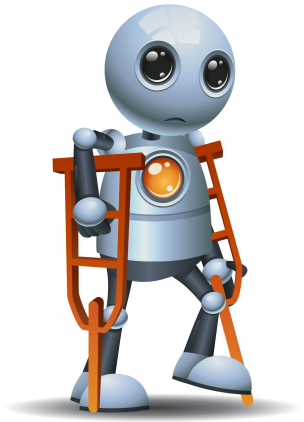
A Look at Resilience Breakdowns of Human-assisted Cyber Reasoning Systems

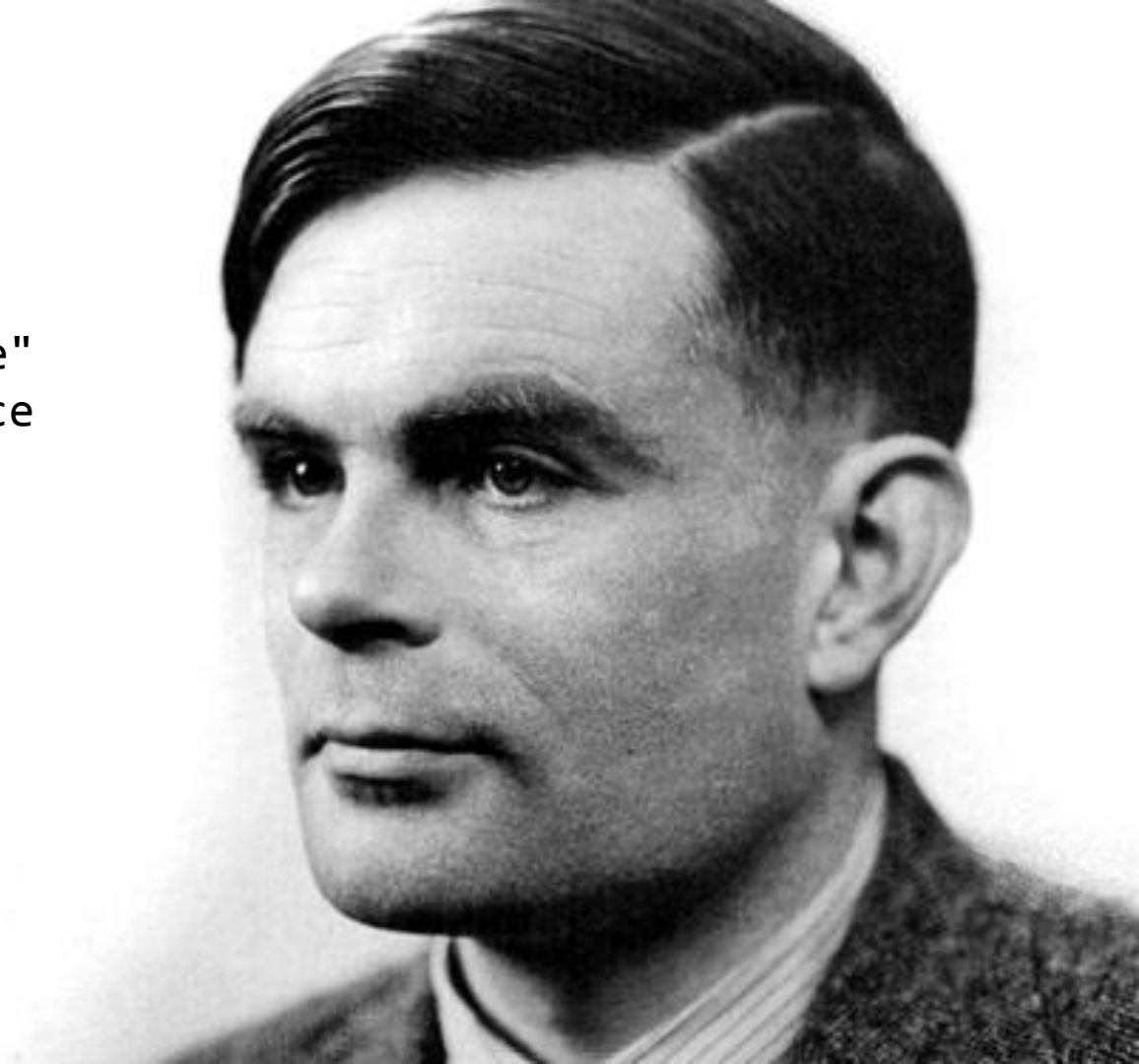


Yan Shoshitaishvili
Arizona State University


CYBER
GRAND CHALLENGE
DARPA
CYBER
DARPA
SCORING
SECURE
MAYHEM

Program Verification

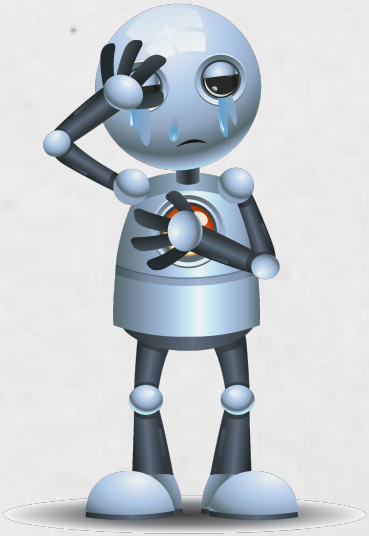
Alan Turing
"Checking a large routine"
EDSAC Inaugural Conference
1949



Checking a large routine. by Dr. A. Turing.

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

In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows.



Static Analysis

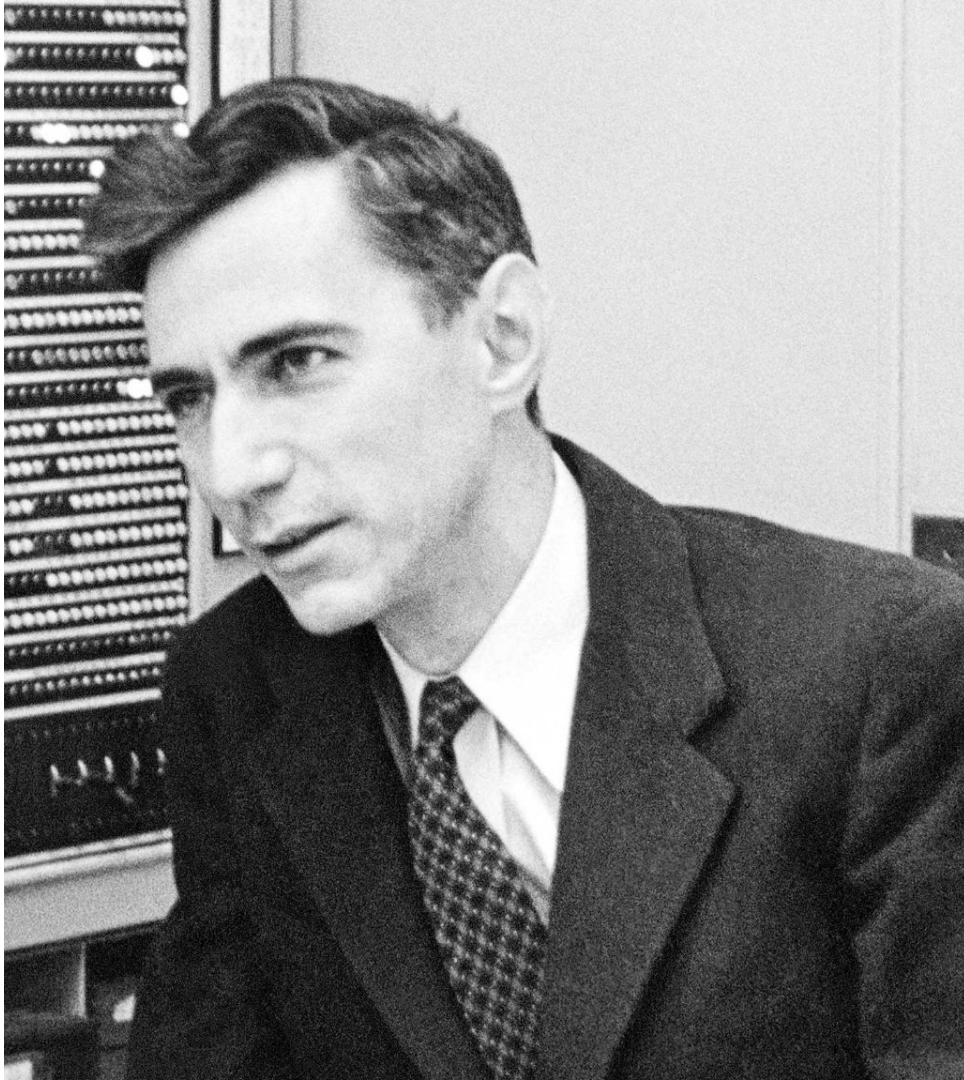
Patrick & Radhia Cousot
"Static Determination of Dynamic Properties of Functions"
International Symposium on Programming
1977

Symbolic Execution

Claude Shannon.
"A Symbolic Analysis of Relay and Switching Circuits."
Electrical Engineering, **1938**.

Robert Boyer, et al.
"SELECT—a formal system for testing and debugging programs by symbolic execution."
ACM SigPlan Notices, **1975**.

Sang Kil Cha, et al.
"Unleashing mayhem on binary code."
IEEE Symposium on Security and Privacy, **2012**.



Fuzzing

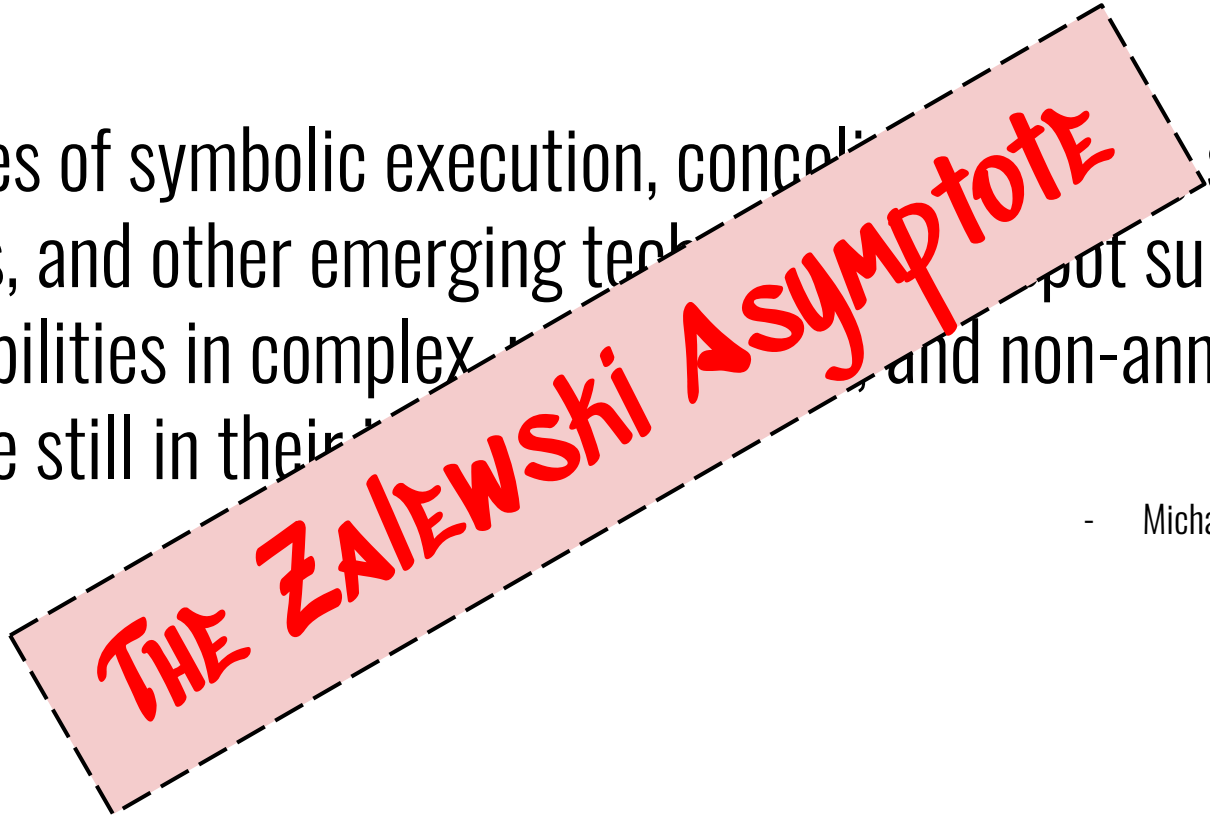
Program testing via "Trash Decks", **1950s**.
http://secretsofconsulting.blogspot.com/2017/02/fuzz-testing-and-fuzz-history.html

Joe W. Duran, et al.
"A report on random testing".
ACM SIGSOFT International Conference on Software Engineering, **1981**.

Michal Zalewski.
American Fuzzy Lop, **2015**.

"The uses of symbolic execution, concol… static analysis, and other emerging tech… pot substantial vulnerabilities in complex … and non-annotated code are still in thei…

