their requirements

Software Testing

For all n in non-negative integers, prove that 3 divides n^3+2n

-5: -25-10 = -35; -35/3 = -11.67; false – pass

0: 0+0 = 0; 0/3 = 0; true – pass

1: 1+2 = 3; 3/3 = 1; true – pass

Measuring Security Testing

???

TODO what to say about this

Density of Vulnerabilities

- Software testing and vulns: **non-correlates**!
- LoC coverage and test usefulness: **non-correlates**!
- Test completeness against external predicates: **unknowable**! (ish)

Density of Vulnerabilities In Turing-Complete Code

Any nontrivial property of a Turing machine is undecidable

=> Number of test cases approaches infinity

Ye Olde Automata

Push-down automation (CFG): proof by induction

Any language simpler than (or at) DCF: equivalence to spec

Diagonalizable cases not amenable to induction:
how do you test this?!

Some Strategies

Developers should write static analysis rules

But which ones?

But which tests right now?

Some Strategies

Developers should write static analysis rules

Memory correctness: a ternary conditional

- 2 answers mean no merge

An Open Problem

Say we do lightweight formal methods and SAST...

Compositional properties of models?

Vulnerability Topologies

They exist as n-dimensional solids

- Each dimension represents the possible system interactions of one participant (stakeholder or system)

How large is the n-dimensional space?

Vulnerabilities in Complex Systems

Based on the aforementioned, could be infinite

Test cases are uncountably infinite for sure

Could be multiple vulnerabilities per line

Limiting the Space

Random walks through infinite n-dimensional solids? No thanks.

Constrained interfaces are the way to avoid that walk.

Write Poetry -
In Any Language

How do you read the code?

Is every part imbued with meaning?

Is it precise, or open to interpretation?

You Have to Actually Read It Though

#1 security analyst question:

> How, precisely, does the concrete implementation work?!

Typical developer response:

> (rejects the premise of the question)

Abstractions are Great Tho

Abstract filters do not exist.

The semantics of everything come from how it is used.

Remember: preventing shells is only one goal among many

What Code-Data Boundary

Every program is a limited-purpose VM

The machine code is the program's input

SAST!

SAST with default rules mostly lives here:

1. A constraint has been instantly violated
--- Most of CVE probably lives here (WIP) ---

Testing! And SAST!

Domains that blend SAST and integration testing:

2. Two or more parts interact badly
3. The effects of multiple layers render the result incorrect
4. An external effect on one layer can violate a requirement

Let’s Move Up Here

5. The program does not meet specification
6. The specification is not decidable
   --- LangSec helps above this line ---
7. The specification does not match intent
8. The specifiers were not aware of all their requirements