ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
The University of Texas at Dallas

SE 4367.001

Software Testing Verification Validation and Quality Assurance

Wei Yang

Department of Computer Science
University of Texas at Dallas

Outline

- Overview
- Course Content

Outline

- Overview
- Course Content

Course Info

- Instructor: Wei Yang
- Office: ECSS 4.225
- Email: wei.yang@utdallas.edu
- Homepage: http://youngwei.com/

- TA: TBD

- Course web page
  - http://youngwei.com/page/CS4367-001-22S/index.html

$ Whoami

SHANGHAI JIAO TONG UNIVERSITY
1896

N
C
S

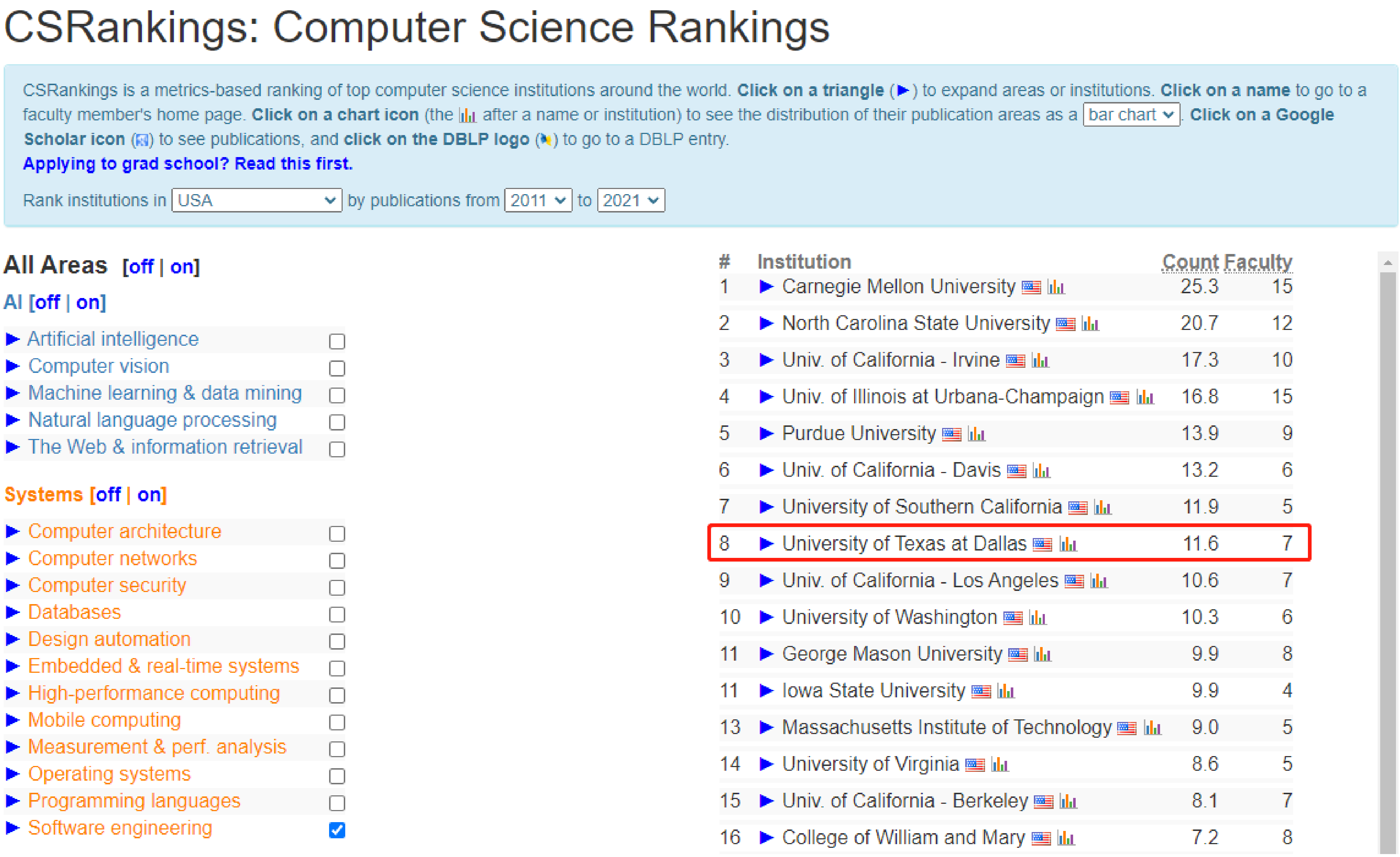
THE UNIVERSITY OF CALIFORNIA BERKELEY
LET THERE BE LIGHT
1868

THE UNIVERSITY OF TEXAS AT DALLAS
UTD
EST. 1969

Research Interests

- **MobileSecurity** (WHYPER, Pluto, AppContext[1][2][3], Telemade, MRV, CLAP[1], EnMobile, MalScan)
- **Automated Testing** (ORBIT, WCTester[1][2], NMTtest[1], REINAM)
- **SE/Security for Machine Learning** (PerInv, MRV, Telemade, NMTtest)
- **ML/NLP for SE/Security** (WHYPER, Pluto, CLAP, SemRegex[1], REINAM)
- **IoT Security** (iRuler)

Software Engineering in UT Dallas



CSRankings: Computer Science Rankings

CSRankings is a metrics-based ranking of top computer science institutions around the world. **Click on a triangle** (▶) to expand areas or institutions. **Click on a name** to go to a faculty member's home page. **Click on a chart icon** (the 📊 after a name or institution) to see the distribution of their publication areas as a bar chart. **Click on a Google Scholar icon** () to see publications, and **click on the DBLP logo** () to go to a DBLP entry.

Applying to grad school? Read this first.

Rank institutions in USA by publications from 2011 to 2021

All Areas [off | on]

AI [off | on]

- ▶ Artificial intelligence □
- ▶ Computer vision □
- ▶ Machine learning & data mining □
- ▶ Natural language processing □
- ▶ The Web & information retrieval □

Systems [off | on]

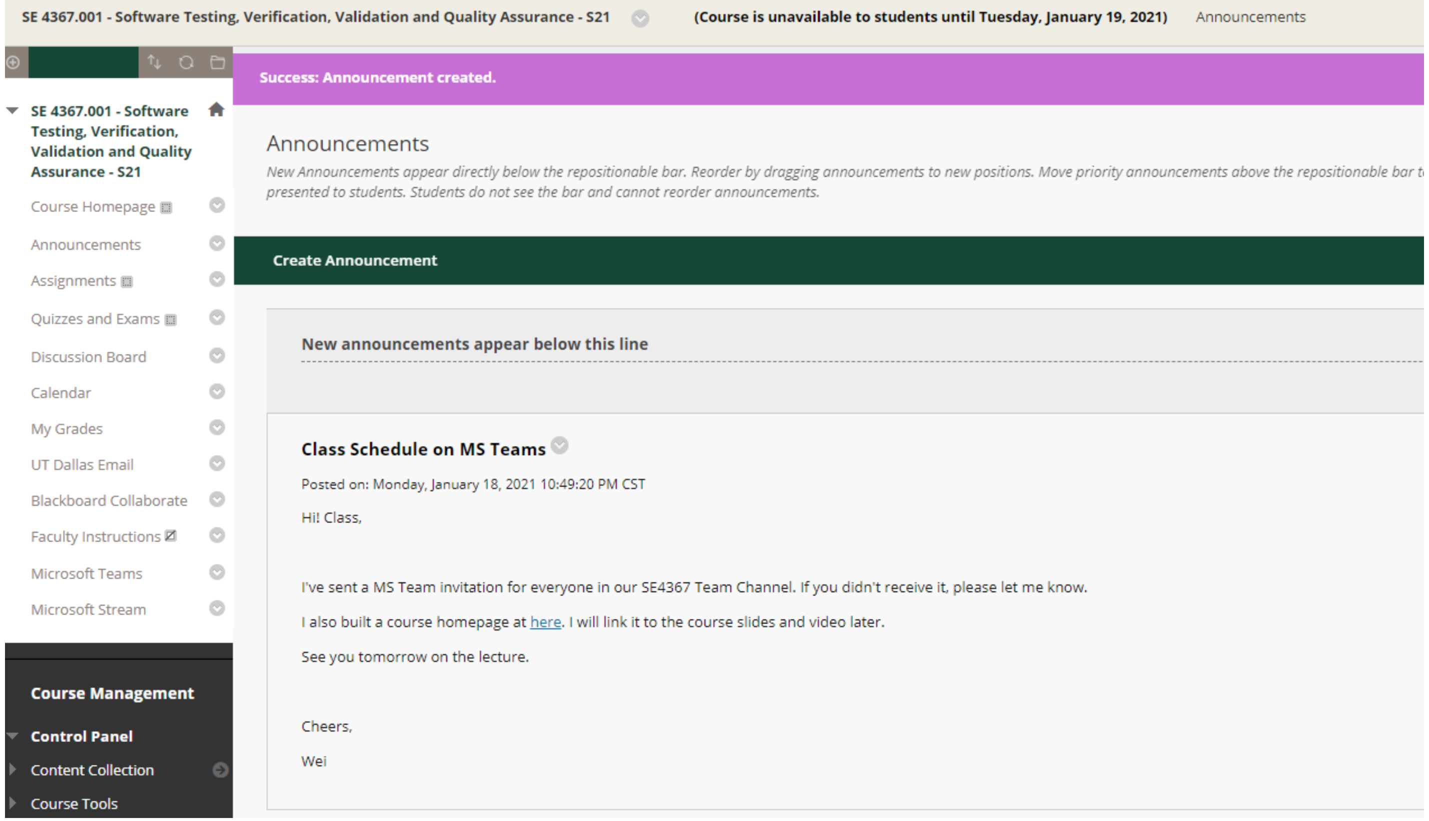
- ▶ Computer architecture □
- ▶ Computer networks □
- ▶ Computer security □
- ▶ Databases □
- ▶ Design automation □
- ▶ Embedded & real-time systems □
- ▶ High-performance computing □
- ▶ Mobile computing □
- ▶ Measurement & perf. analysis □
- ▶ Operating systems □
- ▶ Programming languages □
- ▶ Software engineering ☑