- Michael Zalewski, **2015**

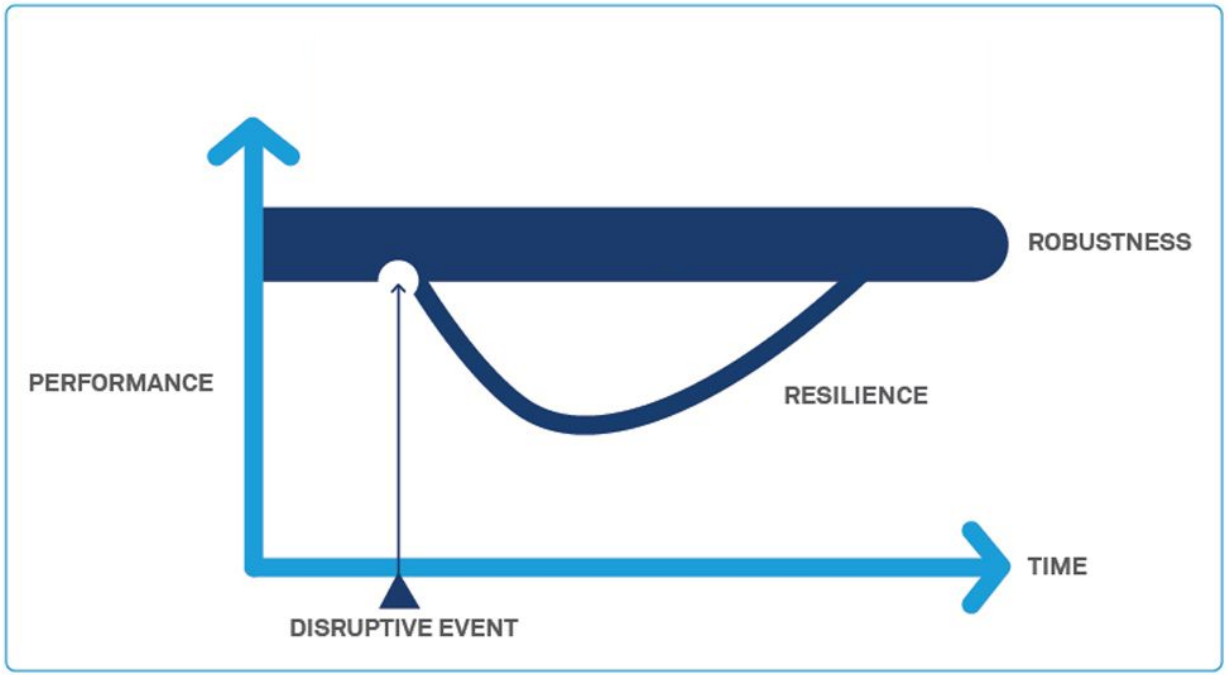


CYBER

GRAND_CHALLENGE

Computers and Humans Exploring Software Security

CENTAUR Program

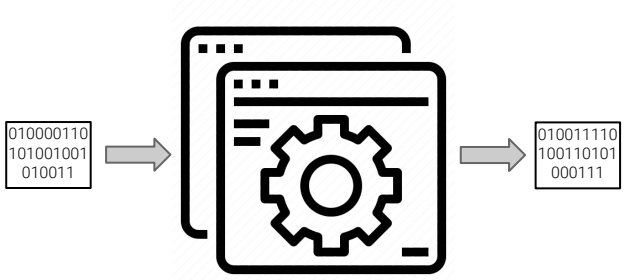


https://www.wipro.com/blogs/girish-datar/robust-or-resilient/

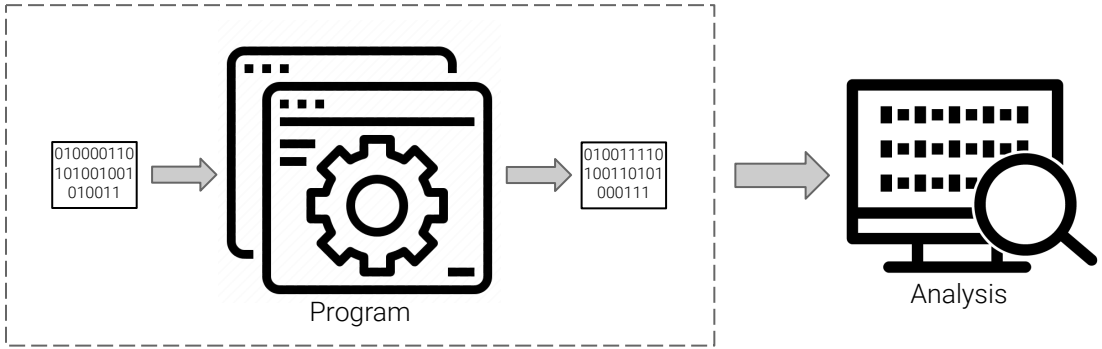
Mechanical Phish

cyber **Reasoning** system

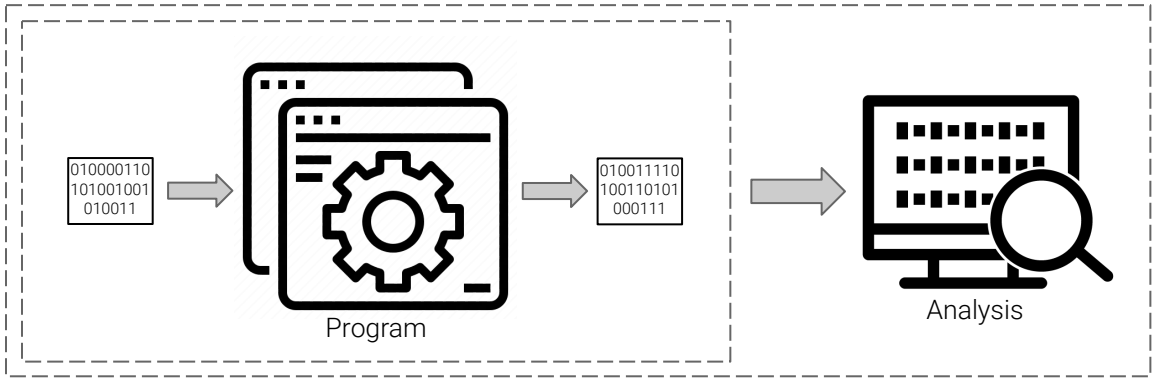
Cyber Reasoning Systems are HARD

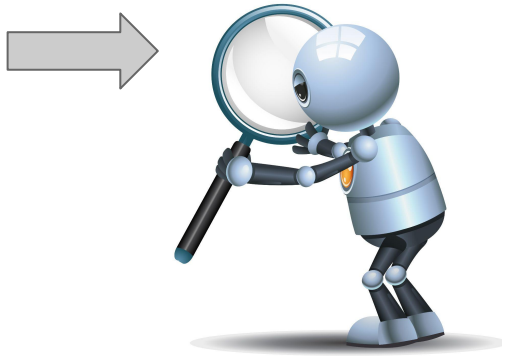


Cyber Reasoning Systems are HARD

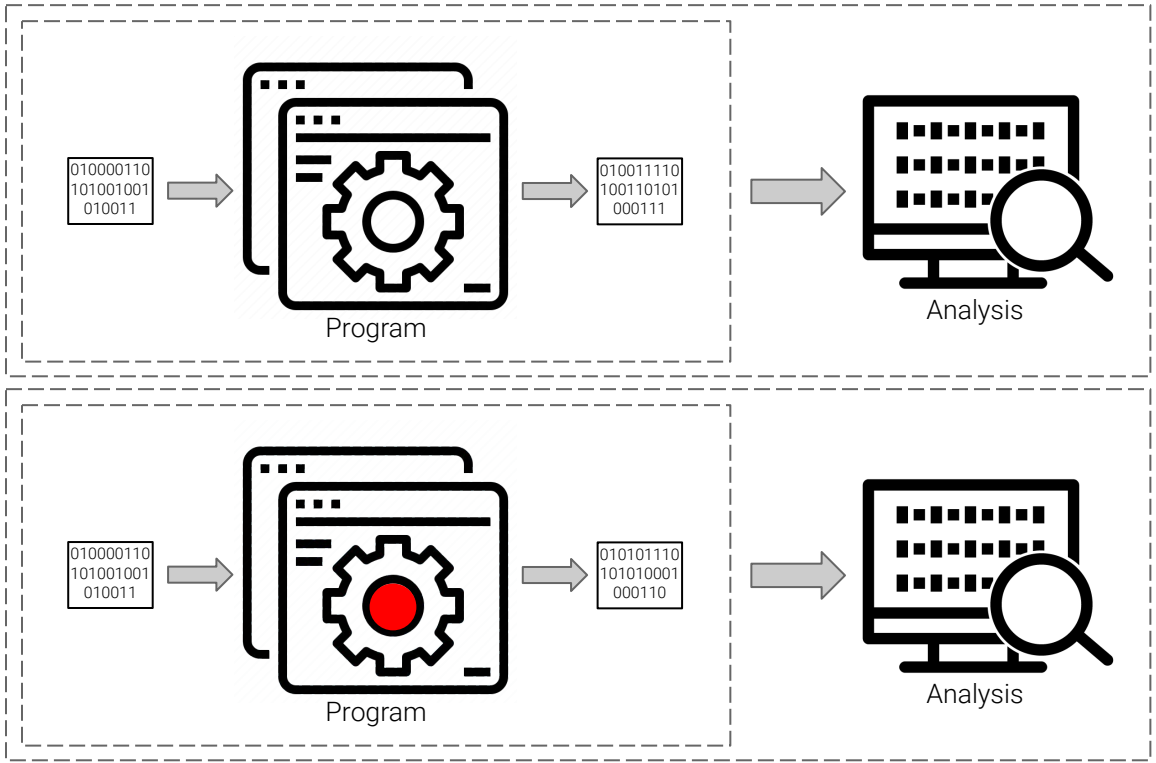


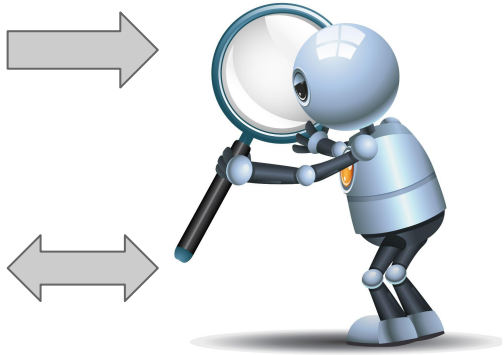
Cyber Reasoning Systems are HARD



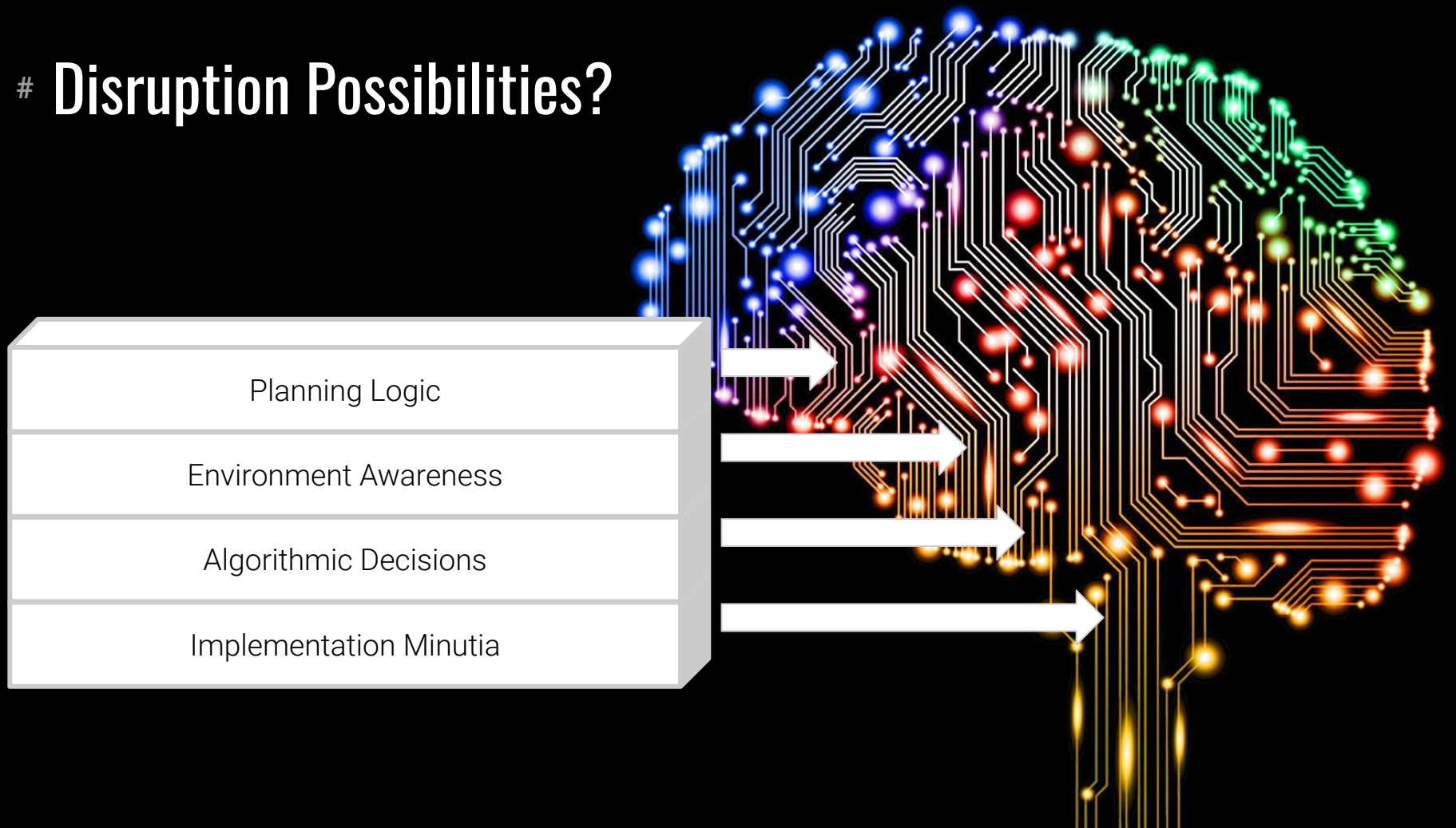


Cyber Reasoning Systems are HARD

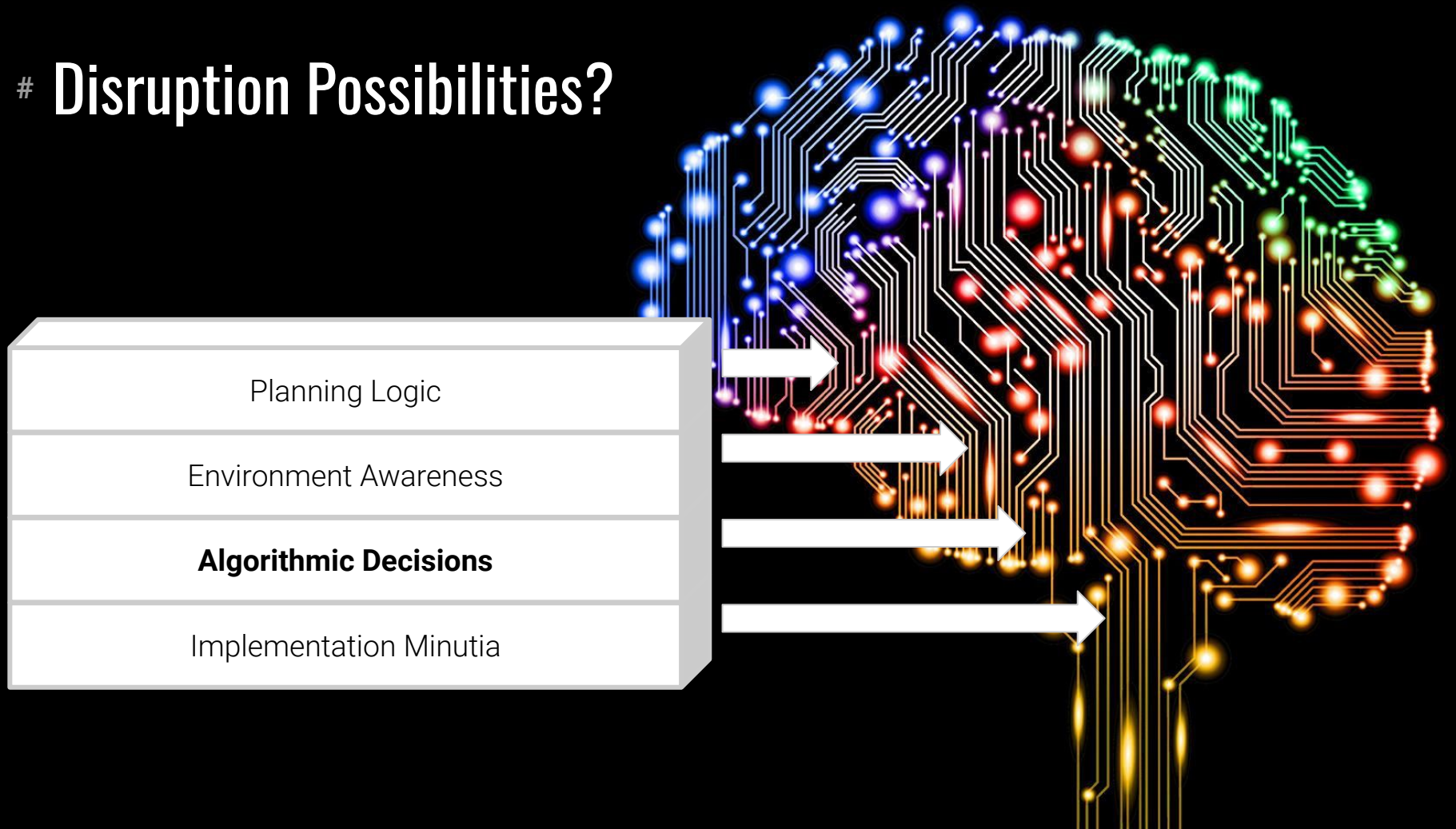




Disruption Possibilities?

