PerfFuzz: Automatically Generating Pathological Inputs

Caroline Lemieux, Rohan Padhye, Koushik Sen, Dawn Song
University of California, Berkeley

source: https://github.com/carolemieux/perffuzz
Nobody Expects Performance Problems
Performance Problems Have Consequences
Performance Problems Have Consequences

CWE-407: Algorithmic Complexity

An algorithm in a product has an inefficient worst-case computational complexity that may be detrimental to system performance and can be triggered by an attacker, typically using crafted manipulations that ensure that the worst case is being reached.

poor user experience
Performance Problems Have Consequences

- Poor user experience
- Excessive resource consumption
Performance Problems Have Consequences

- poor user experience
- excessive resource consumption
- security vulnerabilities (DoS)
Alleviating Performance Problems

Pathological Input → Profiling Tool → Evaluation

PerfFuzz, presented by Caroline Lemieux
Alleviating Performance Problems

Pathological Input → Profiling Tool → Evaluation

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Goal

Automatically generate pathological inputs.
PerfFuzz

- A feedback-directed mutational fuzzing tool
- Uses performance feedback to produce pathological inputs
PerfFuzz

• A feedback-directed mutational fuzzing tool
  • **Fuzzing**: sends inputs to program
  • **Mutational**: creates new inputs by mutating saved inputs
  • **Feedback-directed**: saves inputs if program gives *interesting* feedback
• A feedback-directed mutational fuzzing tool
• Uses performance feedback to produce pathological inputs
  • First idea: interesting if longer execution time, path length [1]
  • PerfFuzz: interesting if higher execution count of any given CFG edge

Example Program: Word Frequency (wf)

• Count # occurrences of words in a string
  input:
  
  the quick brown the dog
  
  output:
  
  brown: 1
dog: 1
quick: 1
the: 2

• wf shipped with Fedora Linux had real performance bugs
Example Program: Word Frequency (wf)

• Count # occurrences of words in a string
  
  input:  
  
  the quick brown the dog

  output:  
  
  brown: 1  
  dog: 1  
  quick: 1  
  the: 2

• wf shipped with Fedora Linux had real performance bugs

```
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        entry = entry.next
        break
    table[id] = new entry(word=word, count=1, next=table[id])
```

PerfFuzz, presented by Caroline Lemieux
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry = entry.next
            entry.count += 1
            break
    if entry
        entry = new entry(word=word, count=1, next=table[id])
        table[id] = entry
    else
        entry = table[id].new(entry=word, count=1, next=table[id])
        table[id] = entry
We look at a subset of CFG edges for illustration purposes.
wf Performance Response

- Usual case:

  the quick brown the dog

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

```
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```
**wf Performance Response**

- Usual case:

```
the quick brown the dog
```

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

```
for word in words
    id = hash(word)
    entry = table[id]
while entry != None
    if entry.word == word
        entry = entry.next
        entry.count += 1
        break
```
wf Performance Response

• Usual case:

the quick brown the dog

---

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

- for word in words
- id = hash(word)
- entry = table[id]
- while entry != None
- if entry.word == word
- entry = entry.next
- entry.count += 1
- break
- table[id] = new entry(word=word, count=1, next=table[id])

---

T        F
T        F
A        B
C        D
4        1

Intro
Method
Evaluation
End
**Usual case:**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    else:
        entry = new entry(word=word, count=1, next=entry)
        table[id] = entry
```

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usual case:**

the quick brown the dog
Performance Response

- Usual case:
  the quick brown the dog

- Hash collisions:
  t ?t xt at$ #a ))t Qwaa

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

```
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry = entry.next
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

T F T F

---

Usual case: the quick brown the dog

Hash collisions: t ?t xt at$ #a ))t Qwaa
**wf Performance Response**

- **Usual case:**
  ```
  the quick brown the dog
  ```

- **Hash collisions:**
  ```
  t ?t xt at$ #a ))t Qwaa
  ```

- **Small words:**
  ```
  t h e q u i c k b r o w
  ```

---

**Evaluation**