#	Institution	Count	Faculty
1	▶ Carnegie Mellon University	25.3	15
2	▶ North Carolina State University	20.7	12
3	▶ Univ. of California - Irvine	17.3	10
4	▶ Univ. of Illinois at Urbana-Champaign	16.8	15
5	▶ Purdue University	13.9	9
6	▶ Univ. of California - Davis	13.2	6
7	▶ University of Southern California	11.9	5
8	▶ University of Texas at Dallas	11.6	7
9	▶ Univ. of California - Los Angeles	10.6	7
10	▶ University of Washington	10.3	6
11	▶ George Mason University	9.9	8
11	▶ Iowa State University	9.9	4
13	▶ Massachusetts Institute of Technology	9.0	5
14	▶ University of Virginia	8.6	5
15	▶ Univ. of California - Berkeley	8.1	7
16	▶ College of William and Mary	7.2	8

Electronic communication

- https://elearning.utdallas.edu : announcements

- MS Teams: questions, answers



Success: Announcement created.

Announcements

New Announcements appear directly below the repositionable bar. Reorder by dragging announcements to new positions. Move priority announcements above the repositionable bar t presented to students. Students do not see the bar and cannot reorder announcements.

New announcements appear below this line

Class Schedule on MS Teams

Posted on: Monday, January 18, 2021 10:49:20 PM CST

Hi! Class,

I've sent a MS Team invitation for everyone in our SE4367 Team Channel. If you didn't receive it, please let me know.

I also built a course homepage at here. I will link it to the course slides and video later.

See you tomorrow on the lecture.

Cheers,

Wei

Textbook and Readings

- This course will mainly be a walk through for https://www.fuzzingbook.org/.

- Some (text)books recommended

- Reading linked from Schedule (Provided later)

- Course Website: http://youngwei.com/page/CS4367-001-22S/index.html

Grading

- Mid-term Exam (30%)
- Assignments (20%)
- Online Discussion & Class Participation (10%)
- Final Exam(40%)

- For fairness, we REPORT all cheating
  - Please **avoid copy-pasting** as much as possible. For any material (especially graphics and anything included by copy-pasting) not created by you but included in your deliverable, you **must acknowledge the source on the same page**.

Outline

- Overview
- Course Content
- Sample presentation by the instructor

The facts

- Only 32% of software projects are considered successful
  - (full featured, on time, on budget)

- Software failures cost the US economy $59.5 billion dollars every year [NIST 2002 Report]

- On average, 1-5 bugs per KLOC (thousand lines of code) In mature software (more than 10 bugs in prototypes)

✷ 35MLOC
✷ 63K known bugs at the time of release
✷ 2 bugs per KLOC

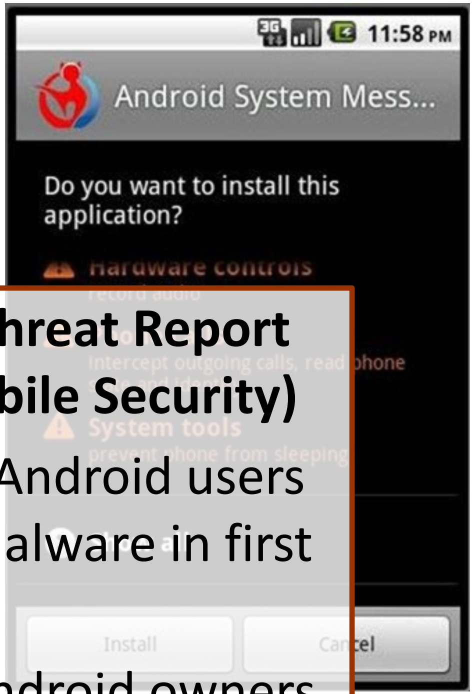
Testing

Testing

- Caused due to numeric overflow error
  - Attempt to fit 64-bit format data in 16-bit space
- Cost
  - $100M’s for loss of mission
  - Multi-year setback to the Ariane program
- Read more at http://www.around.com/ariane.html

Security Vulnerabilities

- Exploits of errors in programs
- Widespread problem
  - Moonlight Maze (1998)
  - Code Red (2001)
  - Titan Rain (2003)
  - Stuxnet Worm
- Getting worse …



Do you want to install this application?

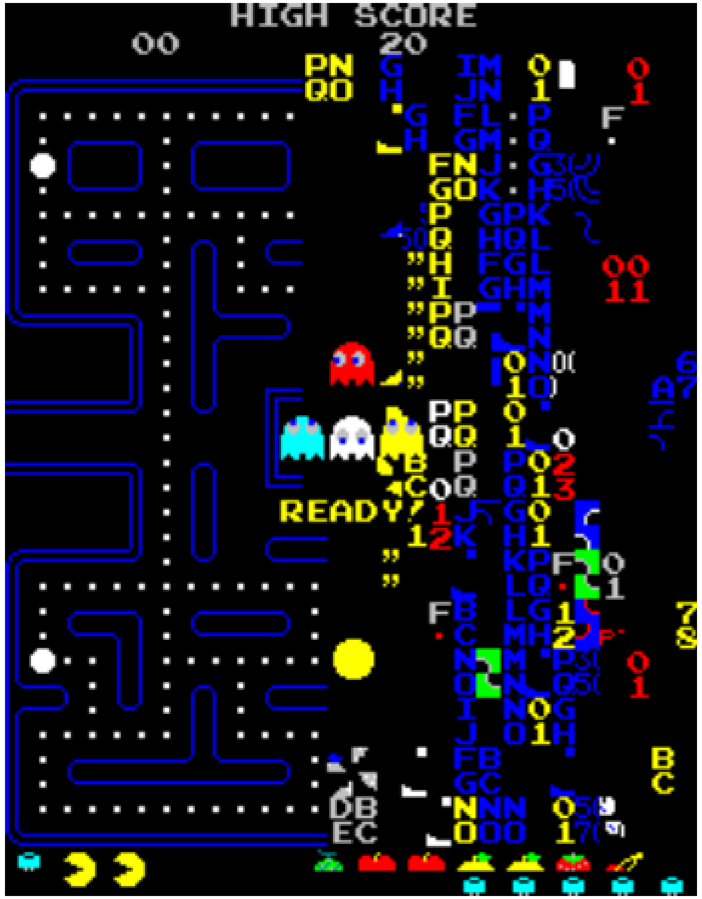
2011 Mobile Threat Report (Lookout™ Mobile Security)

- 0.5-1 million Android users affected by malware in first half of 2011
- 3 out of 10 Android owners likely to face web-based threat each year
- Attackers using increasingly sophisticated ways to steal data and money

A few more examples

Pac-Man (1980)

- Should always have no ending
- Has “Split Screen” at level 256
- Cause: Integer overflow
- 8 bits: maximum representable value



1	1	1	1	1	1	1	1	+	0	0	0	0	0	0	0	1	=	1	0	0	0	0	0	0	0	0

255 + 1 = 0

A few more examples

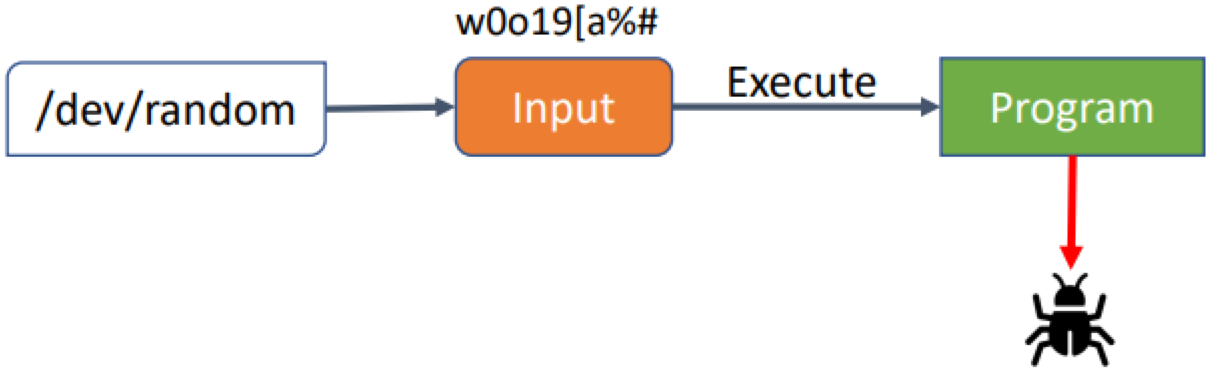
- Mars Climate Orbiter (1998)
  - Sent to Mars to relay signal from Mars
- Lander
  - Smashed to the planet
- Cause: Failing to convert between different metric standards
  - Software that calculated the total impulse presented results in pound-seconds
  - The system using these results expected its inputs to be in newton-seconds