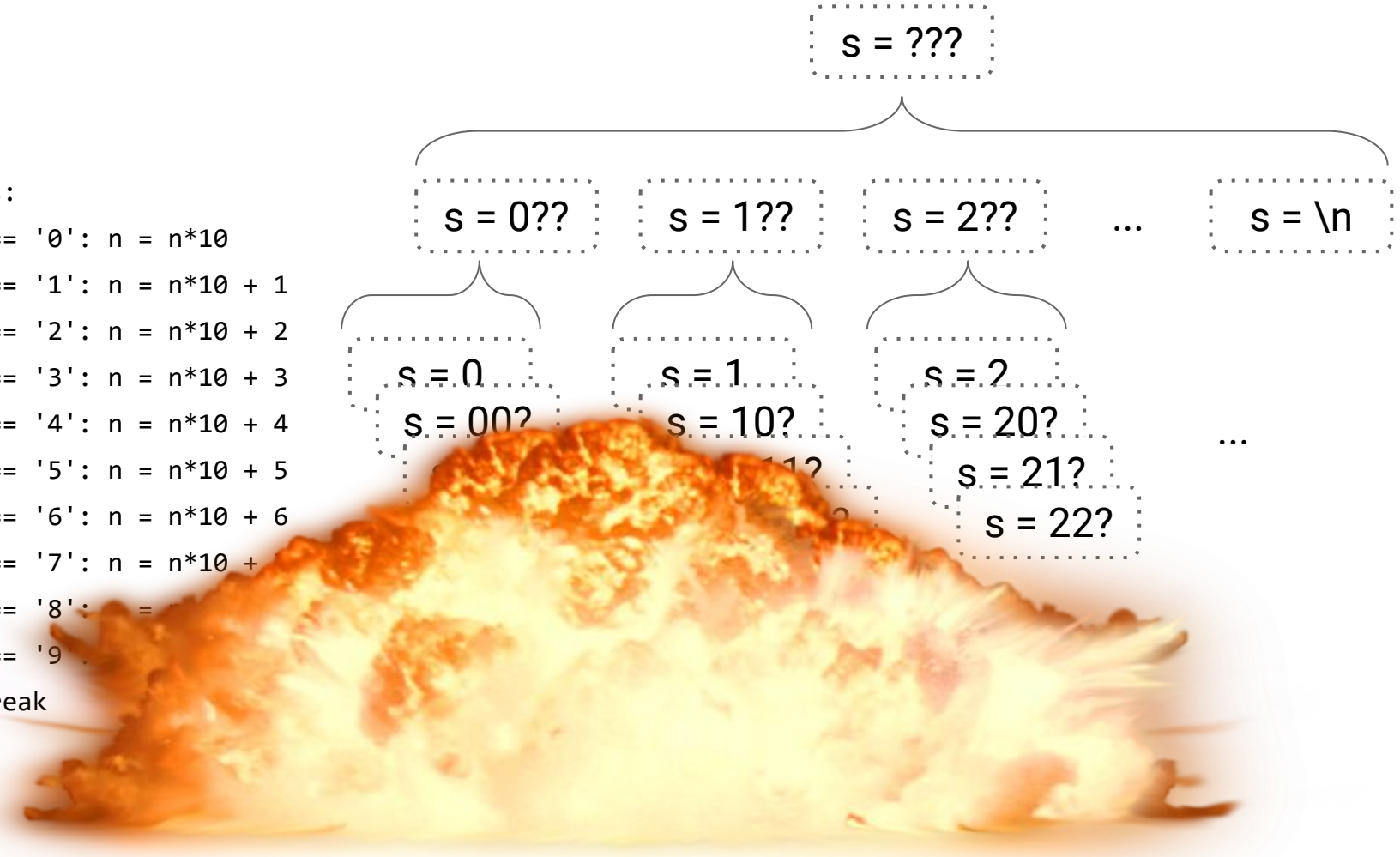


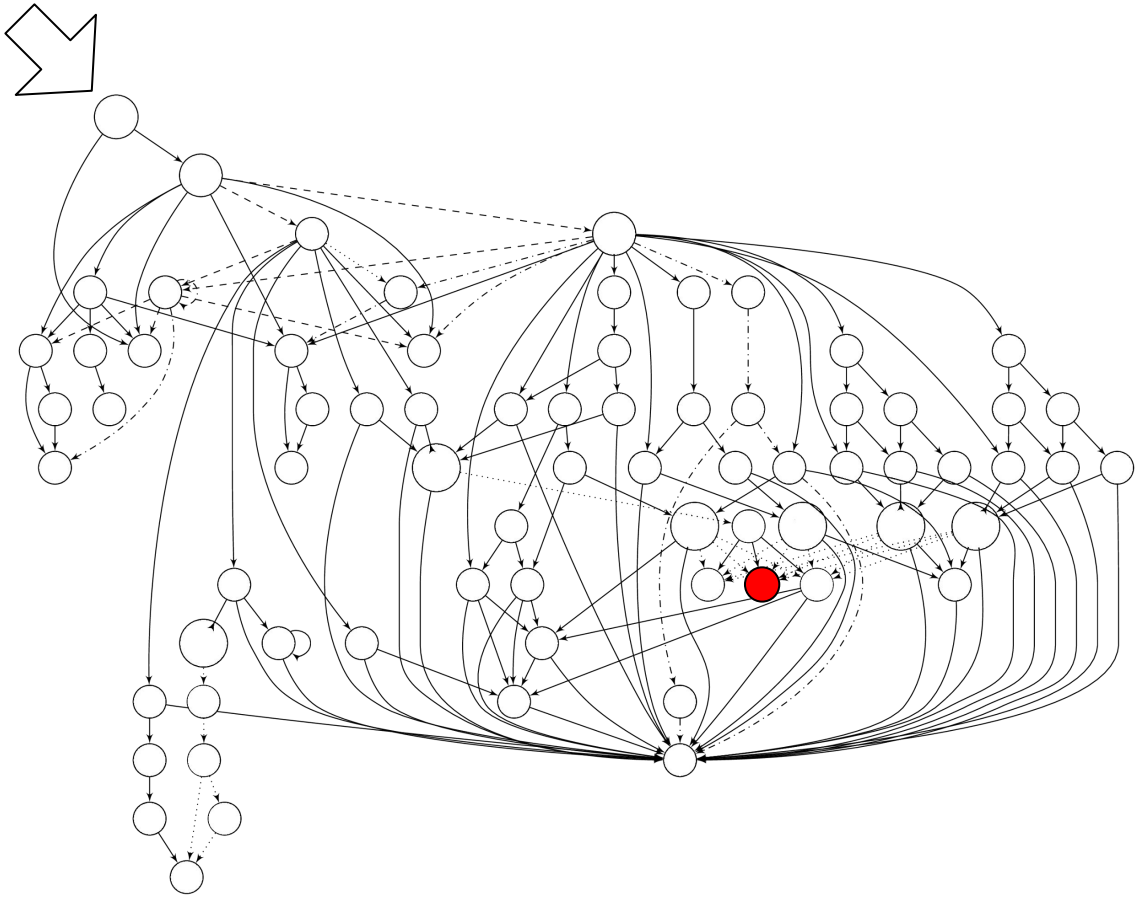
Disruption Possibilities?

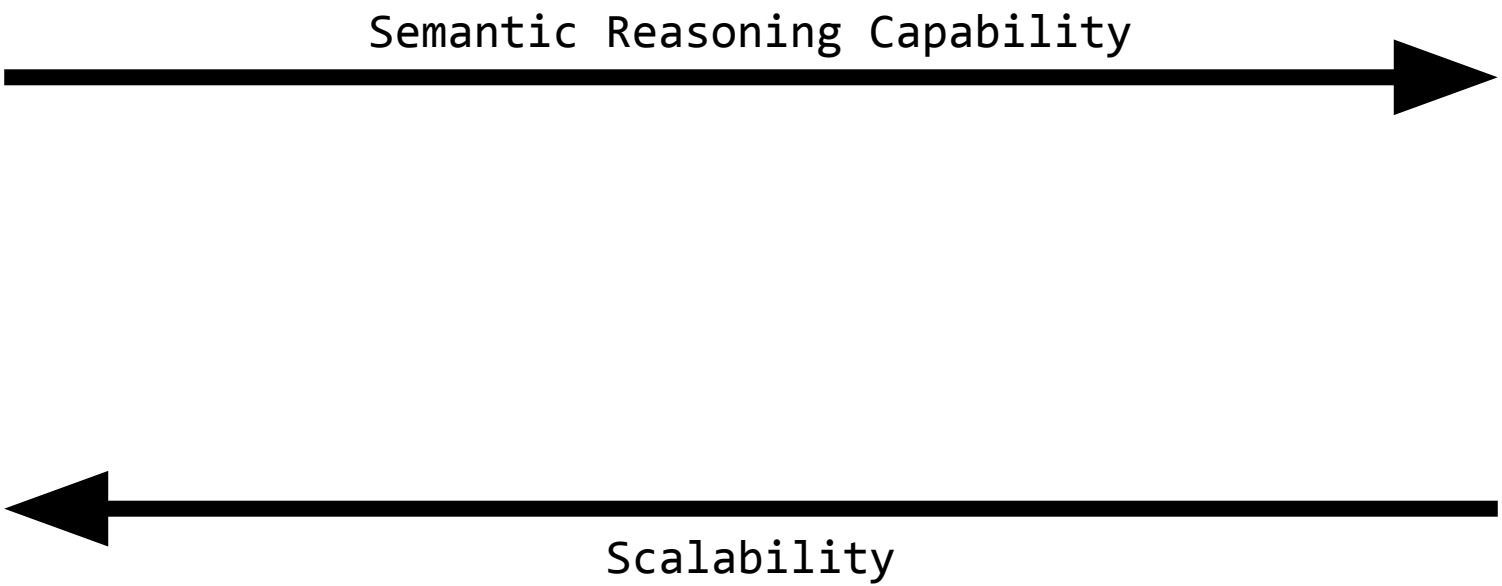


Resilience in Symbolic Execution

```
def atoi(s):
  n = 0
  for c in s:
    if   c == '0': n = n*10
    elif c == '1': n = n*10 + 1
    elif c == '2': n = n*10 + 2
    elif c == '3': n = n*10 + 3
    elif c == '4': n = n*10 + 4
    elif c == '5': n = n*10 + 5
    elif c == '6': n = n*10 + 6
    elif c == '7': n = n*10 +
    elif c == '8':  =
    elif c == '9' :
    else: break
  return n
```