```
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    entry = table[id] = new entry(word=word, count=1, next=table[id])
```
**Performance Response**

- **Usual case:**
  
  the quick brown the dog

- **Hash collisions:**
  
  t ?t xt at$ #a ))t Qwaa

- **Small words:**
  
  the quick brown

---

**Code Snippet:**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

---

**Table:**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Another Table:**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>

**Yet Another Table:**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>
**Performance Response**

- **Usual case:**
  - the quick brown the dog

- **Hash collisions:**
  - t ?t xt at$ #a ))t Qwaa

- **Small words:**
  - the quick brown

### Table:

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

### Code:
```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        break
```

→ Pathological behavior characterized by a few CFG edges.

- **Pathological behavior:**
  - Characterized by a few CFG edges.

---

**Intro**

- Method
- Evaluation
- End
**Performance Response**

- **Usual case:**

  the quick brown the dog

- **Hash collisions:**

  t ?t xt at$ #a ))t Qwaa

- **Small words:**

  the quick brown

---

**Table: # Hits**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table: # Hits**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table: # Hits**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>

**Diagram:**

- `id = hash(word)`
- `entry = table[id]`
- `for word in words`

**Pathological behavior characterized by a few CFG edges.**

**Increasing execution counts of edges:**
- less noisy than path length.
- Greedy approach won’t get stuck.

**Path**

- `table[id] = new entry(word=word, count=1, next=table[id])`
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mutation Engine**

- Parent to Mutate

**Input Running, Feedback Analysis**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Program Under Test**

```
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry.count += 1
            entry = entry.next
        break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

**Table**

```
Edge  Max # Hits  Maximizing Input
      |            |                  |
      |            |                  |
      |            |                  |
```

**Evaluation**

PerfFuzz, presented by Caroline Lemieux

University of California, Berkeley
PerfFuzz Algorithm

**Program Under Test**

- **for** `word in` `words`

- **id = hash(word)**

- **entry = table[id]**

- **while** `entry != None`

- **if** `entry.word == word`

- **entry = entry.next**

- **entry.count += 1**

- **break**

- **table[id] = new entry(word=word, count=1, next=table[id])**

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

**Input Running, Feedback Analysis**

**Seed Input**

- `the quick brown the dog`

**Current Input**

- `the quick brown the dog`
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Input Corpus</th>
<th>Program Under Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge</strong></td>
<td><strong>Max # Hits</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

**Input Running, Feedback Analysis**

**Current Input**

- the quick brown the dog

```
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    entry.count += 1
    break
```

```
T

F

T

B

A

F

C

D

T

entry = entry.next
entry.count += 1
break
```

```
table[id] = new entry(word=word, count=1, next=table[id])
```
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mutation Engine**

Parent to Mutate

**Input Running, Feedback Analysis**

Current Input

the quick brown the dog

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next

    table[id] = new entry(word=word, count=1, next=table[id])
```

**Intro**

Method

Evaluation

End
PerfFuzz Algorithm

input corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

mutation engine

Parent to Mutate

input running, feedback analysis

Current Input

the quick brown the dog

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

program under test

```
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

T

F

F

PerfFuzz, presented by Caroline Lemieux

University of California, Berkeley
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

**Input Running, Feedback Analysis**

**Current Input**

the quick brown the dog

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
        else:
            entry.count += 1
            break
    entry = entry.next
    entry.count += 1
    break
```

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Program under test**

- `for word in words`
- `id = hash(word)`
- `entry = table[id]`
- `while entry != None`
- `if entry.word == word`
- `entry = entry.next` (for A, B, D)
- `entry.count += 1` (for A, B, D)
- `break`

**Table**

<table>
<thead>
<tr>
<th>Table Entry</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Performance Fuzz (PerfFuzz)**, presented by Caroline Lemieux
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

program under test

