PRINCIPLED SAMPLING FOR ANOMALY DETECTION

Brendan Juba, Christopher Musco, Fan Long, Stelios Sidiroglou-Douskos, and Martin Rinard
Anomaly detection trade-off

- Catch **malicious/problematic inputs** before they reach target application.
- Do not filter too many **benign inputs**.
Anomaly detection trade-off

- Catch **malicious/problematic inputs** before they reach target application.
- Do not filter too many **benign inputs**.
Anomaly detection trade-off

- Catch malicious/problematic inputs before they reach target application.
- Do not filter too many benign inputs.
Anomaly detection trade-off

- Catch malicious/problematic inputs before they reach target application.
- Do not filter too many benign inputs.
Detectors need to be tuned!

![Graph showing the relationship between Benign Error Rate and Aggressiveness. The graph shows an increase in Benign Error Rate as Aggressiveness increases.]
Detectors need to be tuned!

![Graph showing the relationship between Benign Error Rate and Aggressiveness.]
Detectors need to be tuned!
Requires accurate error estimation

- Shooting for very low error rates in practice: .01%
- Cost of false positives is high
Estimating error rate

![Diagram showing Anomaly Detector with options for Pass and Reject.]
Estimating error rate

Anomaly Detector

Pass

Reject
Estimating error rate

Anomaly Detector

Pass

Reject
Estimating error rate

Estimated Error Rate:

\[
\frac{\text{(# falsely rejected inputs)}}{\text{(total inputs)}}
\]
What’s needed from a test generator?

Anomaly Detector

Reject
Pass
What’s needed from a test generator?

- Test Case Generator
- Anomaly Detector

Output: Reject, Pass
1) Massive output capability
1) Massive output capability

“With 99% confidence, estimated error rate accurate to within .01%”

Need \( \approx \frac{1}{\varepsilon \log(1/\delta)} \approx 46,000 \) samples
1) Massive output capability
1) Massive output capability

![Diagram showing the relationship between Benign Error Rate and Aggressivenessness. The curve indicates an increase in error rate as aggressiveness increases.]
1) Massive output capability
2) Samples from representative distribution
2) Samples from representative distribution

Typical vs. Testing
2) Samples from representative distribution

With \( \approx \frac{1}{\epsilon \log(1/\delta)} \) samples from distribution \( D \):

“With 99% confidence, estimated error rate accurate to within .01% for inputs drawn from distribution \( D \)”. 
2) Samples from representative distribution

With $\approx \frac{1}{\varepsilon \log(1/\delta)}$ samples from distribution $D$:

“With 99% confidence, estimated error rate accurate to within .01% for inputs drawn from distribution $D$”.

Only meaningful for similar distributions!
"With 99% confidence, our anomaly detector errs on <.01% of benign inputs drawn from distribution D".
Meaningful statistical bounds

“With 99% confidence, our anomaly detector errs on <.01% of benign inputs drawn from distribution D”.

≈ “With 99% confidence, our anomaly detector errs on <.01% of benign inputs seen in practice”.
Easier said than done

Samples need to be:
1. Cheap to generate/collect.
2. Representative of typical input data.

Getting both speed and quality is tough.
Possible for web data

Claim: We can quickly obtain test samples from a distribution representative of typical web inputs.
Possible for web data

Claim: We can quickly obtain test samples from a distribution representative of typical web inputs.

Fortuna: An implemented system to do so.
Random Search

Web Data: Images, JavaScript files, music files, etc.
Not enough coverage
Not enough coverage

Typical vs. Testing
Explicit Distribution

Can obtain a very large (although not quite complete) index of the web from public data sources like Common Crawl

- npr.org
- seahawks.com
- ask.com
- wikipedia.org
- google.com
- patriots.com
- cnn.com
- arxiv.org
- facebook.com
- mit.edu
- dblp.de
Uniform sampling not sufficient
Uniform sampling not sufficient

Typical vs. Testing
Can weight distribution
Can weight distribution
Computationally infeasible

- Need to calculate, store, and share weights (based on traffic statistics, PageRank, etc.) for ~2 billion pages.
- Weights will quickly become outdated.
Web Crawl

Web Data: Images, JavaScript files, music files, etc.
Locally biased

Typical Inputs

Testing Inputs
Locally biased

Typical vs. Testing
Potential Fix?