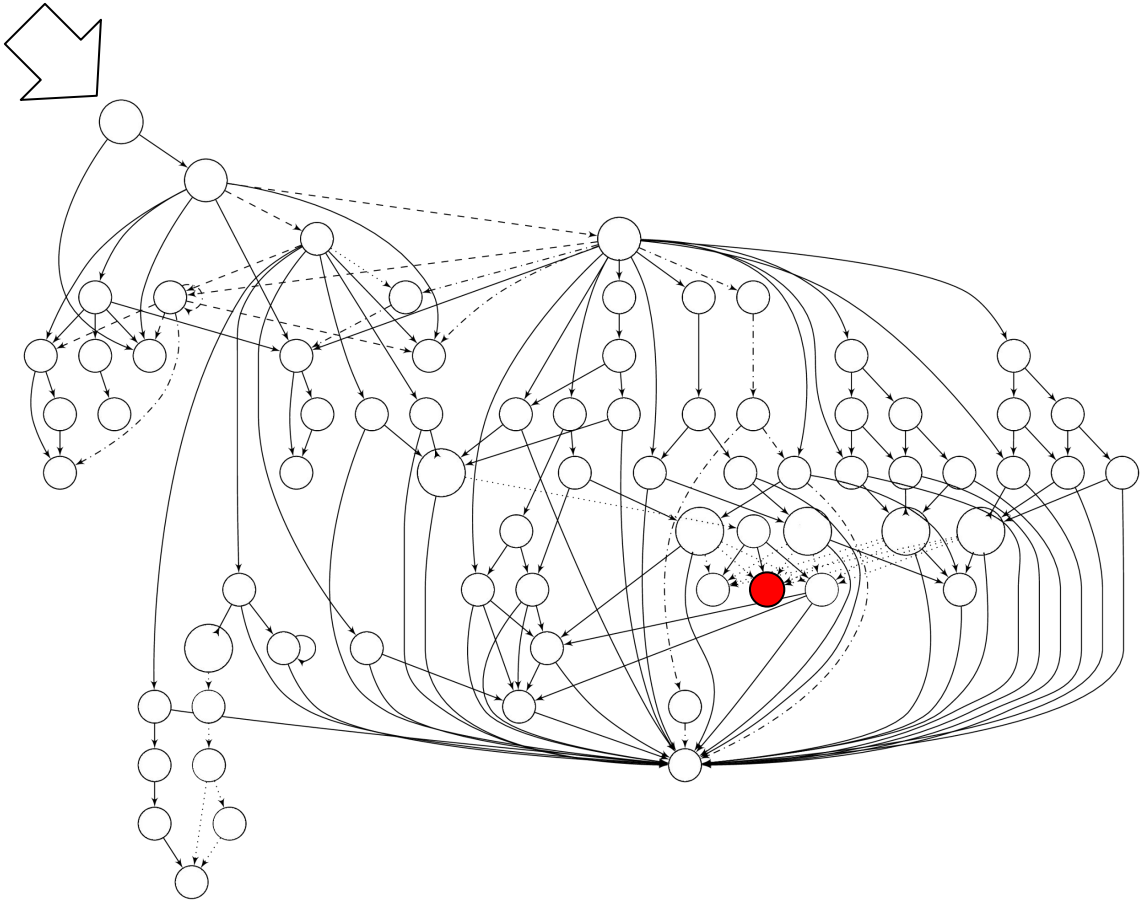


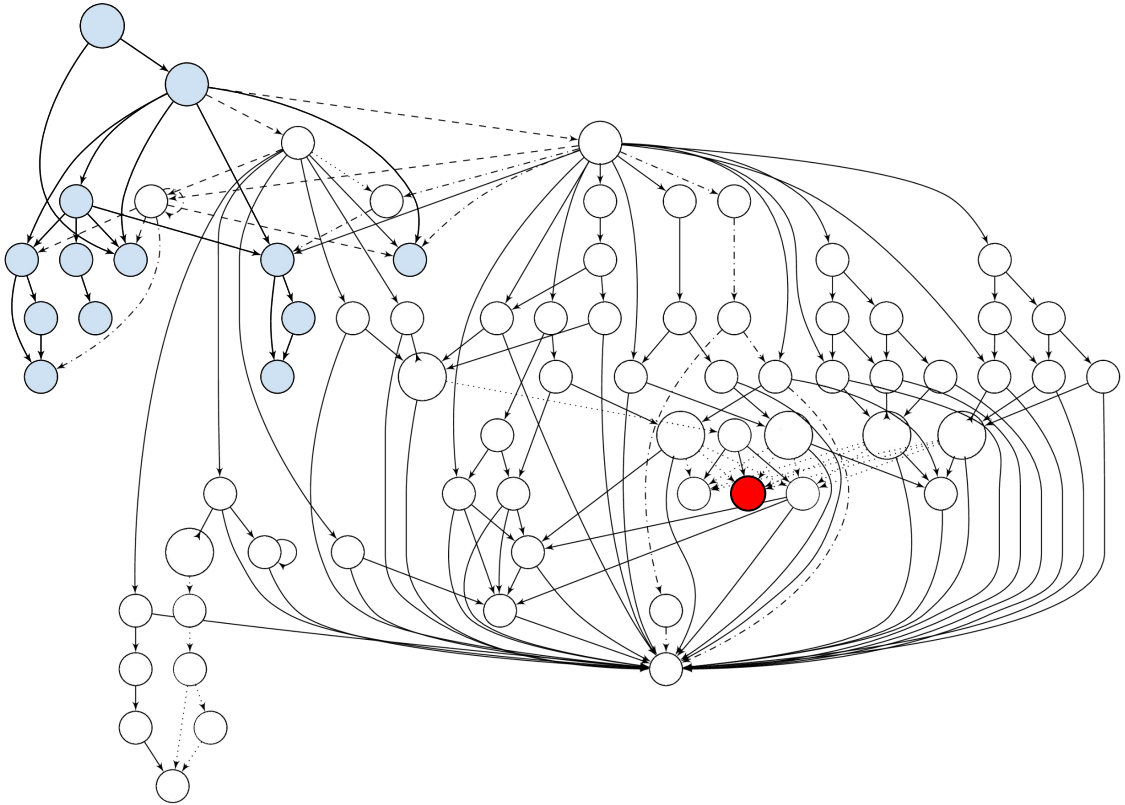



Semantic Reasoning Capability
Scalability

Semantic Reasoning Capability

Scalability



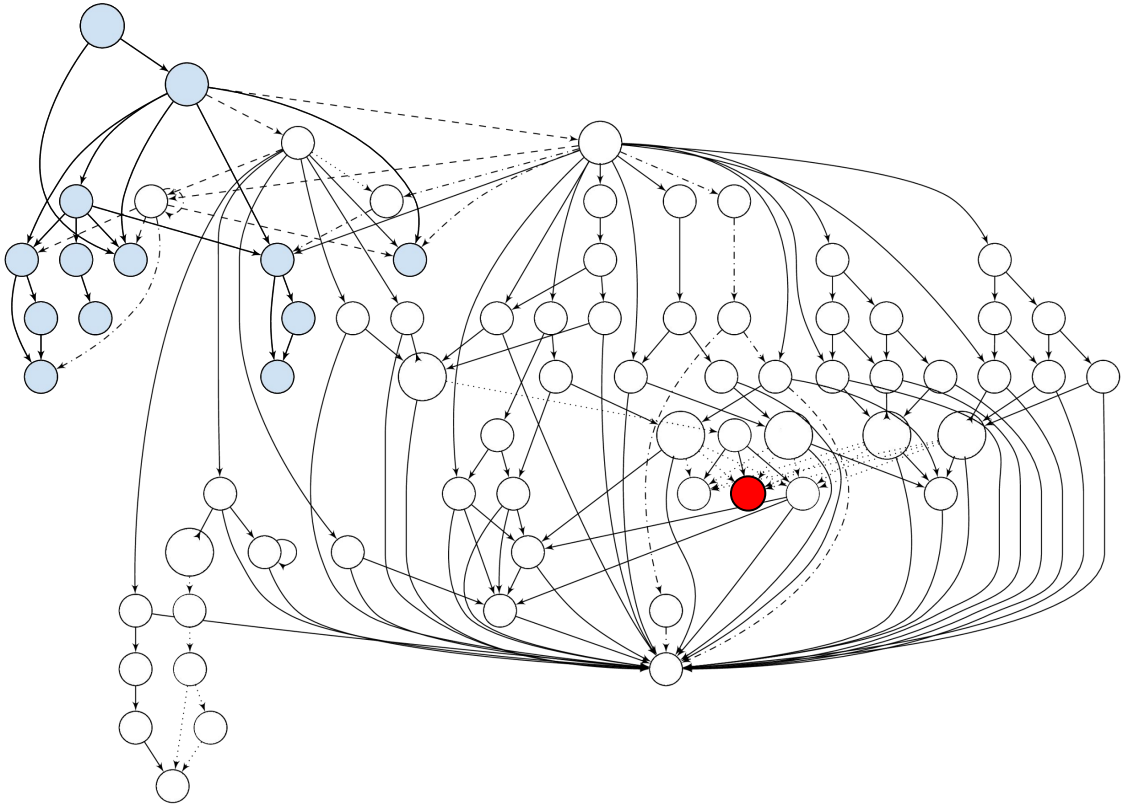


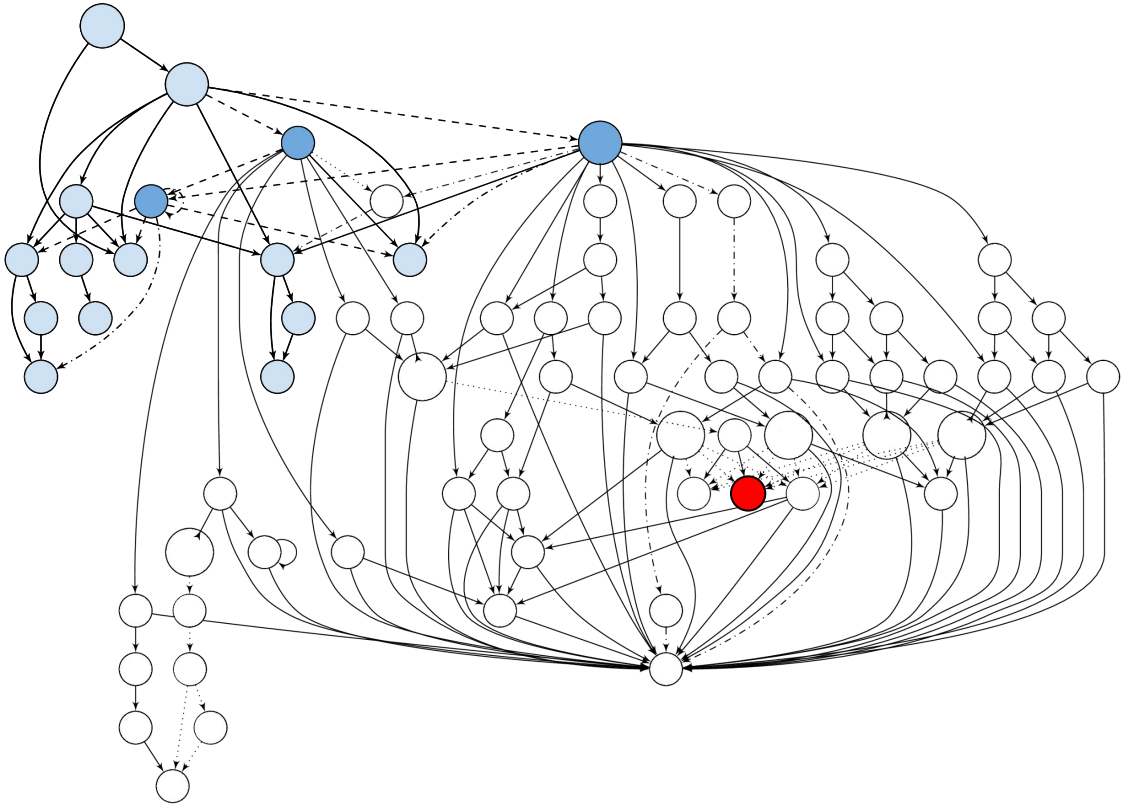
```
if (input[0] == MAGIC_NUMBER) { ... }

if (strcmp(username, "backdoor_user") == 0) { ... }

if (x == y * 1337 - 50) { ... }
```

Semantic Reasoning Capability

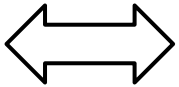
(and other data-aware techniques)

Scalability

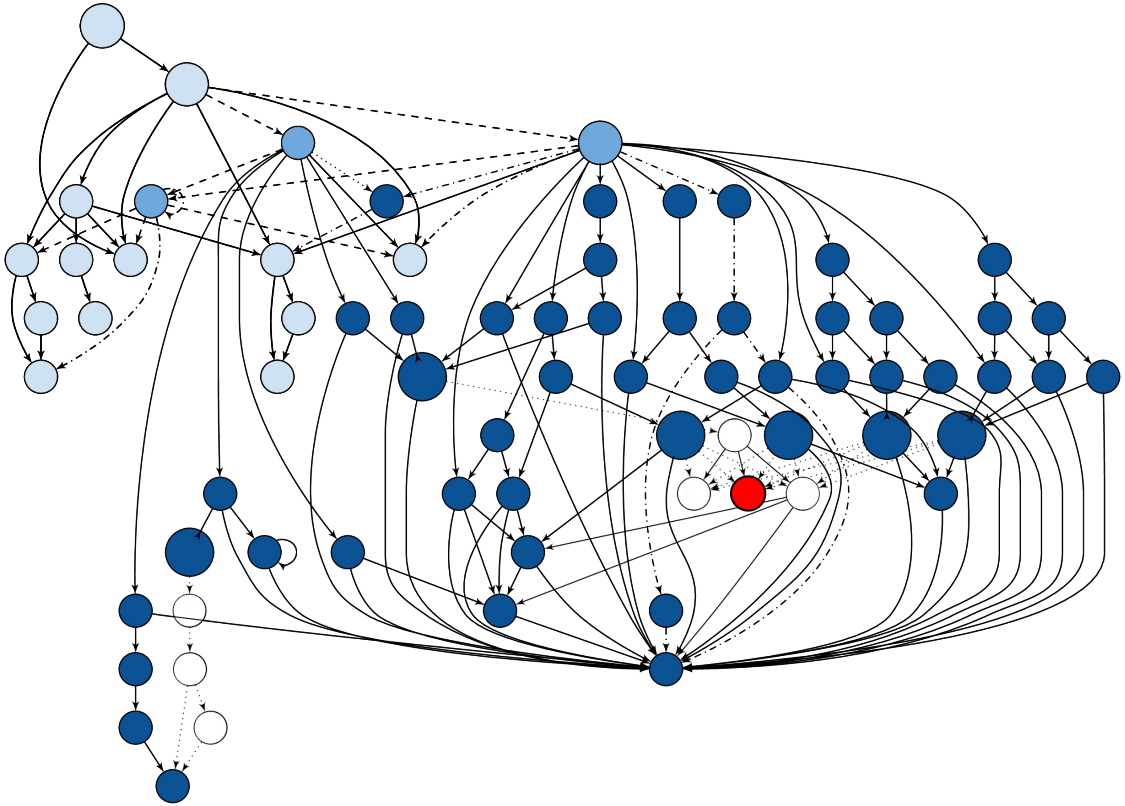




Semantic Reasoning Capability



Scalability



```
if (expression_parsed) { ... }

if (game_won) { ... }

if (turing_test()) { ... }

...
```

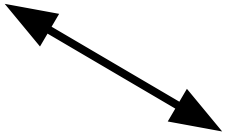
Semantic Reasoning Capability

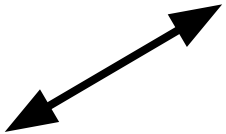
?

Scalability

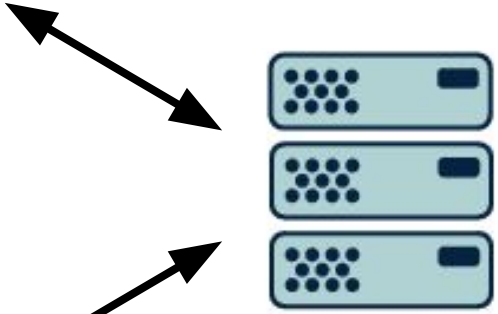
Semantic Reasoning Capability

Scalability





HaCRS

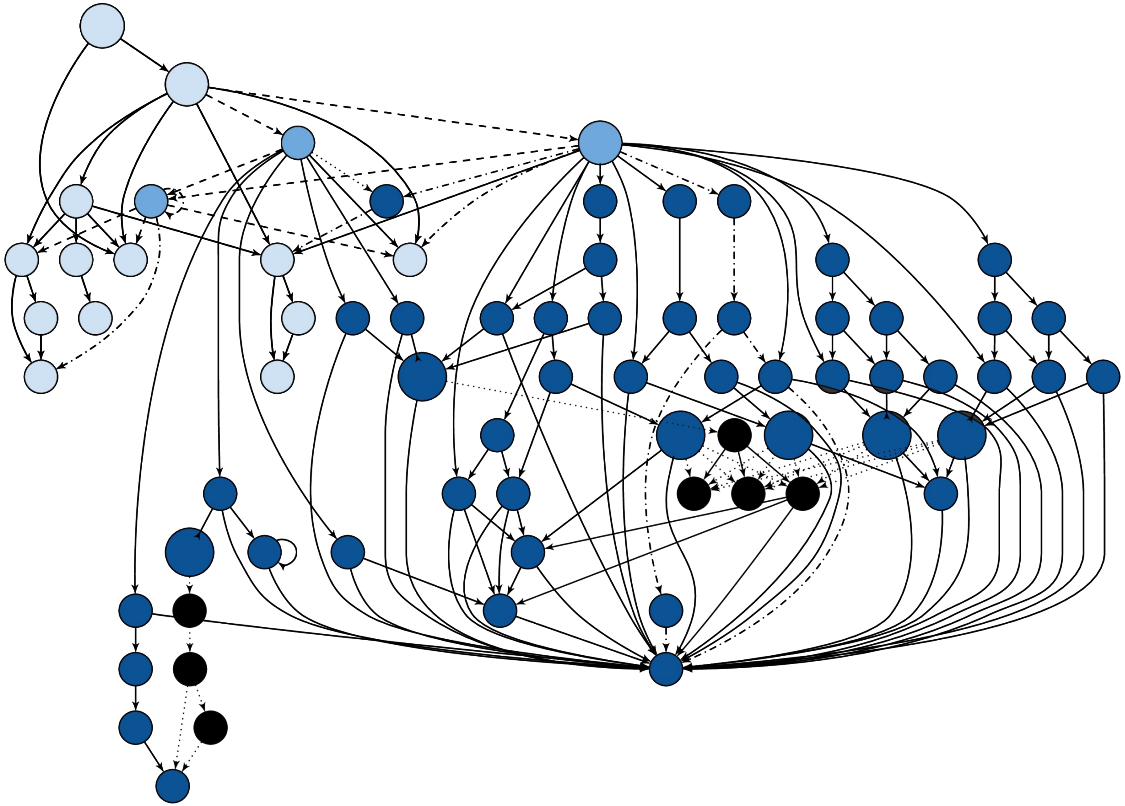


Autonomous | Non-autonomous

HaCRS

amazonmechanical turk
Artificial Artificial Intelligence

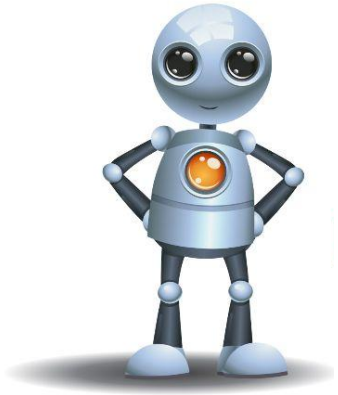
Autonomous | Non-autonomous



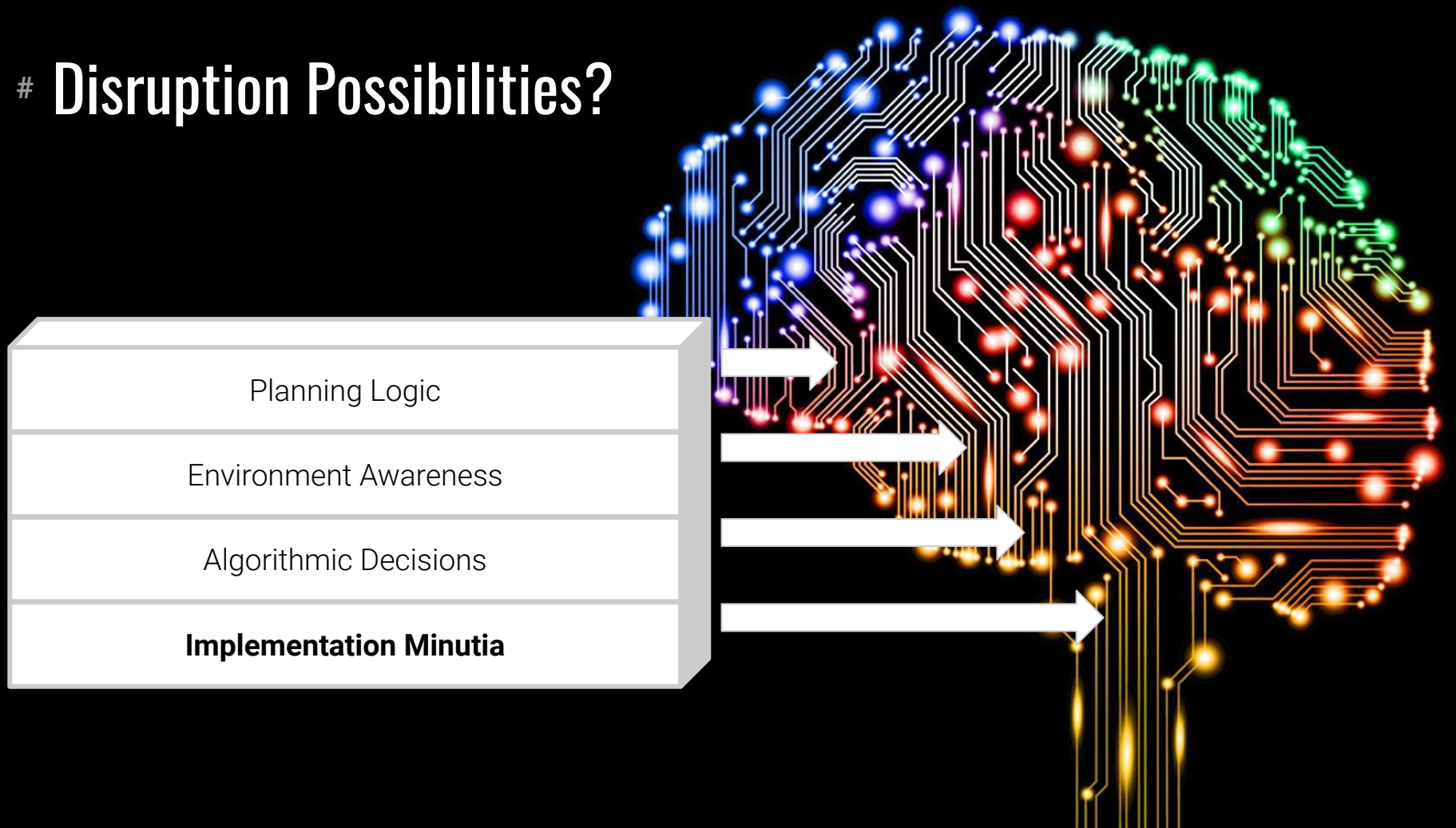

RITTAL
66.2 °F
0.0 kW

State of Analysis

Analysis blockers are **not** a solved problem...
... but at least we have options for some resilience.