```
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry.count += 1
            break
        entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

PerfFuzz, presented by Caroline Lemieux
### PerfFuzz Algorithm

#### Input Corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

#### Mutation Engine

**Parent to Mutate**

the quick brown the dog

#### Input Running, Feedback Analysis

<table>
<thead>
<tr>
<th>Current Input</th>
</tr>
</thead>
</table>

#### Program Under Test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            entry = entry.next
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

---

### PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

- the quick brown the dog

**Current Input**

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            entry = entry.next
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

**Method**

**Evaluation**

- **Maximizing Input**
  - A: 4
  - B: 1
  - D: 1

**End**
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Input Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge</strong></td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mutation Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent to Mutate</strong></td>
</tr>
<tr>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Running, Feedback Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Current Input</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program Under Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for</strong> word in words</td>
</tr>
<tr>
<td>id = hash(word)</td>
</tr>
<tr>
<td>entry = table[id]</td>
</tr>
<tr>
<td>while entry != None</td>
</tr>
<tr>
<td>if entry.word == word</td>
</tr>
<tr>
<td>entry = entry.next</td>
</tr>
<tr>
<td>entry.count += 1</td>
</tr>
<tr>
<td>break</td>
</tr>
<tr>
<td>table[id] = new entry(word=word, count=1, next=table[id])</td>
</tr>
</tbody>
</table>
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

input corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

program under test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    if entry.count > 0:
        entry = entry.next
    else:
        entry = entry.next
        table[id] = new entry(word=word, count=1, next=table[id])
```

mutation engine

Parent to Mutate

the quick brown the dog

input running, feedback analysis

new mutant

Current Input

the quack brown the dog

University of California, Berkeley

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

Program under test:

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

Input corpus:

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

Mutation engine:

Parent to Mutate:
the quick brown the dog

Current Input:
the quack brown the dog
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

Parent to Mutate

the quick brown the dog

**Current Input**

the quack brown the dog

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        entry = entry.next
    break

T
F

edge = 4
while entry != None:
    if entry.word == word:
        entry.count += 1
        entry = entry.next
        break
```

**Evaluation**

Performance evaluation of PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

- **Parent to Mutate**: the quick brown the dog

**Input Running, Feedback Analysis**

- **Current Input**: the quack brown the dog

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            entry = entry.next
        break
    else:
        table[id] = new entry(word=word, count=1, next=table[id])
```

---

PerfFuzz, presented by Caroline Lemieux

University of California, Berkeley
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

the quick brown the dog

**Input Running, Feedback Analysis**

**Current Input**

the quack brown the dog

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    table[id] = new_entry(word=word, count=1, next=table[id])
```

**Maximizing Input**

A

**# Hits**

Maximizing Input

the quick brown the dog

**Current Input**

the quack brown the dog

**Method**

- PerfFuzz, presented by Caroline Lemieux
- University of California, Berkeley
PerfFuzz Algorithm

Program under test:

```python
for word in words:
    id = hash(word)
    entry = table[id]

while entry != None:
    if entry.word == word:
        entry = entry.next
        entry.count += 1
    break
```

Mutation engine:

Parent to Mutate:
the quick brown the dog

Input running, feedback analysis:

Current Input:
the quack brown the dog

Input corpus:

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

- **Parent to Mutate**: the quick brown the dog
- **Current Input**: the quick brown the dog

**Program Under Test**

```
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry = entry.next
        entry.count += 1
        table[id] = new entry(word=word, count=1, next=table[id])
    else:
        # handle other cases
```

**Table**

```
<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

Program under test:

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

Input corpus:

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input running, feedback analysis:

- Parent to Mutate: the quick brown the dog
- Current Input: the quick brown the dog

Method:

- PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

program under test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        entry = entry.next
    break
```