Combine with uniform distribution to randomly restart the crawl at different pages.
Fortuna based on PageRank
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page 2) Move to random outgoing link 3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page 2) Move to random outgoing link 3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
  1) Start at random page
  2) Move to random outgoing link
  3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page
- 2) Move to random outgoing link
- 3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page 2) Move to random outgoing link 3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page 2) Move to random outgoing link 3) With small probability at each step (15%), jump to new random page
Definition of PageRank

- PageRank is defined by a random surfer process
- 1) Start at random page
   2) Move to random outgoing link
   3) With small probability at each step (15%), jump to new random page
Weight = long run visit probability

- Random surfer more likely to visit pages with more incoming links or links from highly ranked pages.
Weight = long run visit probability

- Random surfer more likely to visit pages with more incoming links or links from highly ranked pages.
The case for PageRank

1. Widely used measure of page importance.
2. Well correlated with page traffic.
3. Stable over time.
The case for PageRank

1. Widely used measure of page importance.
2. Well correlated with page traffic.
3. Stable over time.
PageRank matches typical inputs

Typical Inputs

Testing Inputs
PageRank matches typical inputs
“With 99% confidence, our anomaly detector errs on <.01% of benign inputs drawn from the PageRank distribution”.
Statistically meaningful guarantees

“With 99% confidence, our anomaly detector errs on <.01% of benign inputs drawn from the PageRank distribution”.

≈ “With 99% confidence, our anomaly detector errs on <.01% of benign inputs seen in practice”.
Sample without explicit construction
PageRank Markov Chain

- Surfer process converges to a unique stationary distribution.

- Run for long enough and take the page you land on as a sample. The distribution of this sample will be $\sim$ PageRank.
PageRank Markov Chain

- Surfer process converges to a unique stationary distribution.

- Run for long enough and take the page you land on as a sample. The distribution of this sample will be $\sim$ PageRank.
Sample PageRank by a random walk

Immediately gives a valid sampling procedure:

- Simulate random walk for \( n \) steps. Select the page you land on.

But:

- Need a fairly large number of steps (\( \approx 100 - 200 \)) to get an acceptably accurate sample.
Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = $M$ (probability 85%)
- Jump = $J$ (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:

- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:

- Move = M (probability 85%)
- Jump = J (probability 15%)

JMMMJJMM
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = $M$ (probability 85%)
- Jump = $J$ (probability 15%)

$JMMJMMMM$
Truncating the PageRank walk

Observe Pattern for Movement:

- Move = M (probability 85%)
- Jump = J (probability 15%)

JMMMMMMMM
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:

- Move = $M$ (probability 85%)
- Jump = $J$ (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = M (probability 85%)
- Jump = J (probability 15%)
Truncating the PageRank walk

Observe Pattern for Movement:
- Move = \text{M} (probability 85%)
- Jump = \text{J} (probability 15%)
Fortuna’s final algorithm

1. Flips 85% biased coin n times until a J comes up
2. Choose a random page and take (n-1) walk steps
3. Takes fewer than 7 steps on average!
Fortuna Implementation

- Simple, parallelized Python (700 lines of code)
- Random jumps implemented using a publically available index of Common Crawls URL collection (2.3 billion URLs)

```python
def random_walk(url, walk_length, bias=0.15):
    N = 0
    while True:
        try:
            html_links, soup = get_html_links(url, url, log)
            if (N >= walk_length):
                return get_format_files(soup, url, opts.file_format, log)
            url = random.choice(html_links)
        except Exception as e:
            log.exception("Caught Exception:%s " %type(e))
            url = get_random_url_from_server()
        N += 1
    return []
```
Fortuna Implementation

- Simple, parallelized Python (700 lines of code)
- Random jumps implemented using a publicly available index of Common Crawls URL collection (2.3 billion URLs)

```python
def random_walk(url, walk_length, bias=0.15):
    N = 0
    while True:
        try:
            html_links, soup = get_html_links(url, url, log)
            if (N >= walk_length):
                return get_format_files(soup, url, opts.file_format, log)
            url = random.choice(html_links)
        except Exception as e:
            log.exception("Caught Exception:%s %type(e)")
            url = get_random_url_from_server()
        N += 1
    return []
```

10’s of thousands of samples in just a few hours.
Anomaly Detectors Tested

Sound Input Filter Generation for Integer Overflow Errors:
  SIFT Detector: .011% error

Automatic Input Rectification:
  SOAP Detector: 1.99% error

Detection and Analysis of Drive-by-download Attacks and Malicious JavaScript Code:
  JSAND Detector: .052% error
Anomaly Detectors Tested

Sound Input Filter Generation for Integer Overflow Errors:
  SIFT Detector: .011% error

Automatic Input Rectification:
  SOAP Detector: 1.99% error

Detection and Analysis of Drive-by-download Attacks and Malicious JavaScript Code:
  JSAND Detector: .052% error

Tight bounds with high confidence: can be reproduced over and over from different sample sets.
Additional benefits of Fortuna
Additional benefits of Fortuna

- Adaptable to local networks
Additional benefits of Fortuna

- Adaptable to local networks
- Does not require any data besides a web index
Additional benefits of Fortuna

- Adaptable to local networks
- Does not require any data besides a web index
- PageRank naturally incorporates changes over time
For web data we obtain:

Samples need to be:
1. Cheap to generate/collect.
2. Representative of typical input data.

Getting both speed and quality is very possible.
Step towards rigorous testing

Typical Inputs

Testing Inputs
Step towards rigorous testing

Typical Inputs

Testing Inputs

Thanks!