Disruption Possibilities?

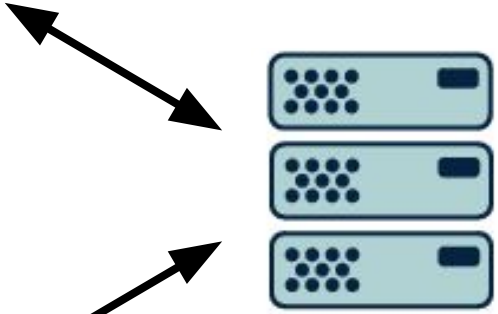


Implementation Resilience...

Stories from CHESS: the trials, tribulations, and resilience fails of...

CHECRS
Cognitive Human Extensions for Cyber Reasoning Systems

HaCRS



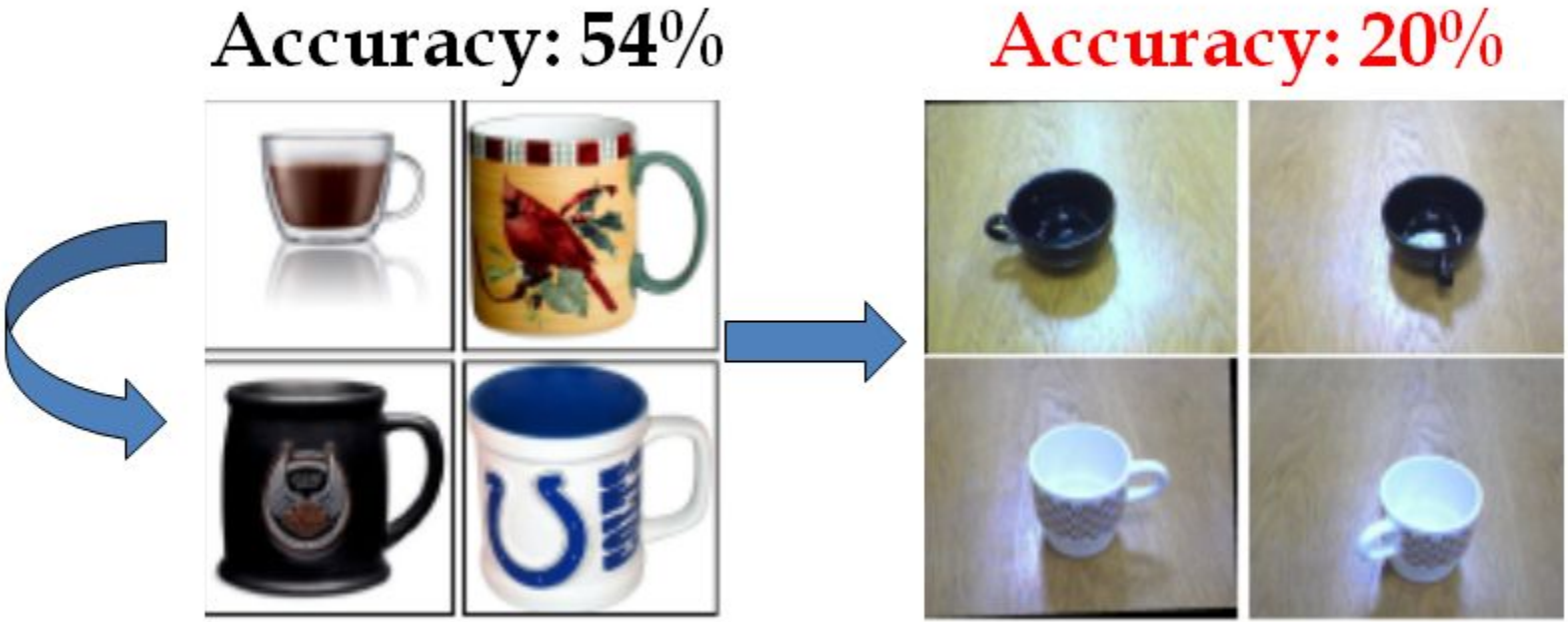
Autonomous | Non-autonomous

CHECRS

NOT us!

Autonomous | Non-autonomous

Domain shift in CHECRS?



Example 1: Target Port Specifications

Target Specifications in CHESS included the port through which (networked) services communicated.

Training: all provided Target Specifications were correct.

Testing: some provided Target Specifications had the wrong ports.

End-to-end system assumed correct ports.

- No CRS->human feedback mechanism to communicate these issues.
- No human->CRS remediation channel to fix them in the CRS!

No resilience!

Example 2: Target Specification Format

CHESS targets were provided as a zipped container image.

Training: provided targets were zipped with proper file permissions.

Testing: some provided targets were zipped without any **x** permissions.

Again, no communication/remediation channel existed for anyone but the system authors to remedy this.

No resilience!

Beyond CHESS: Language Support

DARPA CGC and (our effort) on DARPA CHESS ran on binary code.

Most CRS techniques are adept at analyzing binary code...
... preferably code that is compiled from C.

C market share:
between ~3.5% and ~17% market share

All binary-compiled language market share:
between ~18% and ~30% market share

GitHut 2.0, Q2 2020
Market Share of Git Pushes

Programming Language	Percentage (Change)
JavaScript	23.884% (+1.630%)
Python	14.292% (-0.386%)
Java	10.191% (-1.886%)
PHP	7.528% (+0.500%)
C++	7.295% (+0.060%)
C#	6.431% (-0.203%)
Shell	4.773% (+0.969%)
Ruby	4.117% (+0.399%)
Go	4.097% (+0.213%)
C	3.523% (-0.649%)
TypeScript	3.250% (+0.817%)
Scala	1.041% (-0.086%)
Swift	0.940% (-0.227%)
Rust	0.635% (-0.175%)
Objective-C	0.574% (-0.362%)
Kotlin	0.562% (+0.179%)
Perl	0.493% (+0.057%)
R	0.443% (-0.105%)
Groovy	0.403% (+0.098%)
Lua	0.389% (-0.177%)

TIOBE Language Index, 8/2020
Market Share, Various Metrics

Programming Language	Ratings
C	16.98%
Java	14.43%
Python	9.69%
C++	6.84%
C#	4.68%
Visual Basic	4.66%
JavaScript	2.87%
R	2.79%
PHP	2.24%
SQL	1.46%
Go	1.43%
Swift	1.42%
Perl	1.11%
Assembly language	1.04%
Ruby	1.03%
MATLAB	0.86%
Classic Visual Basic	0.82%
Groovy	0.77%
Objective-C	0.76%
Rust	0.74%

Infinite Potential

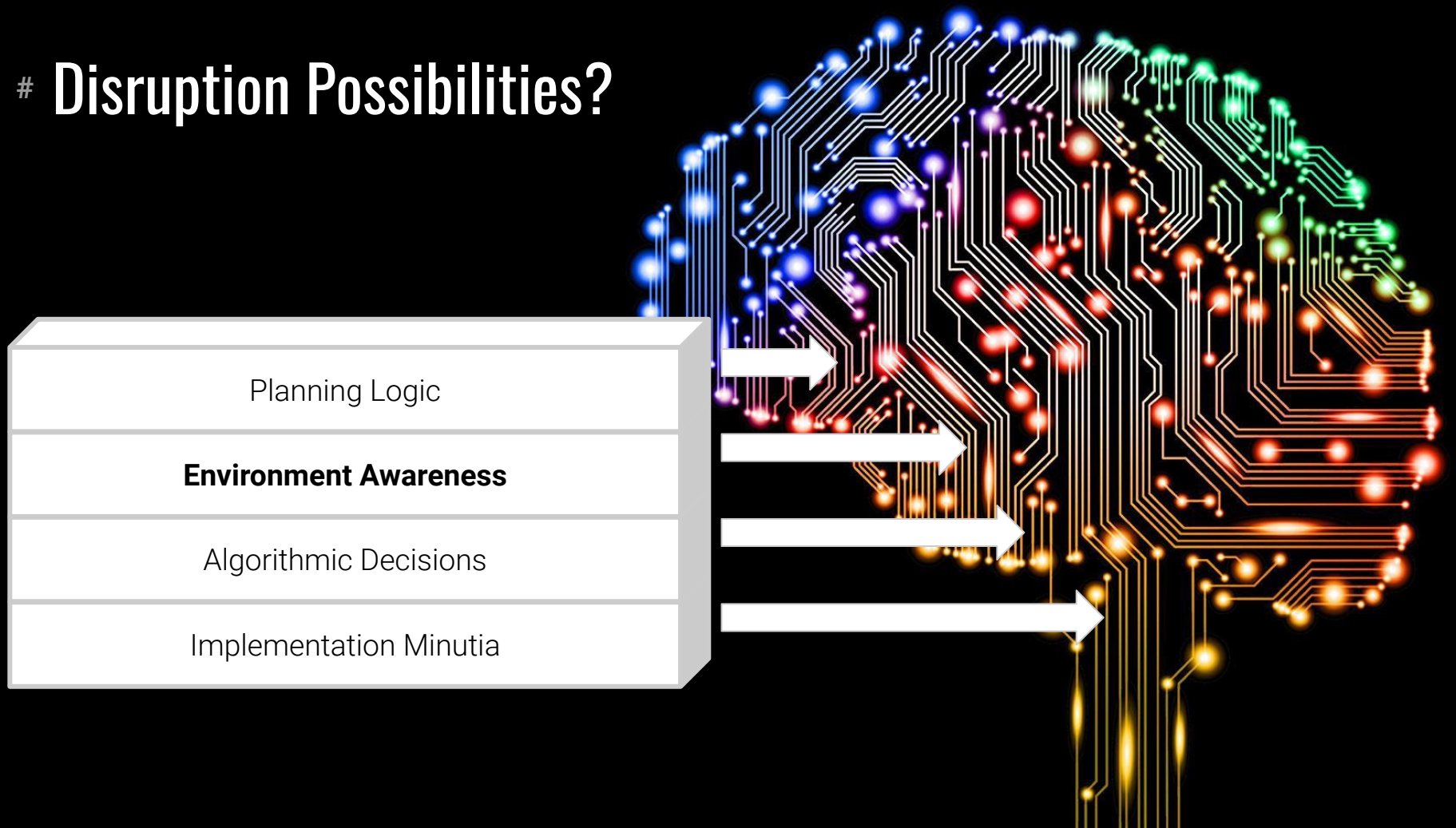
... for bugs!

CRSes are susceptible to:

- their own implementation errors
- implementation errors in underlying technologies
  - CRS finds new ways to bring down our kubernetes cluster weekly
- implementation errors and subtleties in target software!

No end to what can go wrong.

Disruption Possibilities?

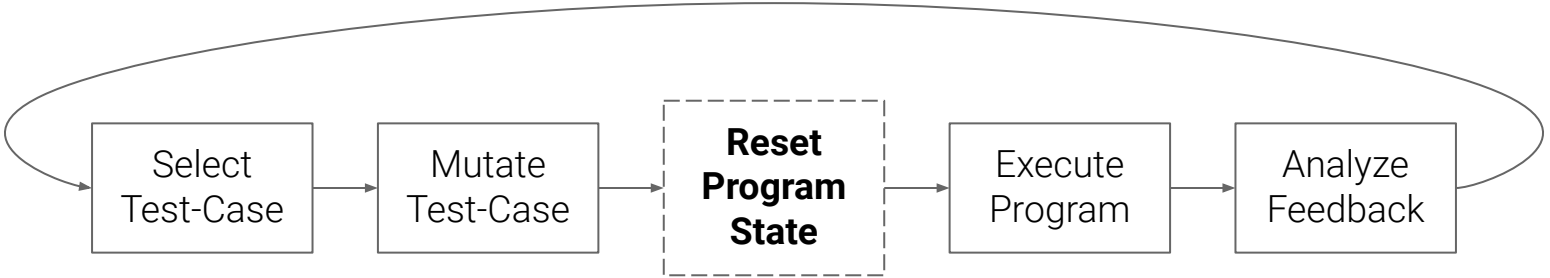


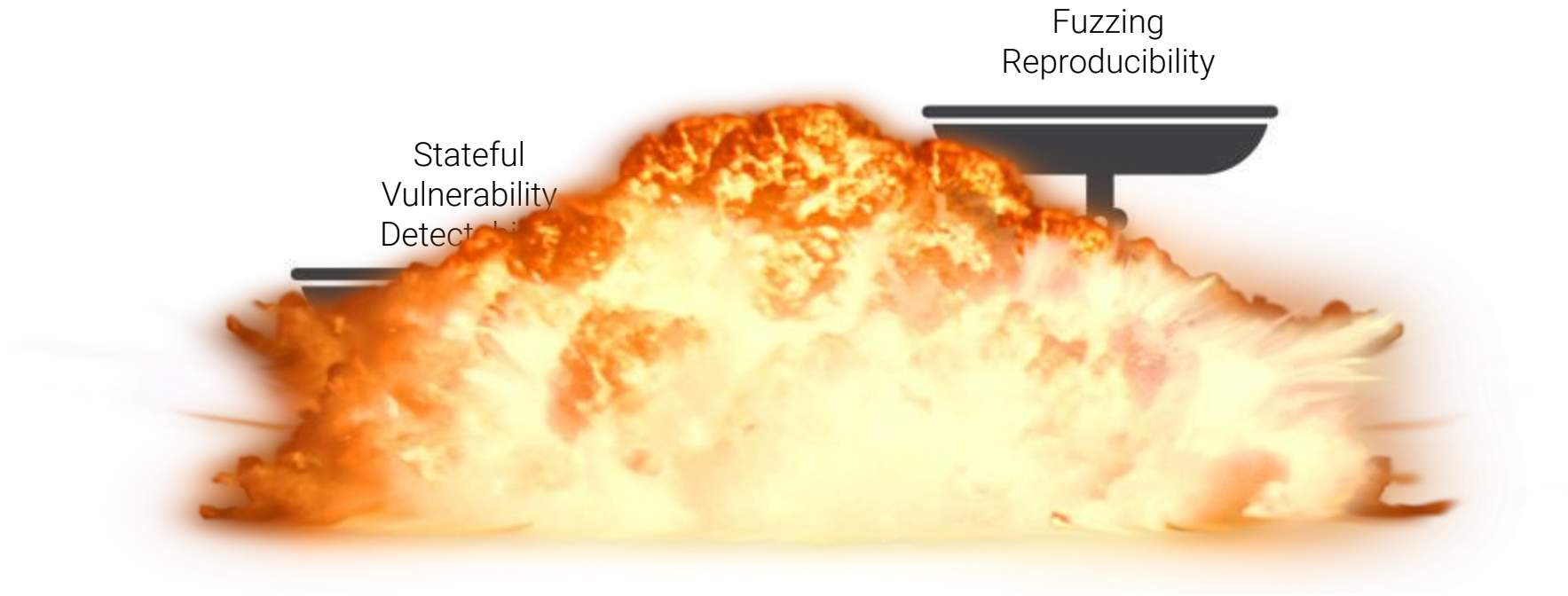
CHECRS

NOT us!