Input corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
</tbody>
</table>

Mutation engine

Parent to Mutate

the quick brown the dog

Input running, feedback analysis

Current Input

the quick brown the dog

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>input corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge</strong></td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>program under test</th>
</tr>
</thead>
<tbody>
<tr>
<td>for word in words</td>
</tr>
<tr>
<td>id = hash(word)</td>
</tr>
<tr>
<td>entry = table[id]</td>
</tr>
<tr>
<td>while entry != None</td>
</tr>
<tr>
<td>if entry.word == word</td>
</tr>
<tr>
<td>entry = entry.next</td>
</tr>
<tr>
<td>entry.count += 1</td>
</tr>
<tr>
<td>break</td>
</tr>
<tr>
<td>table[id] = new entry(word=word, count=1, next=table[id])</td>
</tr>
</tbody>
</table>

PerfFuzz, presented by Caroline Lemieux

University of California, Berkeley
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Parent to Mutate**
the quick brown the dog

**Current Input**
the quick brown the dog

**Input running, feedback analysis**

**Program under test**

```
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

**Input corpus**

**Mutation engine**

**Method**

**Evaluation**

**End**
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

**Parent to Mutate**

the quick brown the dog

**Input Running, Feedback Analysis**

**Current Input**

the quick brown the dog

**Program Under Test**

```python
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry = entry.next
            entry.count += 1
            break
    entry = new entry(word=word, count=1, next=table[id])
    table[id] = entry
```

**Intro**

**Method**

**Evaluation**

**End**

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

*Parent to Mutate*
the quick brown the dog

**Input Running, Feedback Analysis**

*Current Input*
the quick brown the dog

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
        break
    table[id] = new entry(word=word, count=1, next=table[id])
```
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

Program under test:

```
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry = entry.next
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

Input corpus:

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PerfFuzz Algorithm

input corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

mutation engine

Parent to Mutate

the quick brown the dog

input running, feedback analysis

Current Input

the quick brown the dog

program under test

for word in words:

    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry = entry.next
        entry.count += 1
        break

    entry = entry.next

    if entry != None:
        table[id] = new entry(word=word, count=1, next=table[id])

    for word in words:
        id = hash(word)
        entry = table[id]
PerfFuzz Algorithm

**Input Corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

**Mutation Engine**

*Parent to Mutate*

the quick brown the dog

**Input Running, Feedback Analysis**

*Current Input*

the quick brown the dog

**Program Under Test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
            break
        else:
            table[id] = new entry(word=word, count=1, next=table[id])
```

---

**PerfFuzz, presented by Caroline Lemieux**

University of California, Berkeley
PerfFuzz Algorithm

**Program under test**

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
```

**Input corpus**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td><code>the quick brown the dog</code></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td><code>the quick_brown the dog</code></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td><code>the quick_brown the dog</code></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td><code>the quick brown the dog</code></td>
</tr>
</tbody>
</table>

**Mutation engine**

**Parent to Mutate**

- `the quick brown the dog`

**Input running, feedback analysis**

**Current Input**

- `the quick_brown the dog`

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>
PerfFuzz Algorithm

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown t_e^e^e dog</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>the quick_brown the dog</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>the quick_brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

---

**Program Under Test**

```python
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry.count += 1
            break
    entry = entry.next
    table[id] = new entry(word=word, count=1, next=table[id])
```

---

**Mutation Engine**

**Parent to Mutate**

the quick brown the dog

---

**Input Running, Feedback Analysis**

Current Input

---

**Intro**

Method

Evaluation

End
PerfFuzz Algorithm

### Input Corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown t<strong>e</strong> dog</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>the quick_brown the dog</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>the quick_brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

### Mutation Engine

**Parent to Mutate**

the quick brown t**e** dog

### Program Under Test