A few more examples

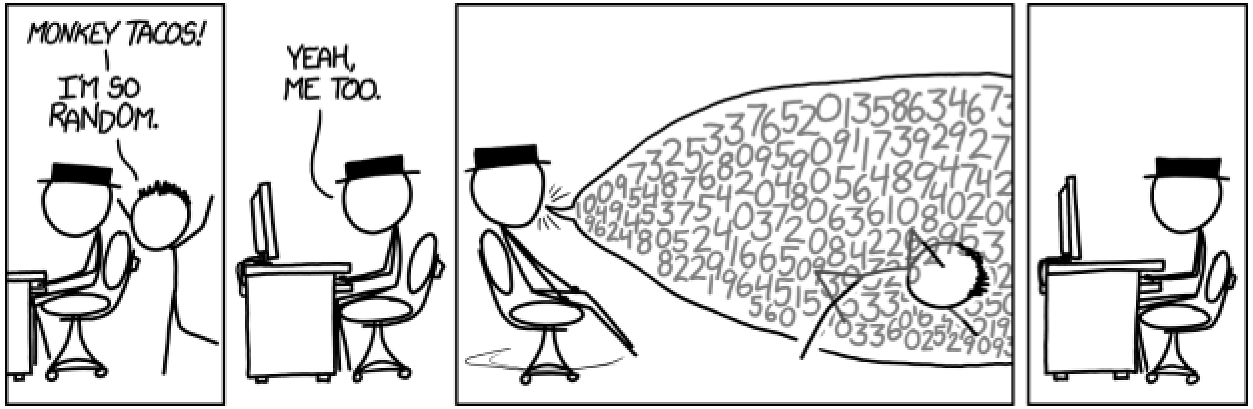
- USS Yorktown (1997)
  - Left dead in the water for 3 hours
- Cause: Divide by zero error

$$\frac{\text{Number}}{0} =$$

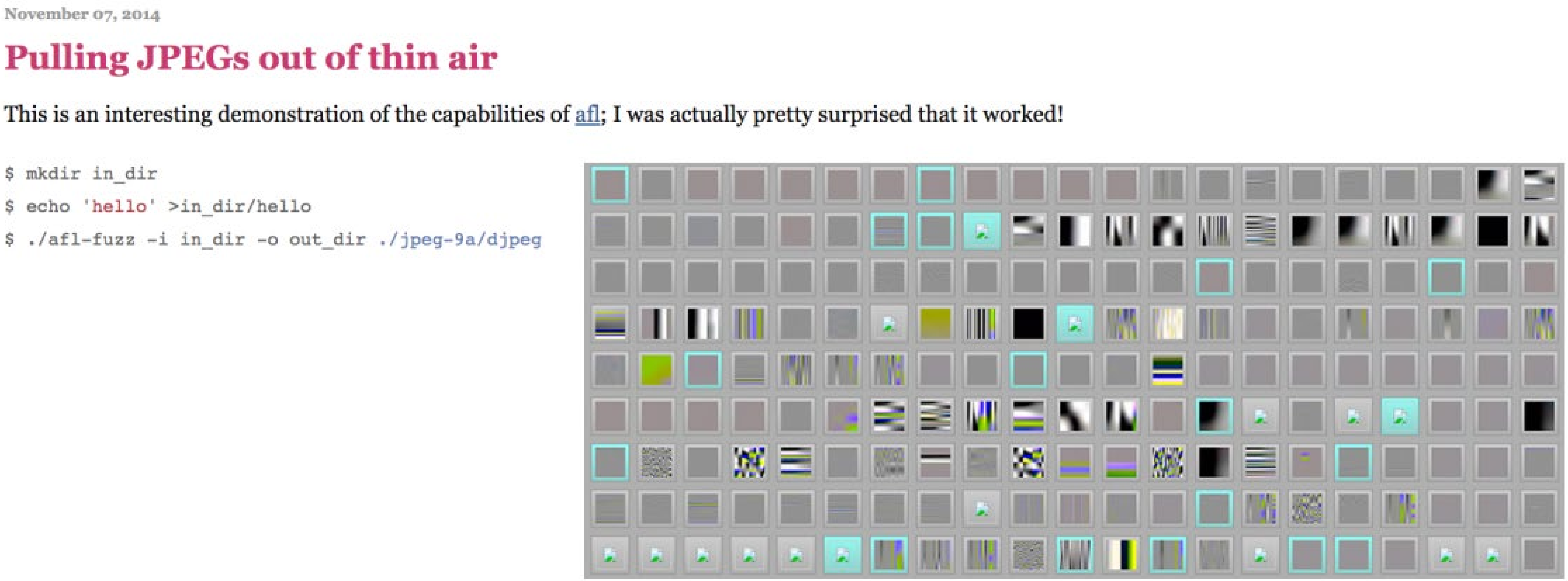
Fuzzing



Bugs = Crashes (segfaults, aborts, etc.)



Fuzzing



November 07, 2014

Pulling JPEGs out of thin air

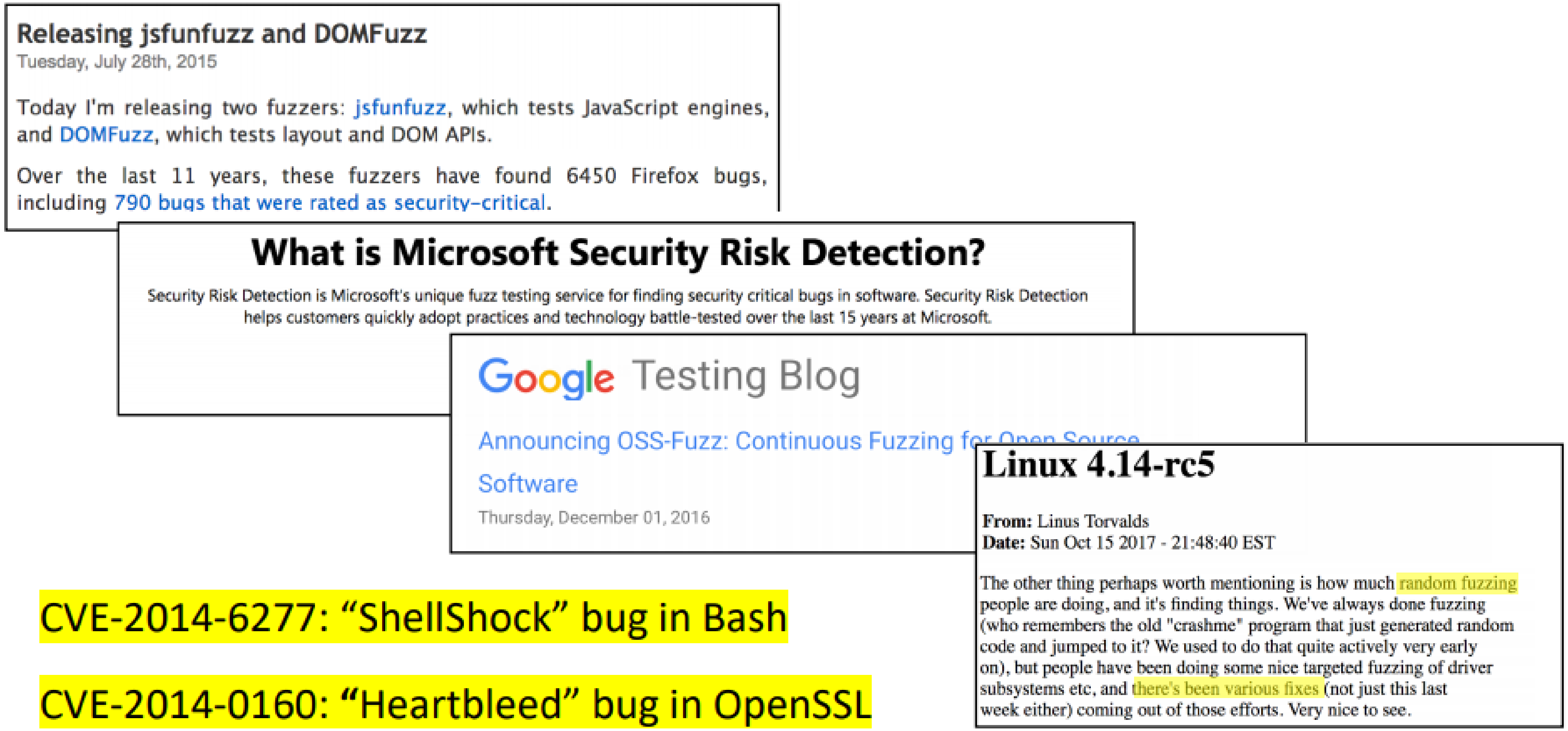
This is an interesting demonstration of the capabilities of afl; I was actually pretty surprised that it worked!

```
$ mkdir in_dir
$ echo 'hello' >in_dir/hello
$ ./afl-fuzz -i in_dir -o out_dir ./jpeg-9a/djpeg
```