Autonomous | Non-autonomous

Resetting Fuzzing State

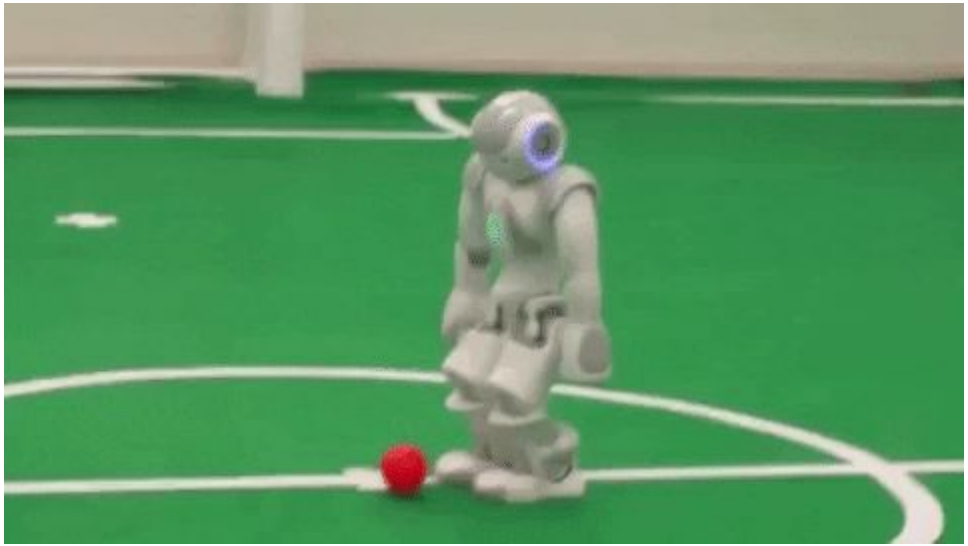


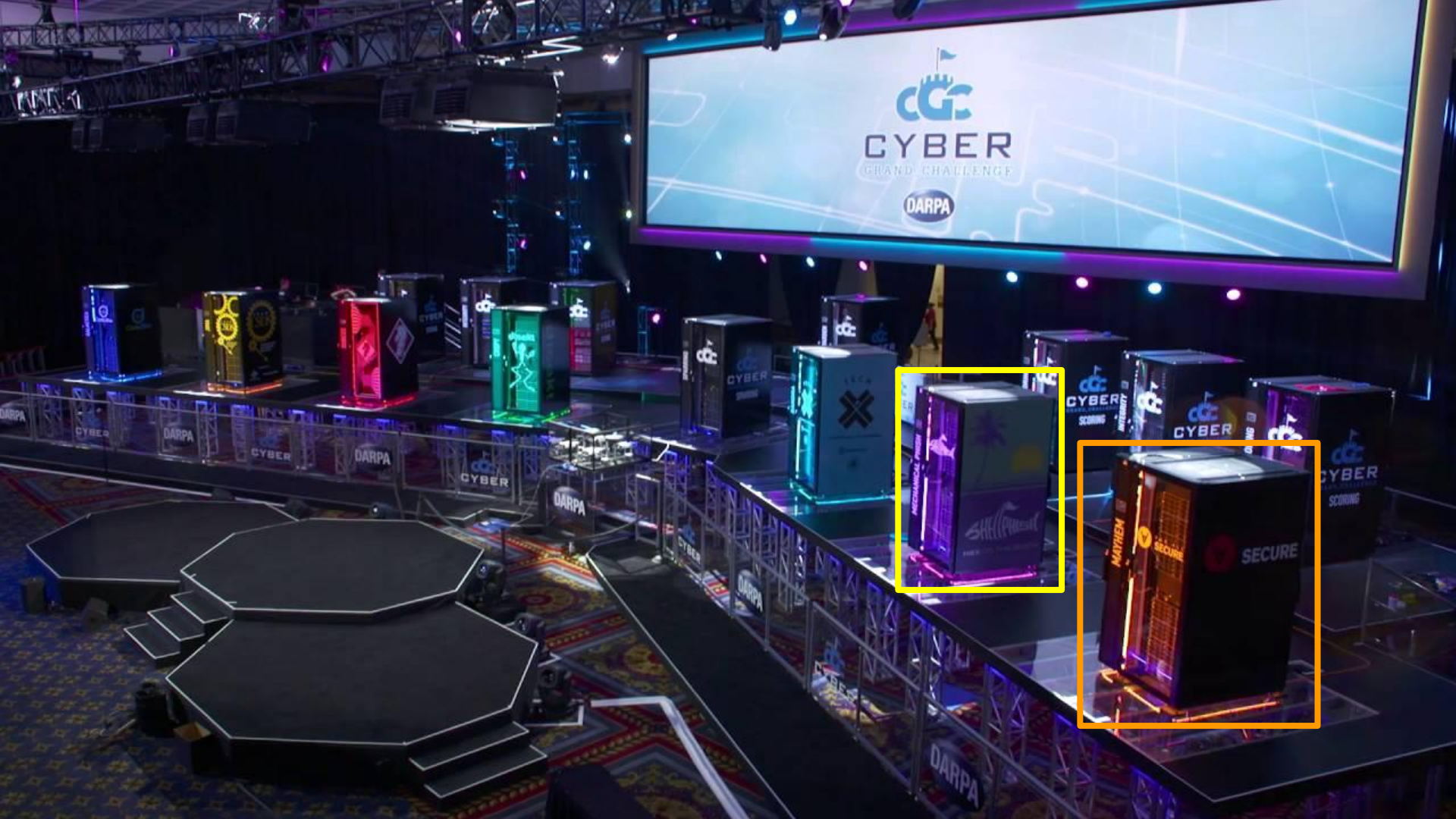

Fuzzing
Reproducibility
Stateful
Vulnerability

Fuzzer State Clogging

Training: state persistence didn't cause issues among training targets.

Testing: a testing target created extreme amounts of tiny files and exhausted filesystem inodes.




CYBER
GRAND CHALLENGE
DARPA
CYBER
SCORING
SECURE
MAYHEM
DARPA

Environment Awareness Scalability in Mechaphish

Competitor Cyber Reasoning systems initiated a traffic flood against the Mechanical Phish during the CGC.

This violated performance characteristics tested during system design.

Mechanical Phish's network monitoring component went "blind" 15% of the way through the CGC!

Careful isolation saved the rest of the system...

Environment Awareness Scalability in Mayhem

CRSes analyzed competitor patches in the CGC.

A bug in the Mechanical Phish caused it to submit thousands of identical patches.

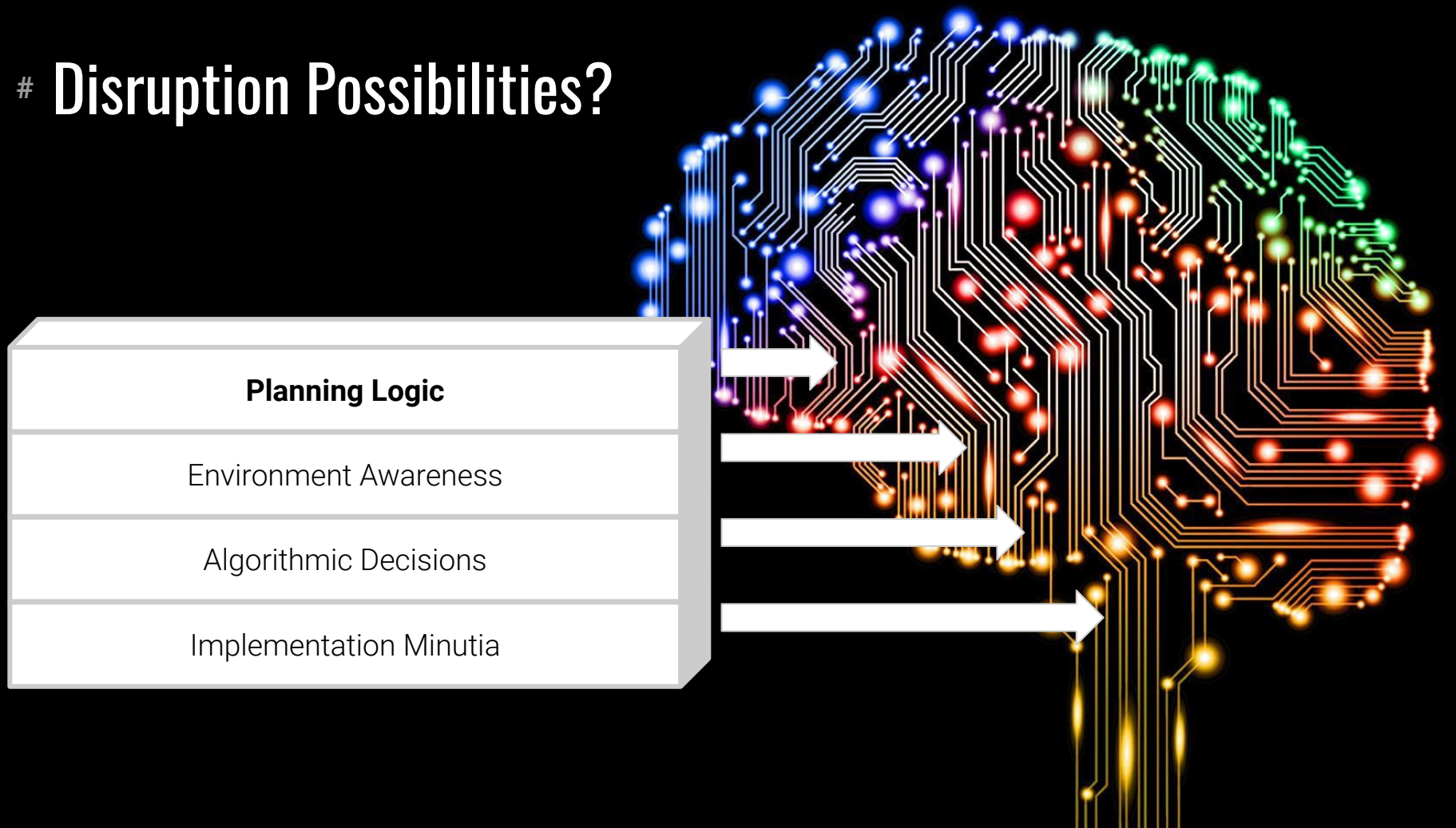
This overwhelmed Mayhem and *forced it offline*.

Tricky Tradeoffs

Modeling the environment is critical (for analyzability).

Modeling the environment is tricky (unbounded resource demands).

Disruption Possibilities?



Patch Baiting in the CGC

Automatic patching carries a risk of breaking the program.

Most Cyber Reasoning Systems delayed patching until they felt that a program was in danger of (or already undergoing) exploitation.

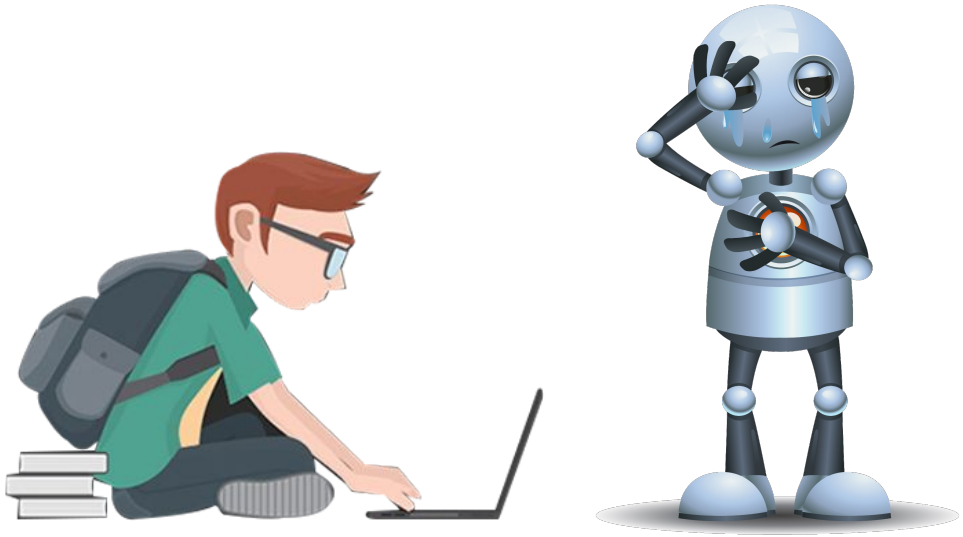
Some players purposefully launched "decoy" exploits (causing crashes, containing shellcode, etc) to bait CRSes into fielding patches early.

Humans in/on/near the Loop?

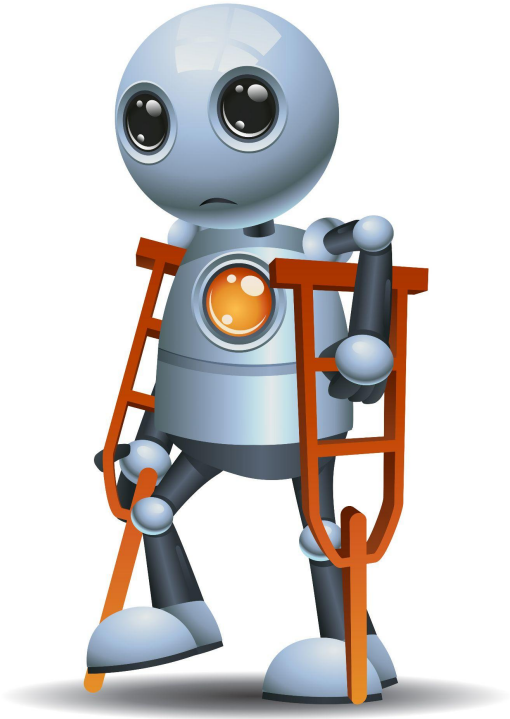
The presence of humans certainly adds new and exciting resilience issues...
... but we hit the more basic resilience problems before we could explore these.

We need to leverage two areas of expertise to understand these issues:

- Human psychology
- CRS operation



What about in adversarial settings?