```
for word in words
    id = hash(word)
    entry = table[id]
    while entry != None
        if entry.word == word
            entry = entry.next
            entry.count += 1
            break
    else:
        entry = entry.next
        entry.count += 1
        table[id] = new entry(word=word, count=1, next=table[id])
```

### Current Input

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
</table>

### Method

**For word in words**

**While entry != None**

**If entry.word == word**

**Entry = entry.next**

**Entry.count += 1**

**Break**

**Table[id] = new entry(word=word, count=1, next=table[id])**
PerfFuzz Algorithm

### Input Corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown <strong>t</strong>e dog</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>the quick_ brown the dog</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>the quick_ brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

### Mutation Engine

**Parent to Mutate**

the quick brown **t**e dog

Let's mutate this many times

### Program Under Test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        entry = entry.next
    break
```

Current Input

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm

### Input Corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>the quick brown the dog</td>
</tr>
</tbody>
</table>

### Program Under Test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    if entry.word == word:
        entry.count += 1
        break
```

### Mutation Engine

**Parent to Mutate:**
the quick brown t

**Current Input:**

- A: 6 hits, the quick brown the dog
- B: 2 hits, the quick brown the dog
- C: 1 hit, the quick brown the dog
- D: 1 hit, the quick brown the dog

Repeat until timeout.
PerfFuzz Algorithm: Results

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>t t t t t t t t t t t t</td>
</tr>
</tbody>
</table>

program under test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

mutation engine

input running, feedback analysis

Current Input

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm: Results

Input Corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>t t t t t t t t t t t</td>
</tr>
</tbody>
</table>

Mutation Engine

Current Input

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
</table>

Program Under Test

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
            break
    table[id] = new entry(word=word, count=1, next=table[id])
```

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm: Results

input corpus

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>t?x?at$ #a )t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>t?x?at$ #a )t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>tttttttttttttt</td>
</tr>
</tbody>
</table>

Program

```python
for word in words:
    id = hash(word)
    entry = table[id]
    while entry != None:
        if entry.word == word:
            entry.count += 1
            break
    entry = entry.next
    entry.count += 1
    table[id] = new entry(word=word, count=1, next=table[id])
```

mutation engine

input running, feedback analysis

Current Input

Table

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PerfFuzz Algorithm: Results

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>??xt at$ #a )t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>??xt at$ #a )t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>t t t t t t t t t t t</td>
</tr>
</tbody>
</table>

many unique words

many hash collisions in same bucket

PerfFuzz, presented by Caroline Lemieux
PerfFuzz Algorithm: Results

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>t ?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>ttttttttttttt</td>
</tr>
</tbody>
</table>

- many unique words
- many hash collisions in same bucket
- many occurrences of same word

Program:

```
for word in words:
    while entry != None:
        if entry.word == word:
            entry = entry.next
            entry.count += 1
            break

table[id] = new entry(word=word, count=1, next=table[id])
```
PerfFuzz Algorithm: Results

<table>
<thead>
<tr>
<th>Edge</th>
<th>Max # Hits</th>
<th>Maximizing Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td>the quick brown</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>t?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>C</td>
<td>21</td>
<td>t?t xt at$ #a ))t Qwaa</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
<td>tttttttttttt</td>
</tr>
</tbody>
</table>

- many unique words
- many hash collisions in same bucket
- many occurrences of same word
- → variety of performance behaviors

program

```
for word in words
    entry = table[hash(word)]
    if entry.word == word
        entry.count += 1
        entry = entry.next
        break
    entry = table[hash(word)]
    entry = new entry(word=word, count=1, next=table[id])
```

Intro

<table>
<thead>
<tr>
<th>Method</th>
<th>Evaluation</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

University of California, Berkeley

PerfFuzz, presented by Caroline Lemieux
Evaluation Outline

• Compare to SlowFuzz
  • Macro-benchmarks
  • Micro-benchmarks

• Compare to AFL

• Case Studies
Evaluation Outline

• Compare to SlowFuzz
  • Macro-benchmarks
  • Micro-benchmarks

• Compare to AFL

• Case Studies  (more in paper)
Prior Work

- **SlowFuzz**
  → Fuzzing to find algorithmic complexity vulnerabilities

- Saves inputs that increase *total* path length
- Randomly chooses parent
- Prioritizes mutations that increase path length
- Faster than PerfFuzz (based on LibFuzzer)