Fuzzers

- AFLFast [CCS 2016]
- Driller [NDSS 2016]
- AFLGo [CCS 2017]
- Vuzzer [NDSS 2017]
- Steelix [FSE 2017]
- SlowFuzz [CCS 2017]
- PerfFuzz [ISSTA 2018]
- FairFuzz [ASE 2018]
- Angora [IEEE S&P 2018]
- T-Fuzz [IEEE S&P 2018]
- NEUZZ [IEEE S&P 2019]
- Nautilus [NDSS 2019]
- Redqueen [NDSS 2019]
- Superion [ICSE 2019]
- MOPT [Usenix Sec 2019]
- GRIMOIRE [Usenix Sec 2019]
- MemFuzz [ICST 2019]
- Zest [ISSTA 2019]
- DifFuzz [ICSE 2019]
- AFLSmart [IEEE TSE 2019]
- FuzzChick [OOPSLA 2019]
- …

Fuzzing for Security



Releasing jsfunfuzz and DOMFuzz

Tuesday, July 28th, 2015

Today I'm releasing two fuzzers: jsfunfuzz, which tests JavaScript engines, and DOMFuzz, which tests layout and DOM APIs.

Over the last 11 years, these fuzzers have found 6450 Firefox bugs, including 790 bugs that were rated as security-critical.

What is Microsoft Security Risk Detection?

Security Risk Detection is Microsoft's unique fuzz testing service for finding security critical bugs in software. Security Risk Detection helps customers quickly adopt practices and technology battle-tested over the last 15 years at Microsoft.

Google Testing Blog

Announcing OSS-Fuzz: Continuous Fuzzing for Open Source Software

Thursday, December 01, 2016

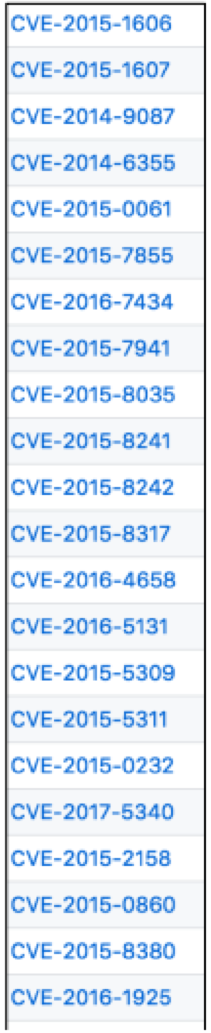
Linux 4.14-rc5

From: Linus Torvalds
Date: Sun Oct 15 2017 - 21:48:40 EST

The other thing perhaps worth mentioning is how much random fuzzing people are doing, and it's finding things. We've always done fuzzing (who remembers the old "crashme" program that just generated random code and jumped to it? We used to do that quite actively very early on), but people have been doing some nice targeted fuzzing of driver subsystems etc, and there's been various fixes (not just this last week either) coming out of those efforts. Very nice to see.

CVE-2014-6277: “ShellShock” bug in Bash

CVE-2014-0160: “Heartbleed” bug in OpenSSL



CVE-2015-1606
CVE-2015-1607
CVE-2014-9087
CVE-2014-6355
CVE-2015-0061
CVE-2015-7855
CVE-2016-7434
CVE-2015-7941
CVE-2015-8035
CVE-2015-8241
CVE-2015-8242
CVE-2015-8317
CVE-2016-4658
CVE-2016-5131
CVE-2015-5309
CVE-2015-5311
CVE-2015-0232
CVE-2017-5340
CVE-2015-2158
CVE-2015-0860
CVE-2015-8380
CVE-2016-1925

Fairness

Responses from Computing Researchers to HUD's Implementation of the Fair Housing Act's Disparate Impact Standard

January 8th, 2020 / in Announcements, CCC, policy, research horizons, Research News / by Helen Wright

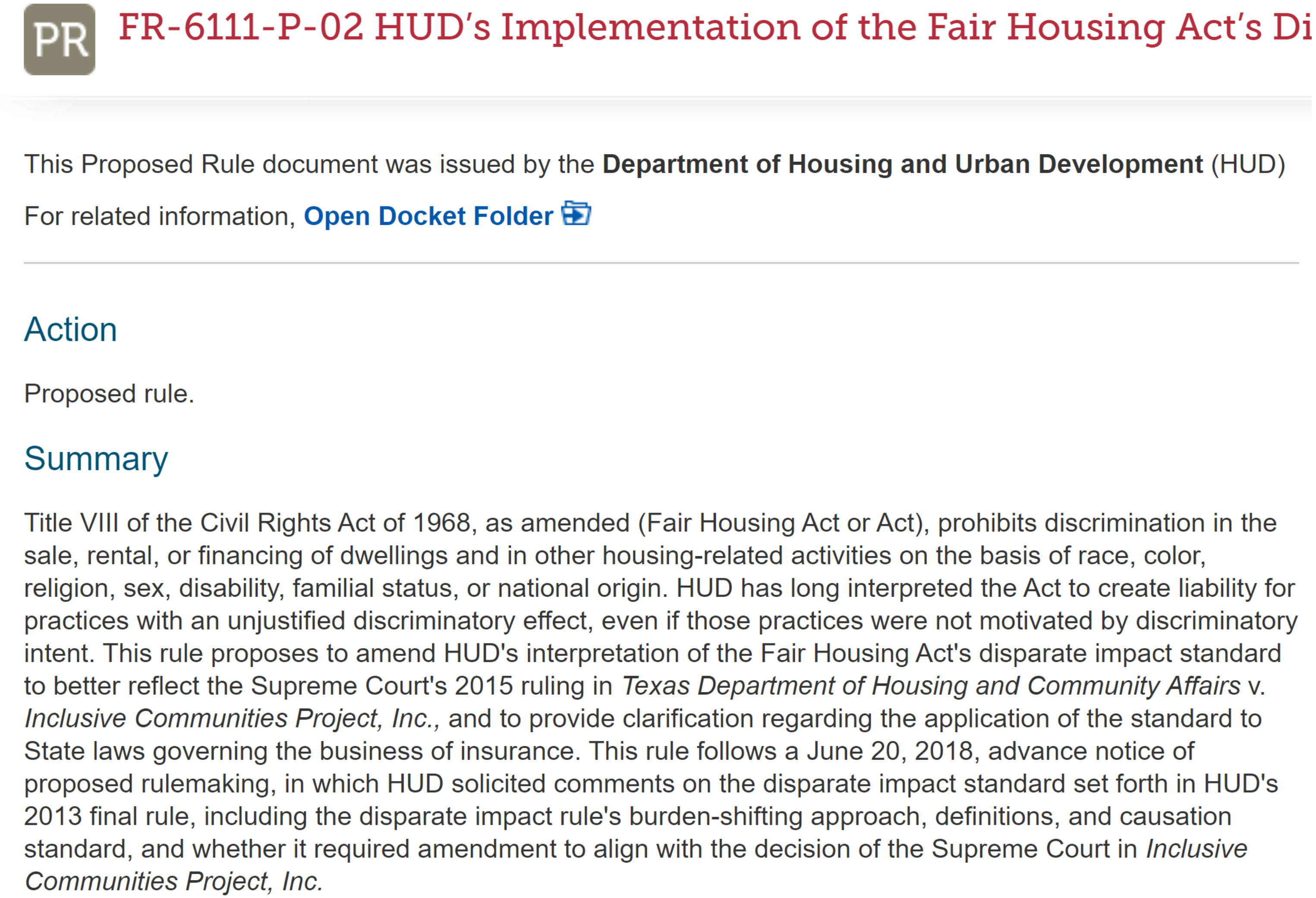
The following blog post is from Computing Community Consortium (CCC) Vice Chair Elizabeth Bradley (University of Colorado Boulder) and CCC Executive Council member Suresh Venkatasubramanian (University of Utah).