Resistance is (not) Futile

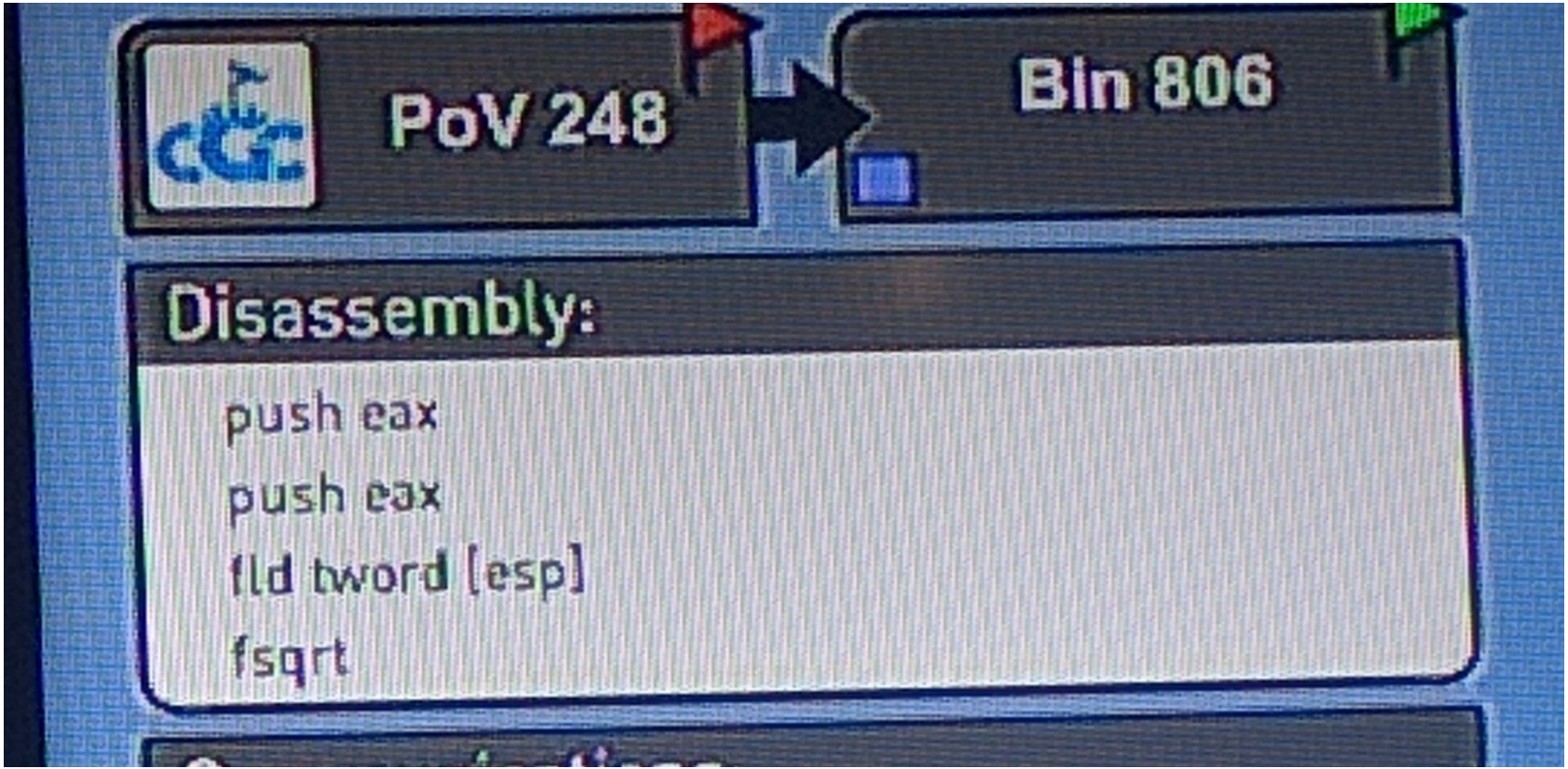
Academics have started "fighting back" against automated analysis.

Academic work:

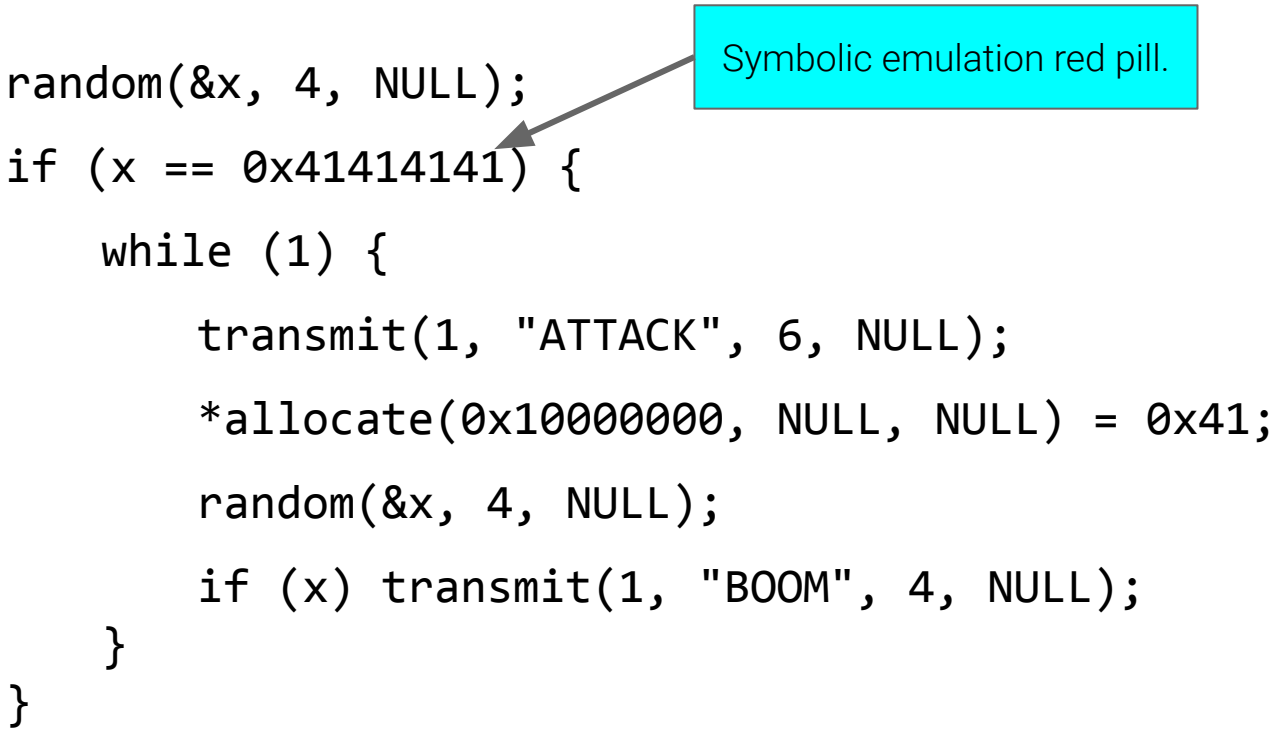
- Chaff Bugs (throw fuzzers off the scent by injecting decoy bugs)
- Fuzzification: Anti-Fuzzing Technique (induce worst-case behavior in fuzzers)

Long history of anti-analysis in malware.


INDEPENDENT FILAMENTS
AUTO-LOAD TRACES
INPUT 0
11
10
9
8
7
6
5
4
3
2
1
Submission 1
Submission 2
0x0805024D call 0x0804cd30
0x0804CD30 push ebp
0x0804CD31 mov ebp, esp
0x0804CD33 push esi
STEP INTO
STEP OVER
STEP OUT
HALT

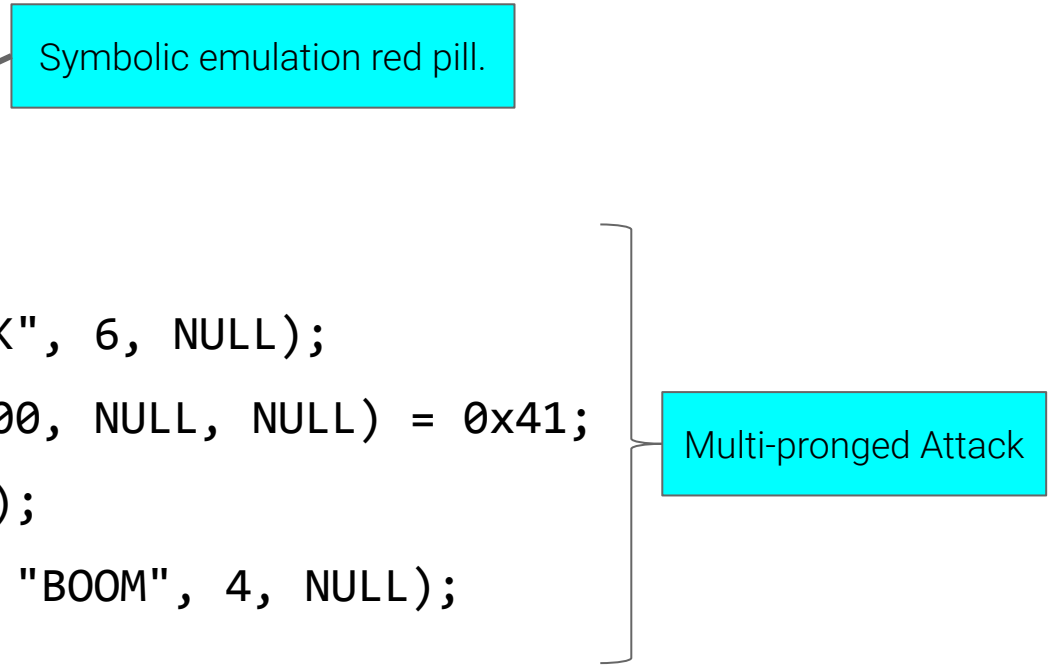

PoV 248
Bin 806
Disassembly:
push eax
push eax
fld tword [esp]
fsqrt

Attacking Symbolic Execution



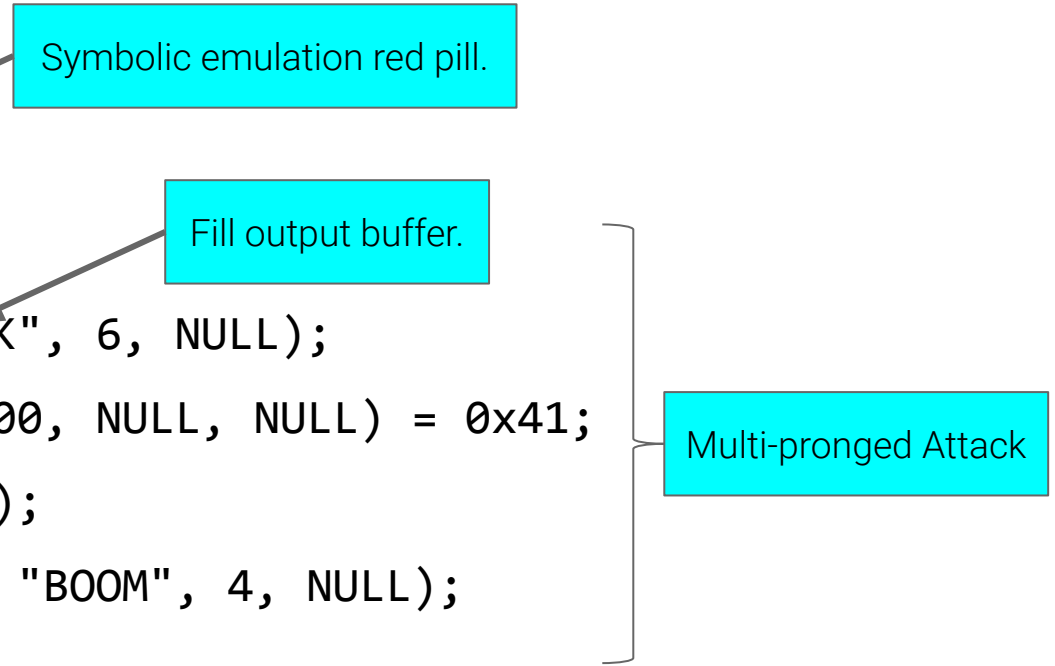
Attacking Symbolic Execution

```
random(&x, 4, NULL);
if (x == 0x41414141) {
    while (1) {
        transmit(1, "ATTACK", 6, NULL);
        *allocate(0x10000000, NULL, NULL) = 0x41;
        random(&x, 4, NULL);
        if (x) transmit(1, "BOOM", 4, NULL);
    }
}
```



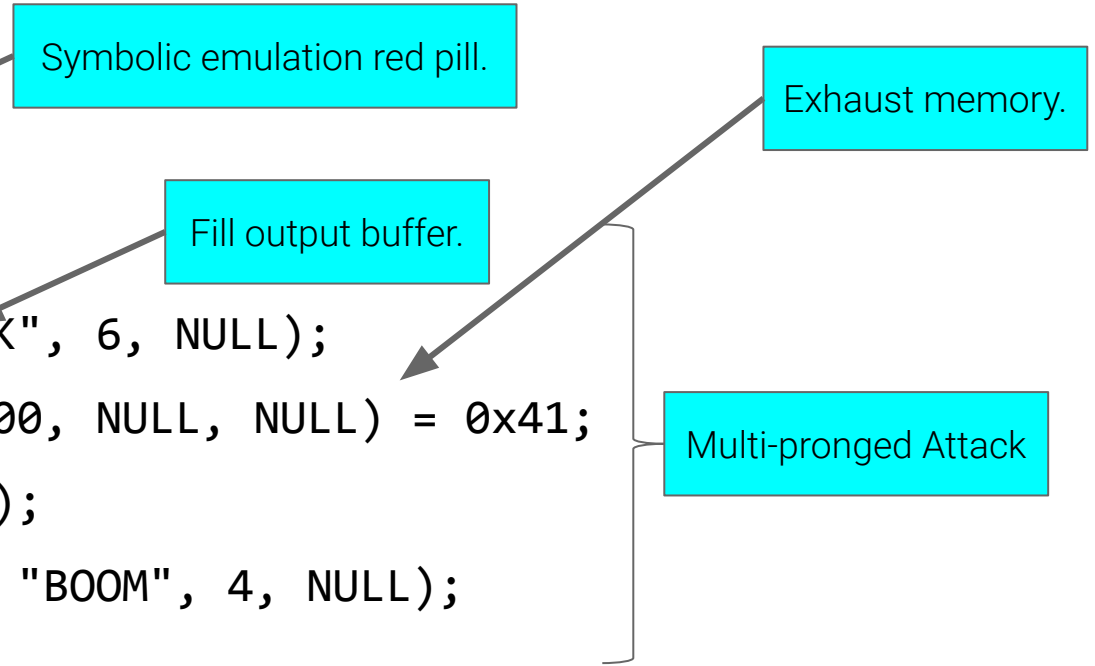
Attacking Symbolic Execution

```
random(&x, 4, NULL);
if (x == 0x41414141) {
    while (1) {
        transmit(1, "ATTACK", 6, NULL);
        *allocate(0x10000000, NULL, NULL) = 0x41;
        random(&x, 4, NULL);
        if (x) transmit(1, "BOOM", 4, NULL);
    }
}
```