Prior Work

SlowFuzz

Fuzzing to find algorithmic complexity vulnerabilities

- Saves inputs that increase total path length
- Randomly chooses parent
- Prioritizes mutations that increase path length
- Faster than PerfFuzz (based on LibFuzzer)

Experimental Setup: Macro-Benchmarks

• Max input size: 500 bytes
• Seeds: AFL default seed for each format
• Run each tool for 6 hours
• Repeat 6-hour runs 20 times

<table>
<thead>
<tr>
<th>Library</th>
<th>LoC</th>
<th>Function Exercised</th>
</tr>
</thead>
<tbody>
<tr>
<td>libpng</td>
<td>30k</td>
<td>PNG read</td>
</tr>
<tr>
<td>libxml2</td>
<td>70k</td>
<td>XML read</td>
</tr>
<tr>
<td>libjpeg-turbo</td>
<td>30k</td>
<td>JPEG decompress</td>
</tr>
<tr>
<td>zlib</td>
<td>9k</td>
<td>GZIP decompress</td>
</tr>
</tbody>
</table>
Macro-Benchmarks: Maximum Path Length

• Path length: total number of hits of CFG edges by an input

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
</tr>
</tbody>
</table>

path len: 23
Macro-Benchmarks: Maximum Path Length

- Path length: total number of hits of CFG edges by an input

libpng

libxml2

libjpeg-turbo

zlib
Macro-Benchmarks: Maximum Path Length

- Path length: total number of hits of CFG edges by an input

libpng

libjpeg-turbo

libxml2

zlib
Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

<table>
<thead>
<tr>
<th>Edge</th>
<th># Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
</tr>
</tbody>
</table>

hot spot: 11
Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

- **libpng**

- **libjpeg-turbo**

- **libxml2**

- **zlib**
Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

libpng

libjpeg-turbo

libxml2

zlib
What Does It Mean?

libxml2 case study:
libxml2 case study:
libxml2 case study:

Parse error: Double hyphen within comment: <!--3
<a>&gt;&gt;&gt;0&gt;&gt;&gt;#&gt;&lt;--3--6------------------------4---------------------^

Parse error: Double hyphen within comment: <!--3--6
<a>&gt;&gt;&gt;0&gt;&gt;&gt;#&gt;&lt;--3--6------------------------4---------------------^

Parse error: Double hyphen within comment: <!--3--6
<a>&gt;&gt;&gt;0&gt;&gt;&gt;#&gt;&lt;--3--6------------------------4---------------------^
What Does It Mean?

libxml2 case study:

- a loop in xml strncpy

Output of read XML on that input:

parser error: Double hyphen within comment: "<!--3
<aaaaaaa>"^G<!--3--6--4--

PerfFuzz, presented by Caroline Lemieux
Experimental Setup: Micro-Benchmarks

• Choose benchmarks with known worst-case complexity:

<table>
<thead>
<tr>
<th>Micro-benchmark</th>
<th>Complexity</th>
<th>Seed</th>
<th>Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion sort</td>
<td>(n^2)</td>
<td>List of 0’s</td>
<td>10 min</td>
</tr>
<tr>
<td>(SlowFuzz example)</td>
<td>(n = \text{input len})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCRE regex match</td>
<td>(n^2)</td>
<td>Null string</td>
<td>60 min</td>
</tr>
<tr>
<td>(URL regex)</td>
<td>(n = \text{input len})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wf-0.41</td>
<td>(m^2)</td>
<td>“the quick brown fox jumps over the lazy dog”</td>
<td>60 min</td>
</tr>
<tr>
<td>(Fedora Linux)</td>
<td>(m = \text{num words})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Repeat 20 runs for each input length: 10, 20, …, 60 bytes.
Micro-Benchmarks: Algorithmic Complexity