Algorithmic bias can be insidious, making it all but impossible to pinpoint factors that contribute to discrimination. This is particularly concerning in the context of high-stakes decisions. The new Department of Housing and Urban Development (HUD) guidelines around the use of algorithms to aid in housing decisions are an example of this. This HUD proposal acknowledges the existence of algorithmic bias but would shift much of the burden of proof to demonstrate discriminatory behavior back onto the plaintiffs, using standards for algorithmic transparency and explainability that seem unmoored from extant science about what we can hope to extract from algorithmic decision pipelines. Among other things, this would allow landlords and lenders to deflect lawsuits with an overly naive statistical approach, looking at individual factors rather than taking them in combination and thereby ignoring the potential collective effect of many lenders using the same third-party algorithm. Writing in Forbes, Elizabeth Fernandez suggests that this could undermine the Fair Housing Act.

Computing researchers who study these issues have submitted formal responses to the public call for comments regarding these new guidelines. These included a coordinated response by members of the GRAIL network, a new initiative led by the Center for Democracy and Technology (CDT) and the R Street Initiative. GRAIL's goal is to connect technical and policy experts to inform discussions around technology policy in Washington and provide deep, rapid responses to questions of tech policy. Their response, which was led by Natasha Duarte at CDT and involved CCC Council member Suresh Venkatasubramanian, details how the different components of the

Fairness – Regulation & Rules

- https://www.regulations.gov/document?D=HUD-2019-0067-0001



PR FR-6111-P-02 HUD's Implementation of the Fair Housing Act's Di

This Proposed Rule document was issued by the **Department of Housing and Urban Development** (HUD)

For related information, **Open Docket Folder**

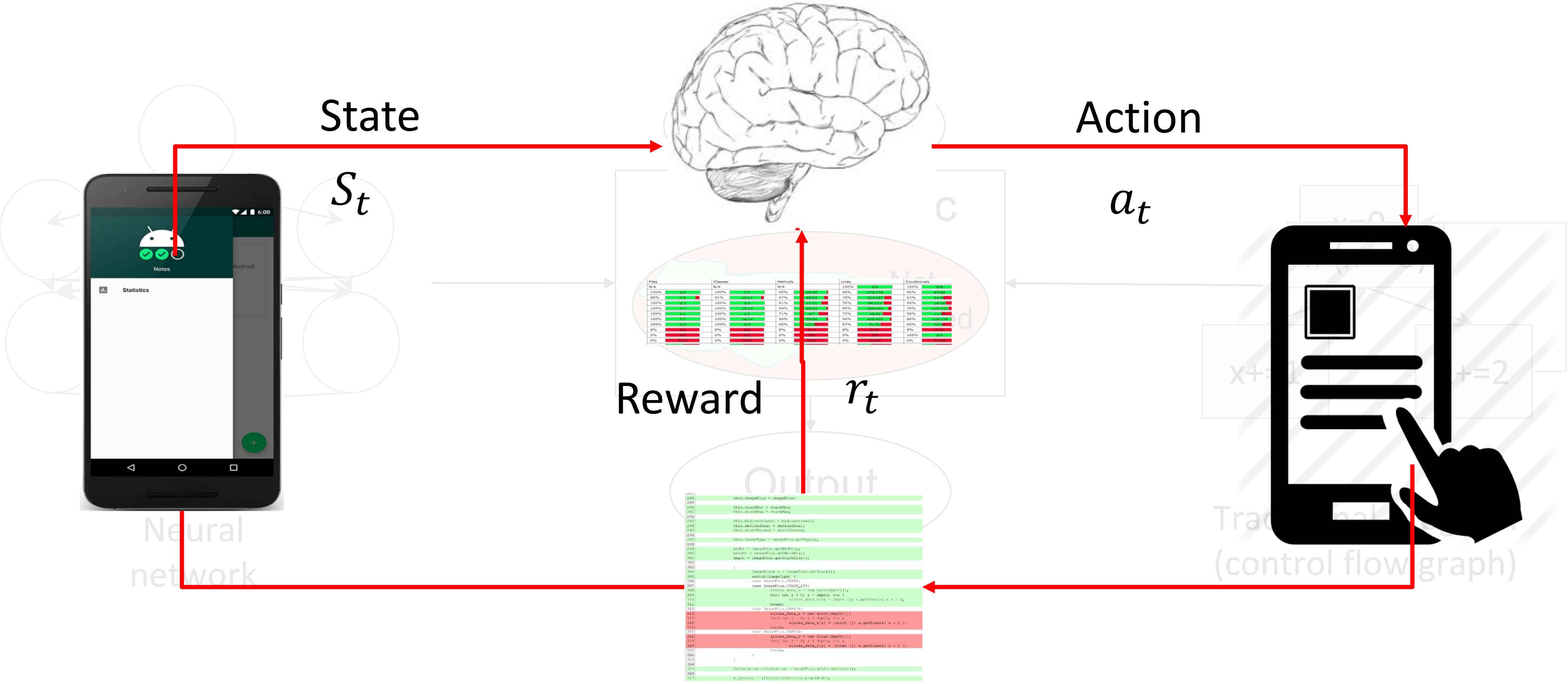
Action

Proposed rule.

Summary

Title VIII of the Civil Rights Act of 1968, as amended (Fair Housing Act or Act), prohibits discrimination in the sale, rental, or financing of dwellings and in other housing-related activities on the basis of race, color, religion, sex, disability, familial status, or national origin. HUD has long interpreted the Act to create liability for practices with an unjustified discriminatory effect, even if those practices were not motivated by discriminatory intent. This rule proposes to amend HUD's interpretation of the Fair Housing Act's disparate impact standard to better reflect the Supreme Court's 2015 ruling in *Texas Department of Housing and Community Affairs* v. *Inclusive Communities Project, Inc.*, and to provide clarification regarding the application of the standard to State laws governing the business of insurance. This rule follows a June 20, 2018, advance notice of proposed rulemaking, in which HUD solicited comments on the disparate impact standard set forth in HUD's 2013 final rule, including the disparate impact rule's burden-shifting approach, definitions, and causation standard, and whether it required amendment to align with the decision of the Supreme Court in *Inclusive Communities Project, Inc.*

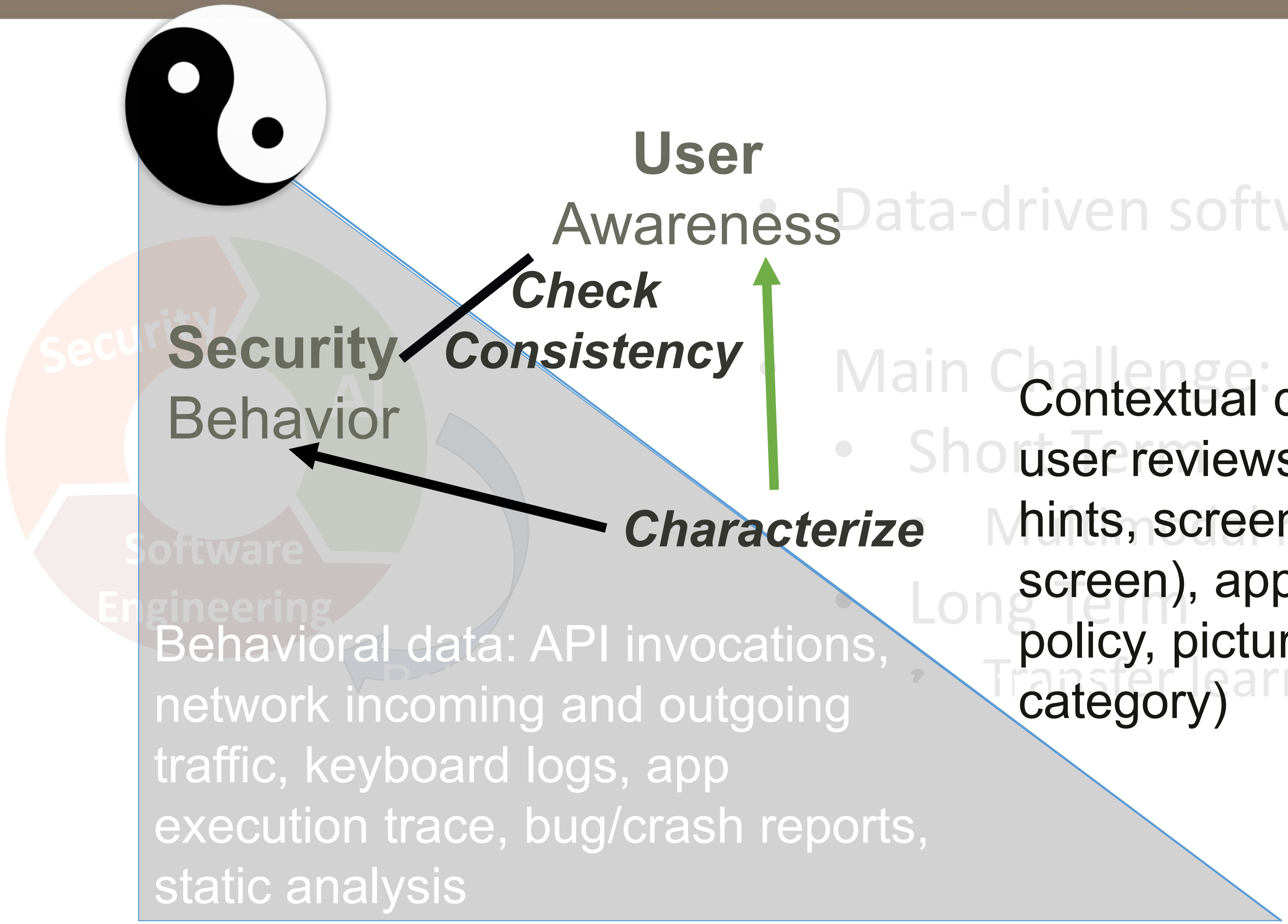
UI testing agent with reinforcement learning