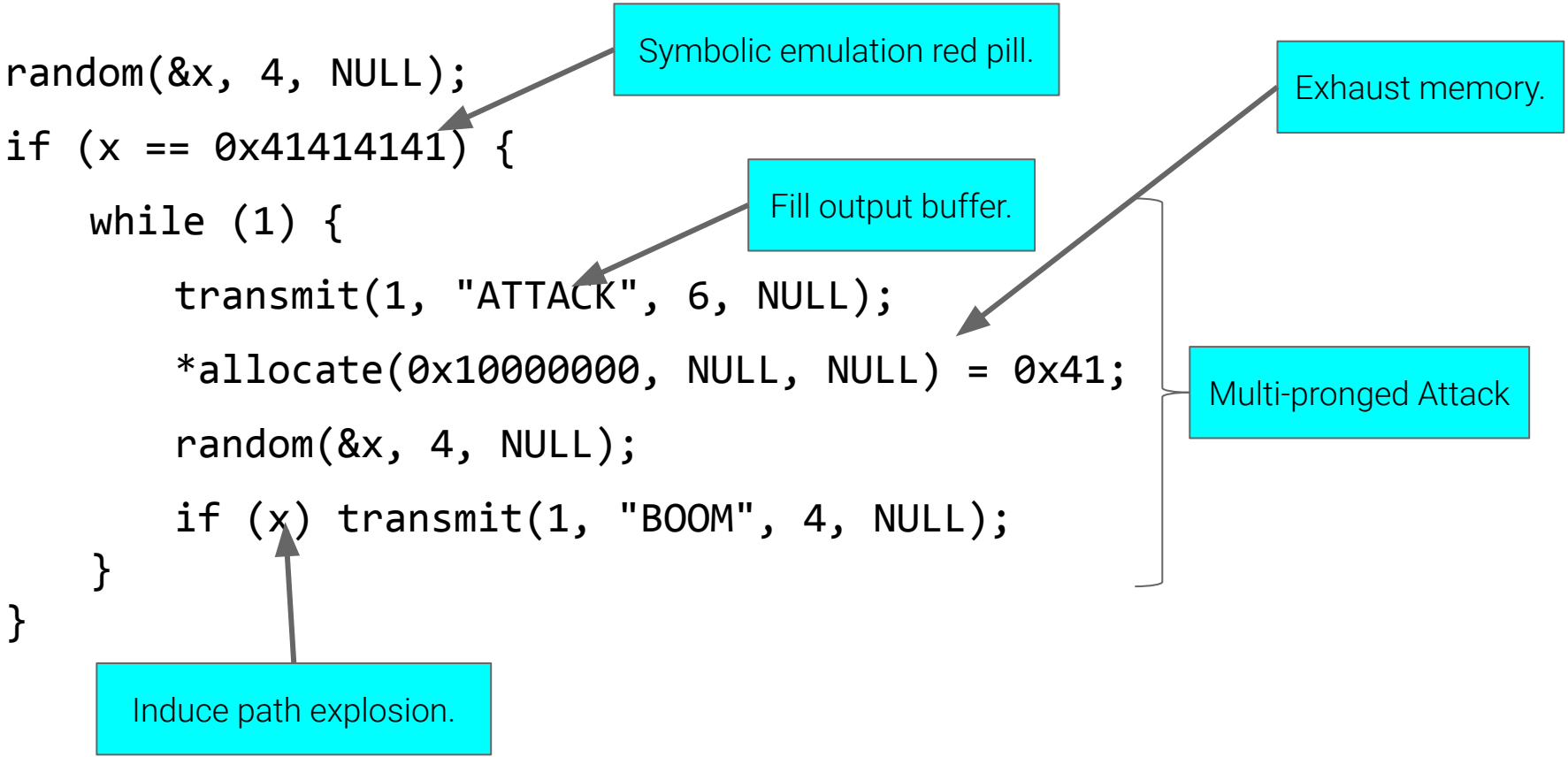


Attacking Symbolic Execution

```
random(&x, 4, NULL);
if (x == 0x41414141) {
    while (1) {
        transmit(1, "ATTACK", 6, NULL);
        *allocate(0x10000000, NULL, NULL) = 0x41;
        random(&x, 4, NULL);
        if (x) transmit(1, "BOOM", 4, NULL);
    }
}
```

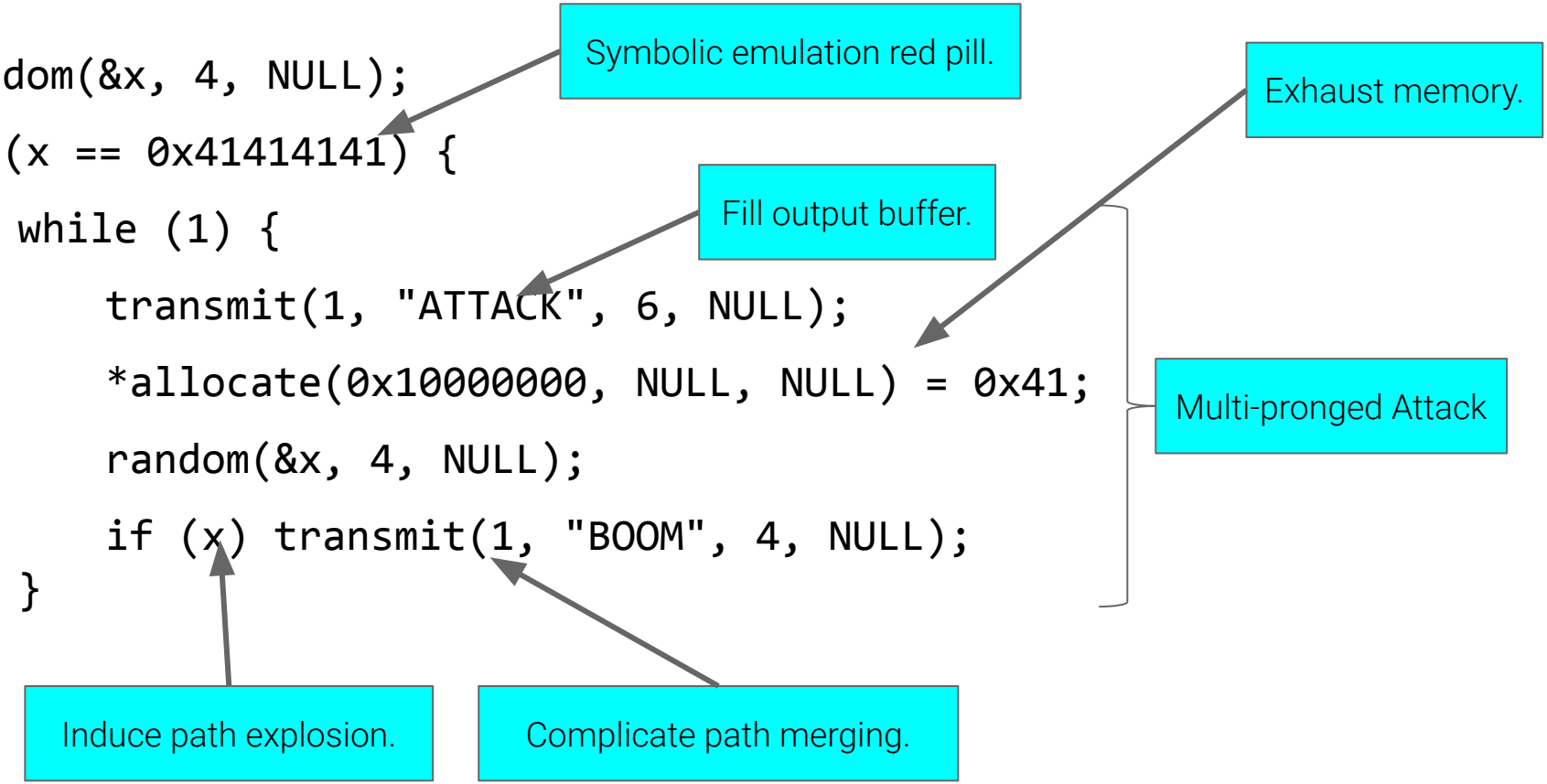


Attacking Symbolic Execution



Attacking Symbolic Execution

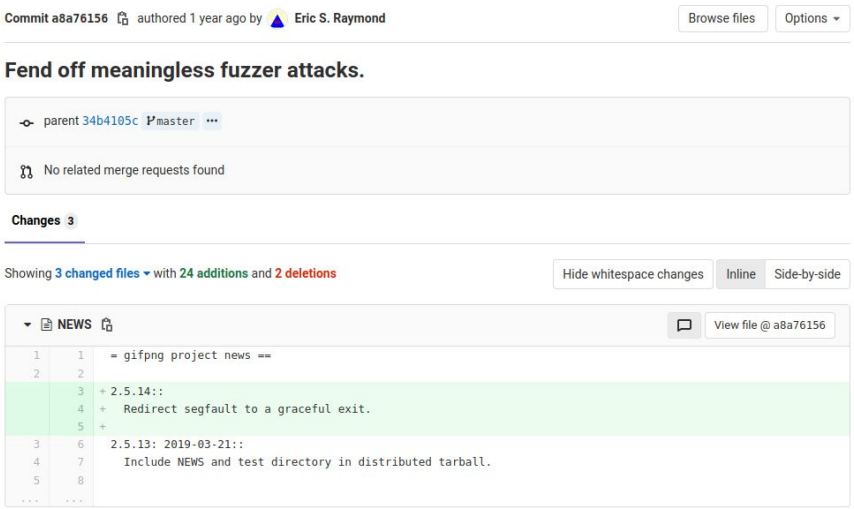
```
random(&x, 4, NULL);
if (x == 0x41414141) {
    while (1) {
        transmit(1, "ATTACK", 6, NULL);
        *allocate(0x10000000, NULL, NULL) = 0x41;
        random(&x, 4, NULL);
        if (x) transmit(1, "BOOM", 4, NULL);
    }
}
```

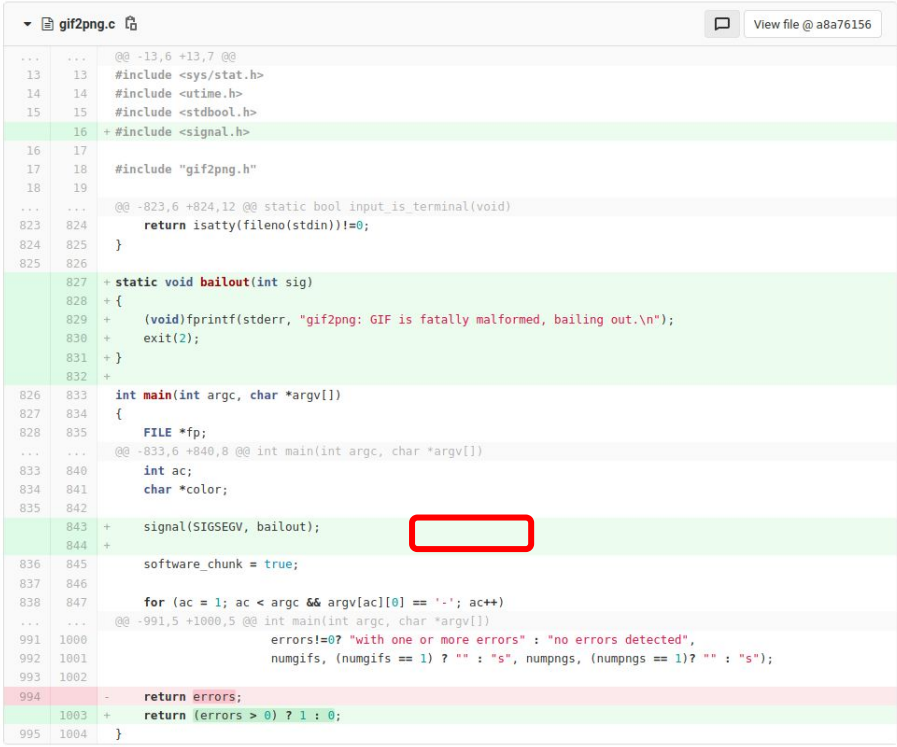


Resistance is (not) Futile

Developers have started fighting back as well!

Defensive commit in gif2png:
"Fend off meaningless fuzzer attacks".





Resilience of the larger ecosystem?

ESR: "Fend off meaningless fuzzer attacks."

Even when automated systems *are* effective, each bug found represents heavy human effort suddenly needed to fix it...

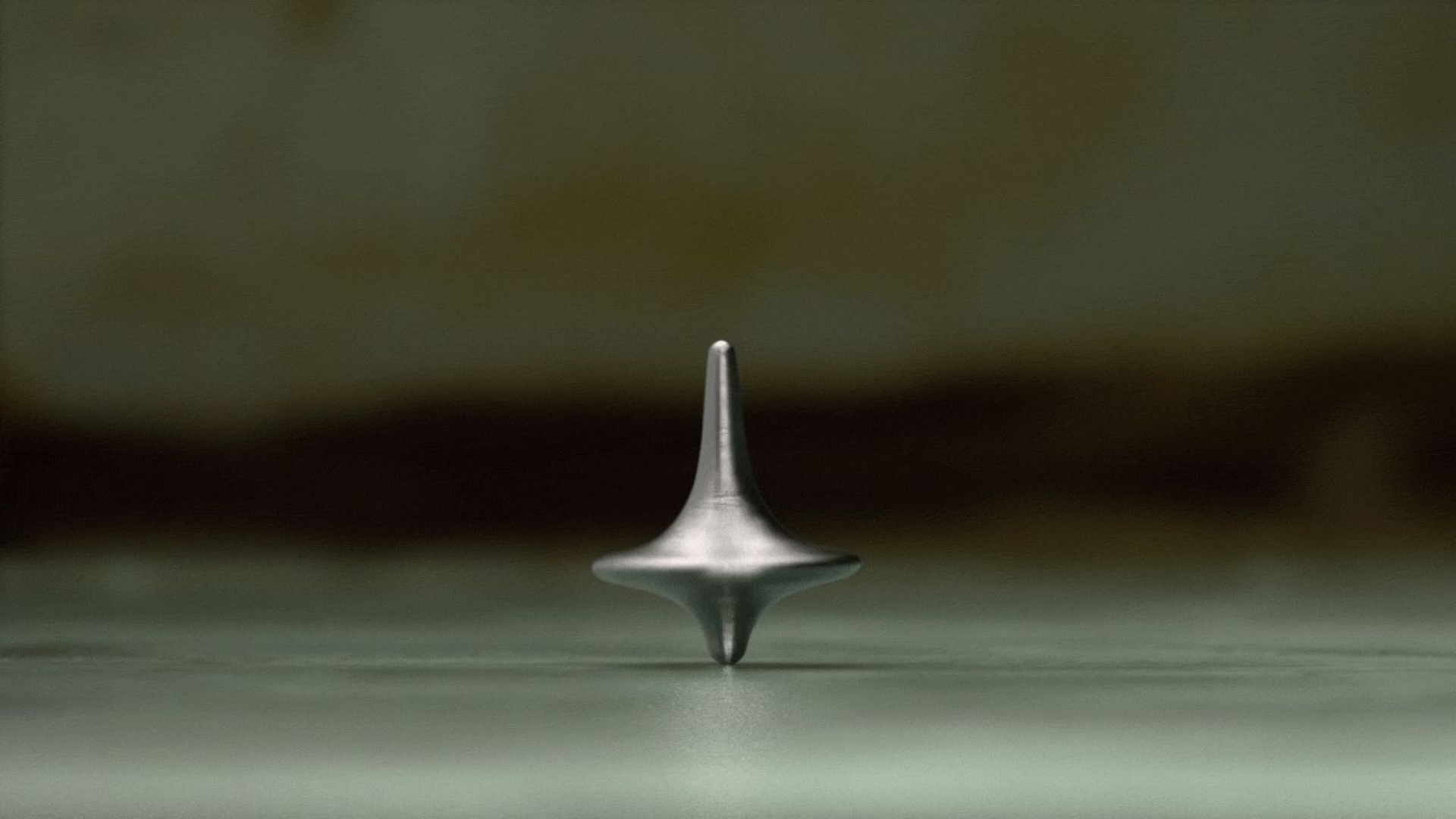
No existing technique allows for safe, automated, *targeted* program repair...

Relevant program: AMP!

Improving Resilience?

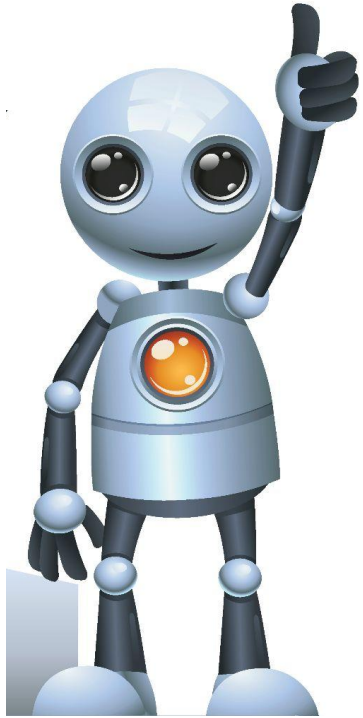
Good news: many of these issues can be addressed through engineering and thorough effort.

Bad news: thorough testing finds more "unknown unknowns" of CRS resilience failures, but we can't get guarantees...



Thank you!

Yan Shoshitaishvili
yans@asu.edu
@Zardus



Interested visitors: sefcom.asu.edu/apprenticeship.html
angr: github.com/angr (slack at angr.io/invite.html)
Learn to hack! https://pwn.college