• Maximum path length for varying input sizes
Micro-Benchmarks: Algorithmic Complexity

- Maximum path length for varying input sizes

**Insertion Sort**
- PerfFuzz
- SlowFuzz

**PCRE URL regex**
- PerfFuzz
- SlowFuzz

**Word Frequency**
- PerfFuzz
- SlowFuzz

PerfFuzz, presented by Caroline Lemieux
Back to our Motivating Example

• SlowFuzz worst case:
  t r t t s f o Öe r t s f o r t x x t s f o r t x x

• PerfFuzz worst case:
  t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t t
Back to our Motivating Example

- SlowFuzz worst case:
  
t t t s f o Ö e r t s f o r t x x t s f o r t x x

- PerfFuzz worst case:
  
t <81>v ^?@t <80>! ^?@t <80>!t t^Rn t t t t t t t t
Back to our Motivating Example

• SlowFuzz worst case:
  \[ t r t t s f o \ddot{e} r t s f o r t x x t s f o r t x x \]

• PerfFuzz worst case:
  \[ t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t t \]
Back to our Motivating Example

- **SlowFuzz worst case:**
  \[ t \leq 81 \]

- **PerfFuzz worst case:**
  \[ t < 81 \\& \& \neg @t < 80 \]
Back to our Motivating Example

- SlowFuzz worst case:
  
  $t \texttt{tsf fo } \texttt{ttsfortxx}$

- PerfFuzz worst case:
  
  $t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t$
Back to our Motivating Example

- SlowFuzz worst case:

  \[ t t t s f o Ö e r t s f o r t x x t s f o r t x x \]

- PerfFuzz worst case:

  \[ t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t \]

\[ \]

- PerfFuzz, presented by Caroline Lemieux
Back to our Motivating Example

- SlowFuzz worst case:
  
  \[
  t \text{ } t \text{ } t \text{ } s \text{ } f \text{ } o \text{ } Ö \text{ } e \text{ } r \text{ } t \text{ } s \text{ } f \text{ } o \text{ } r \text{ } t \text{ } x \text{ } x \text{ } t \text{ } s \text{ } f \text{ } o \text{ } r \text{ } t \text{ } x \text{ } x
  \]

- PerfFuzz worst case:
  
  \[
  t \text{ } <81>v \text{ } ^\text{?}@t \text{ } <80>!\text{ } ^\text{?}@t \text{ } <80>!t \text{ } t \text{ } ^\text{Rn} \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t \text{ } t
  \]

- Diagram:
  
  ![Diagram](attachment://diagram.png)
Back to our Motivating Example

- SlowFuzz worst case:
  \[ t \text{ RR } t s f o \text{ Öer ts for } t x x t s f o r t x x \]

- PerfFuzz worst case:
  \[ t \text{ <81>v } ^?@t \text{ <80>!^?@t <80>!t t^Rn t t t t t t t t t } \]
Back to our Motivating Example

- SlowFuzz worst case:
  
  ```
  t r t s f o \ddot{e} r t s f o r t x x t s f o r t x x
  ```

- PerfFuzz worst case:
  
  ```
  t <81> v ^?@t <80> !^?@t <80> !t t ^Rn t t t t t t t t t
  ```
Back to our Motivating Example

• SlowFuzz worst case:

  t r t t s f o Ö e r t s f o r t x x t s f o r t x x

• PerfFuzz worst case:

  t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t t t

```plaintext
<80>^Rn : 1 <80>t : 1 <80>! ^@t : 1 ^?@t : 1 <81>v : 1 t : 10
```
Conclusion

How to find **pathological inputs**?

Use **feedback-directed mutational fuzzing**!

Multi-dimensional feedback more effective.

Where's the code?

[https://github.com/carolelemieux/perffuzz](https://github.com/carolelemieux/perffuzz)