An Empirical Study of Android Test Generation Tools in Industrial Cases

Wang et al. ASE 2018

Yin-Yang view of data-driven app testing

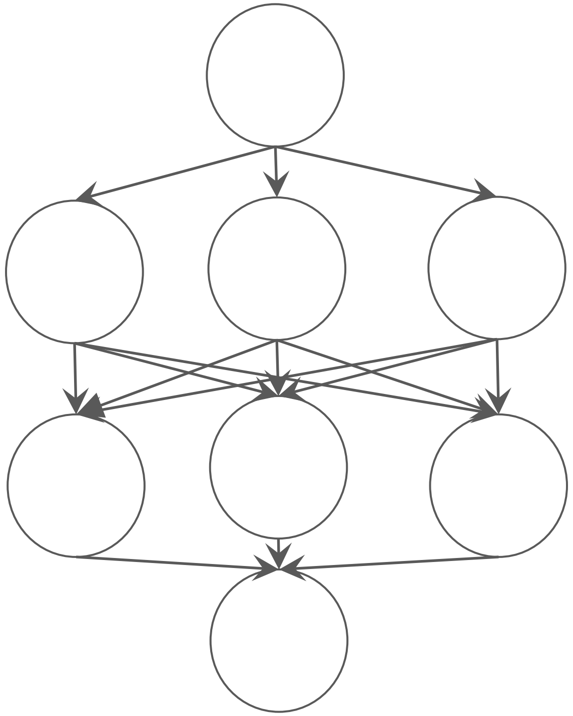


Behavioral data: API invocations, network incoming and outgoing traffic, keyboard logs, app execution trace, bug/crash reports, static analysis

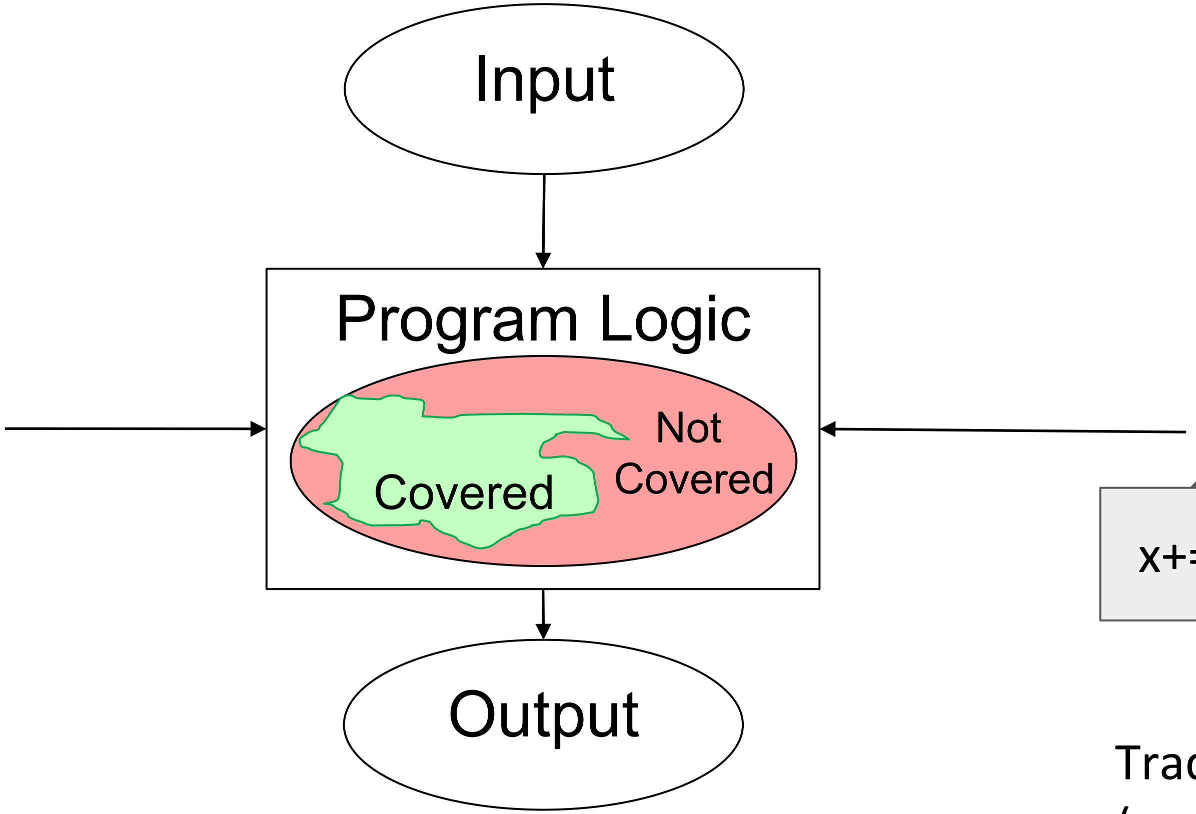
Contextual data: user usage data, user reviews, UI screen (labels, hints, screen, buttons, sequence of screen), app descriptions, privacy policy, pictures/videos, tags (app category)

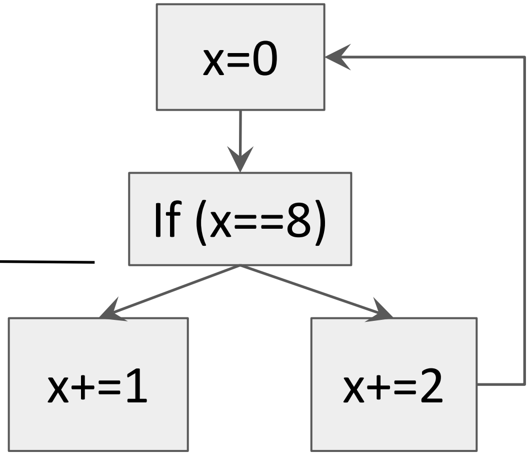
- Data-driven software engineering
- Main Challenge: Lack of labeled data
  - Short Term
    - Multimodal representation learning
  - Long Term
    - Transfer learning

Testing criterion for machine learning



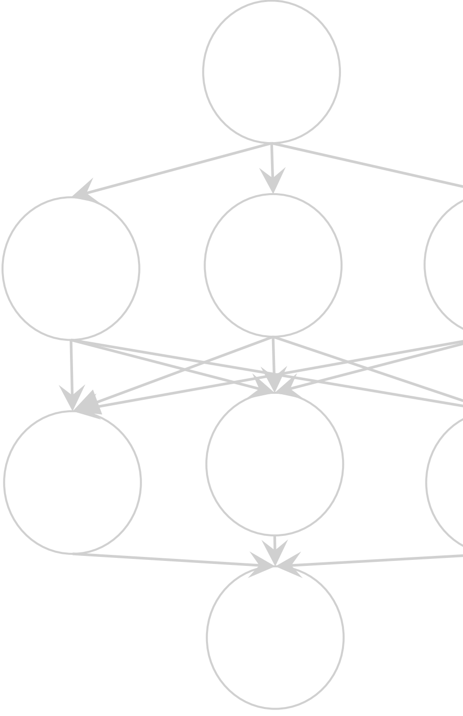
Neural network



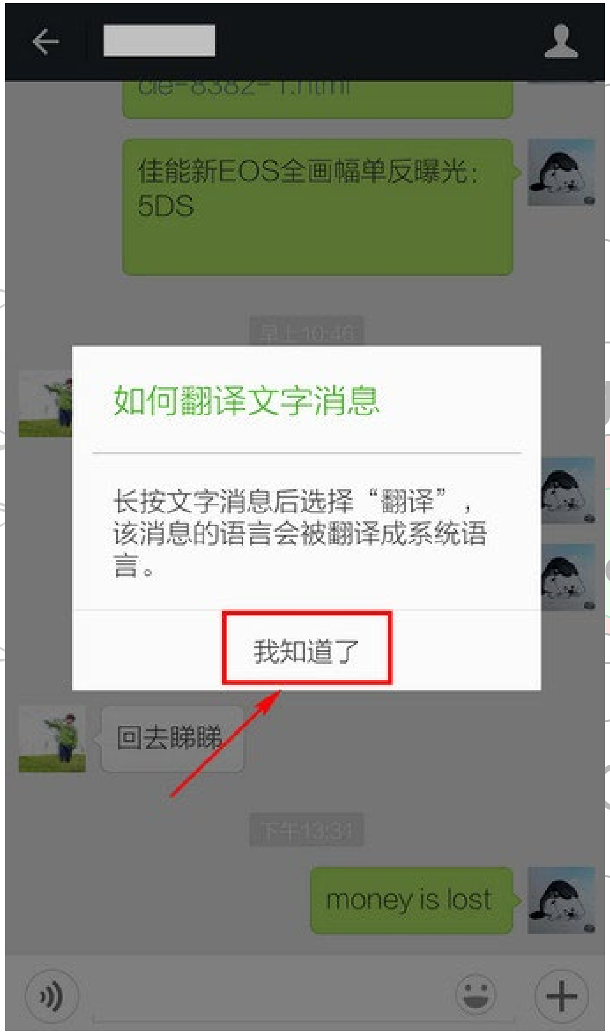


Traditional program (control flow graph)

Testing criterion for NLP



Neural network



[redacted]

cle-8382-1.html

佳能新EOS全画幅单反曝光：5DS

早上10:46

如何翻译文字消息

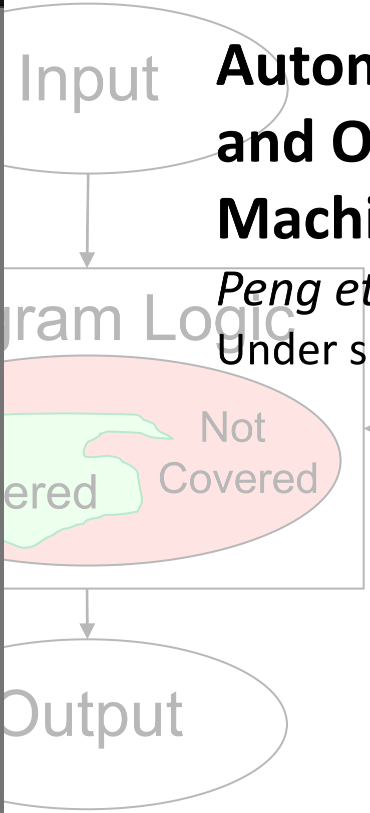
长按文字消息后选择“翻译”，该消息的语言会被翻译成系统语言。

我知道了

回去睇睇

下午13:31

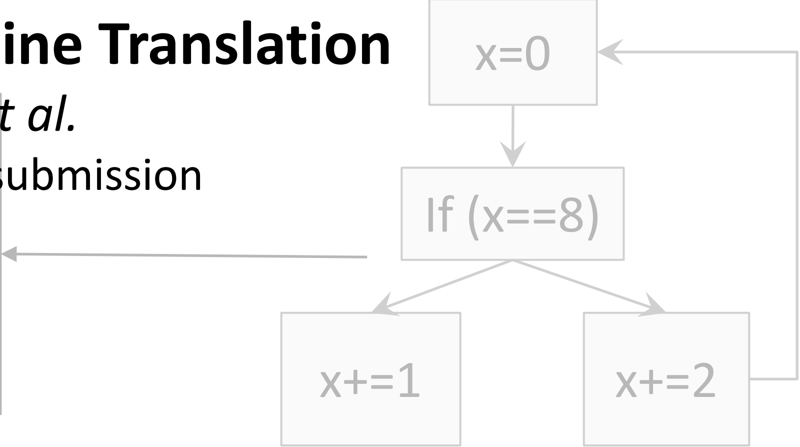
money is lost



Automatic Detection of Under- and Over-Translation in Neural Machine Translation

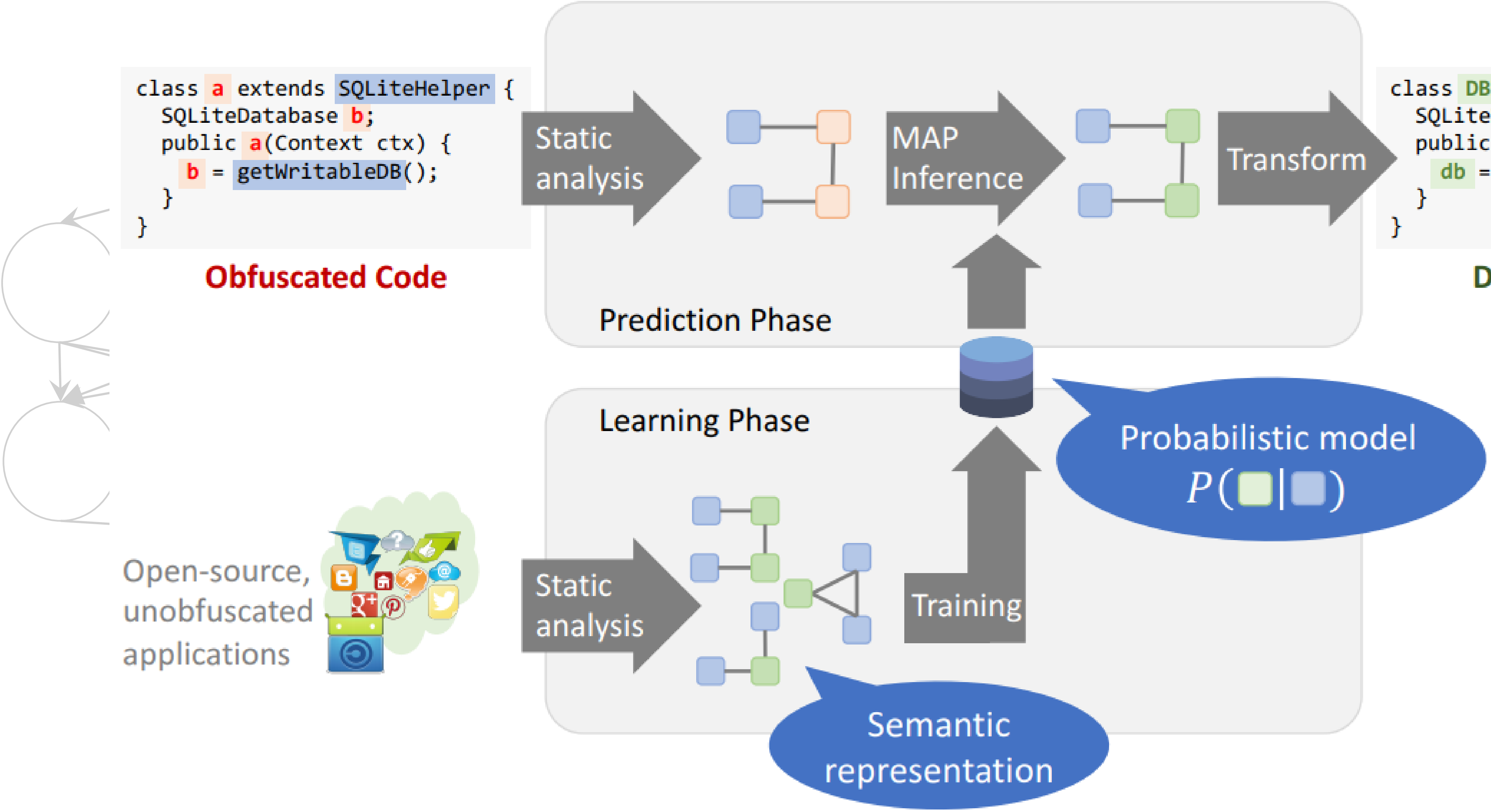
Peng et al.

Under submission



Traditional program (control flow graph)

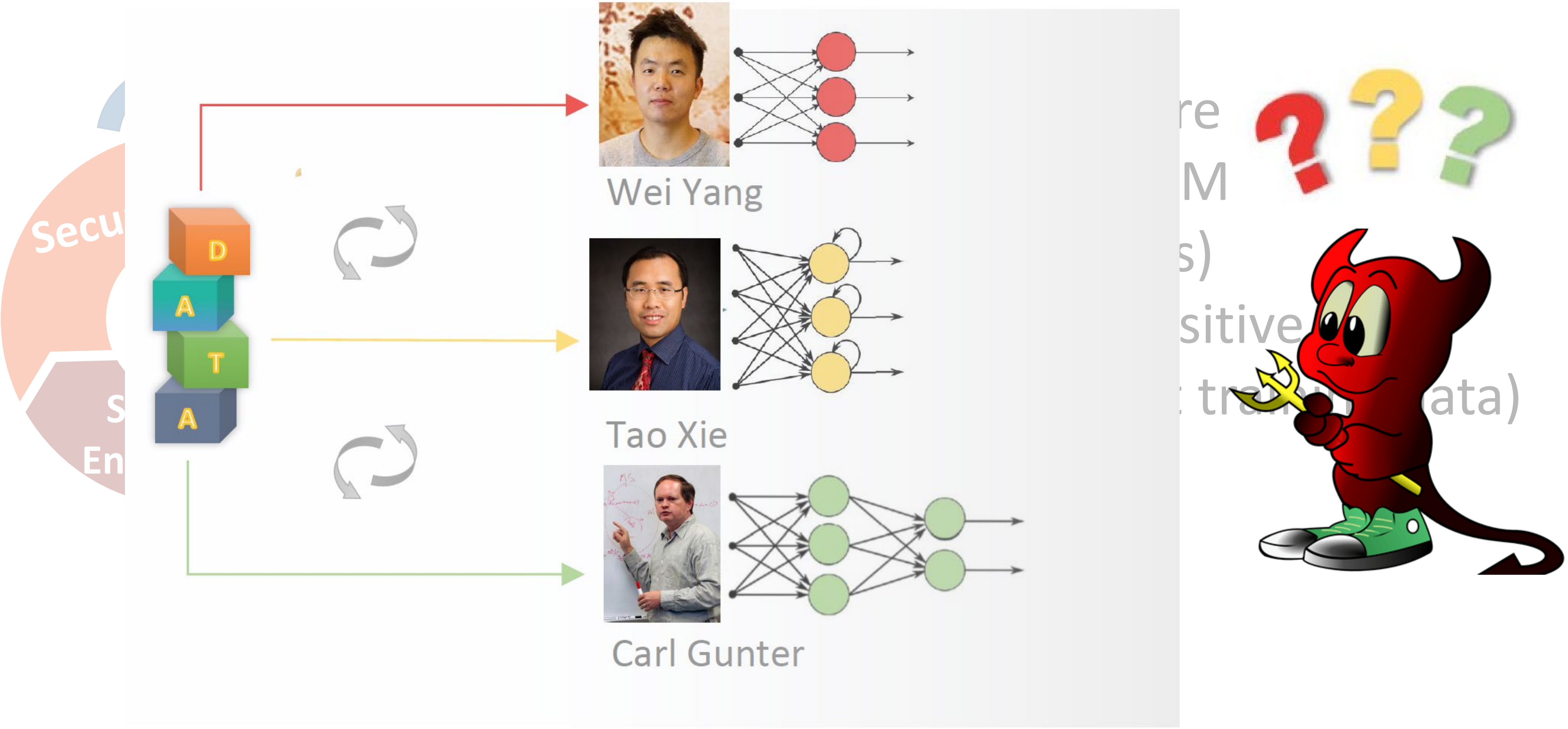
Code Obfuscation/De-obfuscation/Transformation



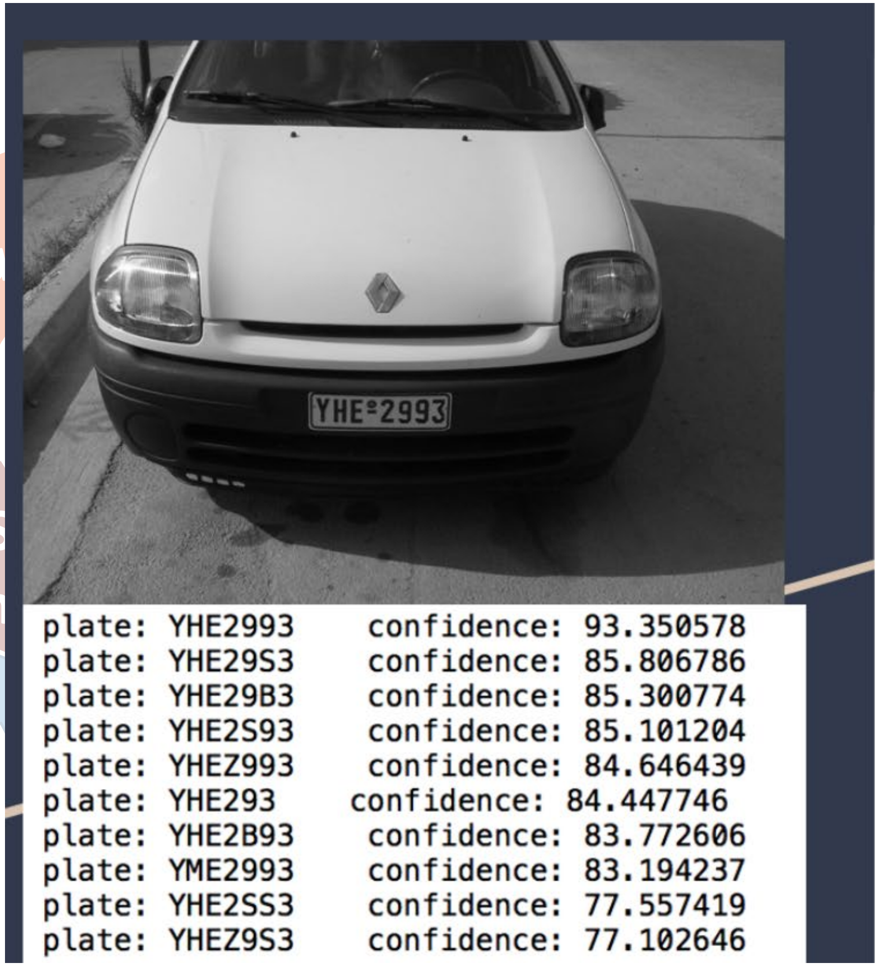
```
class a extends SQLiteHelper {
  SQLiteDatabase b;
  public a(Context ctx) {
    b = getWritableDB();
  }
}
```

```
class DBHelper extends SQLiteHelper{
  SQLiteDatabase db;
  public DBHelper(Context ctx) {
    db = getWritableDB();
  }
}
```

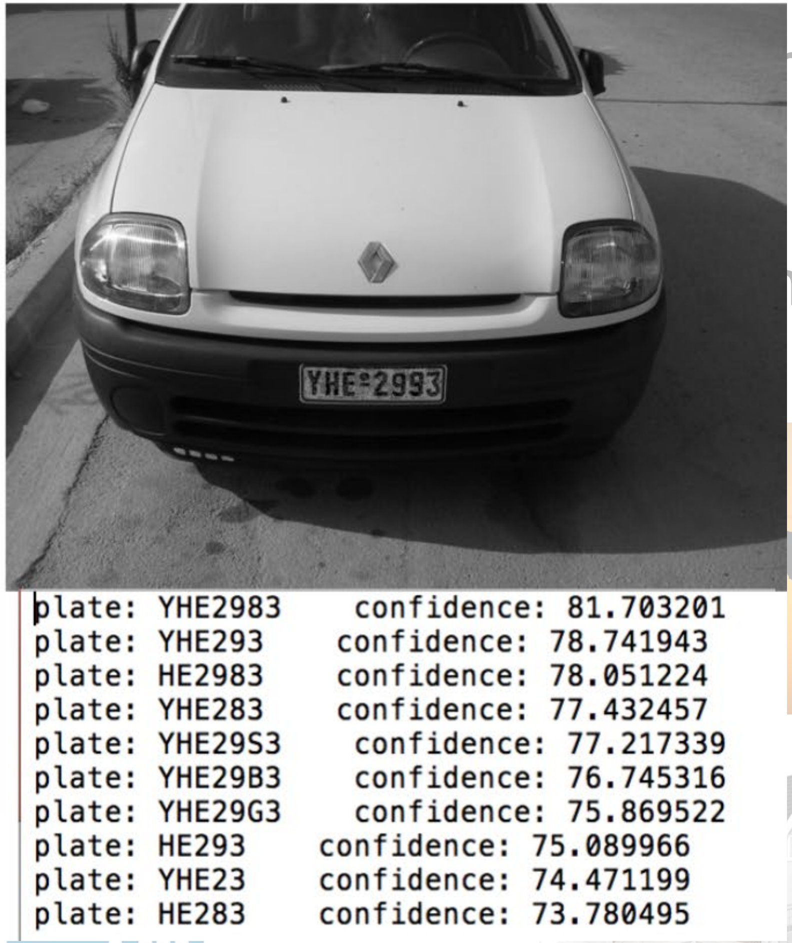
Learning-based testing



Physical testing to smart cities systems



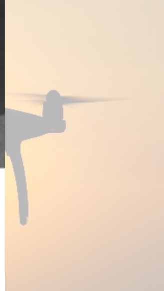
```
plate: YHE2993    confidence: 93.350578
plate: YHE29S3    confidence: 85.806786
plate: YHE29B3    confidence: 85.300774
plate: YHE2S93    confidence: 85.101204
plate: YHEZ993    confidence: 84.646439
plate: YHE293    confidence: 84.447746
plate: YHE2B93    confidence: 83.772606
plate: YME2993    confidence: 83.194237
plate: YHE2SS3    confidence: 77.557419
plate: YHEZ9S3    confidence: 77.102646
```



```
plate: YHE2983    confidence: 81.703201
plate: YHE293    confidence: 78.741943
plate: HE2983    confidence: 78.051224
plate: YHE283    confidence: 77.432457
plate: YHE29S3    confidence: 77.217339
plate: YHE29B3    confidence: 76.745316
plate: YHE29G3    confidence: 75.869522
plate: HE293    confidence: 75.089966
plate: YHE23    confidence: 74.471199
plate: HE283    confidence: 73.780495
```

nart

nt



Reference

- https://securify.chainsecurity.com/
- https://www.probfuzz.com/