NEUZZ: Efficient Fuzzing with Neural Program Smoothing

Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, and Suman Jana
Columbia University
Fuzzing: a popular way to uncover bugs

Number of fuzzing papers in top CS conferences


[Liang et al. 2019]
Evolutionary Fuzzing

Advantage: easy to implement
Disadvantage: inefficient
  • Random mutations are not effective
  • Often get stuck in long sequence of wasteful mutations

Hard to find scalable and adaptive heuristics for guided mutation
A new approach to fuzzing
Fuzzing: An Optimization Problem

- $x$: a program input $x \in X$
- $F(x)$: # of bugs found by input $x$
- $C(X)$: generate $K$ inputs from input space $X$

Maximize $\sum_{x \in C(X)} F(x)$

Find $C(X)$ that can maximize total no. of bugs

$F(x)$ is discrete and hard to optimize
Fuzzing: An Optimization Problem

$F(x)$ : # of bugs

Input $x$

$F(x)$ is hard to find inputs like $x_1$ and $x_2$ among flat plateaus.
Fuzzing: An Optimization Problem

\[ \sum_{x \in C(X)} G(x) \]

Maximize \( \sum_{x \in C(X)} G(x) \)

Find \( C(X) \) that can maximize total number of edges

\( x \rightarrow \) a program input \( x \in X \)

\( G(x) \rightarrow \) edge coverage of input \( x \)

\( C(X) \rightarrow \) generate \( K \) inputs from input space \( X \)
Fuzzing: An Optimization Problem

\[ G(x) : \text{# of edges} \]

Input \( x \)
Evolutionary optimization

$G(x):$ # of edges

Random mutation is not efficient
Gradient-guided Optimization

Smooth Approximation + Gradient-guided Mutation

\[ E(x) : \text{smooth approximation of } G(x) \]
Gradient-guided Optimization

Smooth Approximation + Gradient-guided Mutation

\[ H(x) : \text{smooth approximation of } G(x) \]
Smooth Approximation

**Problem:**
How to smoothly approximate $G(x)$?

**Universal Approximation Theorem:**
A NN can approximate any continuous function

**Neuzz Solution:**
Use a NN to learn a smooth $H(x)$
Gradient-guided Mutation

**Why gradient guidance?**
Gradient indicates critical parts of input

**What are critical parts of the input?**
Critical parts of input affect program branches

**How gradient-guided mutation works?**
Focus mutations on the critical parts of the input
Main Idea behind Neuzz

Gradient-guided mutation
\[ \frac{\partial \text{branch}}{\partial \text{input}} \]

Input

Program

branching Behaviors

Smooth Surrogate

Input

NN

Branching Behaviors
A Peek Into NN Model

Control flow graph of program

Input bytes in hex

01
0A
05
0E
02
0B

Edge bitmap

0
1
0
0

Learn

Training Data X

Neural Network Y=f(X)

Training Data Y
Generalization to Unseen branches

Observations:
- Real world program inputs have critical parts
- Most of branches are affected by the critical parts

Neuzz Solution:
- Identify critical parts based on observed branches
- Perform more mutations on the critical part of inputs to explore unseen branches
Design of NEUZZ

Initial seeds → Neural smoothing → Smooth NN model → Gradient-guided optimization

Refine with incremental learning

Bugs/vulnerabilities → Target program

Test inputs
Evaluation

- 10 real world programs
- Lava-M and DARPA CGC datasets
- Comparison with RNN-based fuzzers
- Performance of different model choices
Evaluations: Edge Coverage
NEUZZ vs. state-of-the-art fuzzers

10 real world applications for 24 hours

NEUZZ achieves on average 3x more edge coverage than other fuzzers
Evaluations: Bug Finding
NEUZZ vs. state-of-the-art fuzzers

<table>
<thead>
<tr>
<th>Programs</th>
<th>AFL</th>
<th>AFLFast</th>
<th>VUzzer</th>
<th>KleeFL</th>
<th>AFL-laf-intel</th>
<th>NEUZZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>readelf</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>nm</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>objdump</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>size</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>strip</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>librjpeg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Detected Bugs per Project

<table>
<thead>
<tr>
<th>Detected Bugs per Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-of-memory</td>
</tr>
<tr>
<td>memory leak</td>
</tr>
<tr>
<td>assertion crash</td>
</tr>
<tr>
<td>integer overflow</td>
</tr>
<tr>
<td>heap overflow</td>
</tr>
</tbody>
</table>

Total: 29 27 7 14 26 60

NEUZZ finds the most number of bugs and all 5 bug types including two new CVEs.
Evaluations: Lava-M and CGC

### Lava-M dataset

<table>
<thead>
<tr>
<th></th>
<th>base64</th>
<th>md5sum</th>
<th>uniq</th>
<th>who</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Bugs</td>
<td>44</td>
<td>57</td>
<td>28</td>
<td>2,136</td>
</tr>
<tr>
<td>FUZZER</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>SES</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>VUzzer</td>
<td>17</td>
<td>1</td>
<td>27</td>
<td>50</td>
</tr>
<tr>
<td>Steelix</td>
<td>43</td>
<td>28</td>
<td>24</td>
<td>194</td>
</tr>
<tr>
<td>Angora</td>
<td>48</td>
<td>57</td>
<td>29</td>
<td>1,541</td>
</tr>
<tr>
<td>AFL-laf-intel</td>
<td>42</td>
<td>49</td>
<td>24</td>
<td>17</td>
</tr>
<tr>
<td>T-fuzz</td>
<td>43</td>
<td>49</td>
<td>26</td>
<td>63</td>
</tr>
<tr>
<td><strong>NEUZZ</strong></td>
<td>48</td>
<td>60</td>
<td>29</td>
<td>1,582</td>
</tr>
</tbody>
</table>

### DARPA CGC dataset

<table>
<thead>
<tr>
<th></th>
<th>AFL</th>
<th>Driller</th>
<th>NEUZZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs</td>
<td>21</td>
<td>25</td>
<td>31</td>
</tr>
</tbody>
</table>

NEUZZ outperforms state-of-the-art fuzzers on LAVA-M and CGC
Evaluations: NEUZZ vs. RNN-based Fuzzer

<table>
<thead>
<tr>
<th>Programs</th>
<th>Edge Coverage</th>
<th>Training Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NEUZZ</td>
<td>RNN</td>
</tr>
<tr>
<td>readelf -a</td>
<td>1,800</td>
<td>215</td>
</tr>
<tr>
<td>libjpeg</td>
<td>89</td>
<td>21</td>
</tr>
<tr>
<td>libxml</td>
<td>256</td>
<td>38</td>
</tr>
<tr>
<td>mupdf</td>
<td>260</td>
<td>70</td>
</tr>
</tbody>
</table>

NEUZZ achieves 6x more edge coverage and 20x less training time
Evaluations: Effect of Different NNs

Edge coverage for 1M mutations

<table>
<thead>
<tr>
<th>Programs</th>
<th>Linear Model</th>
<th>NN Model</th>
<th>NN + Incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td>readelf -a</td>
<td>1,723</td>
<td>1,800</td>
<td>2,020</td>
</tr>
<tr>
<td>libjpeg</td>
<td>63</td>
<td>89</td>
<td>159</td>
</tr>
<tr>
<td>libxml</td>
<td>117</td>
<td>256</td>
<td>297</td>
</tr>
<tr>
<td>mupdf</td>
<td>93</td>
<td>260</td>
<td>329</td>
</tr>
</tbody>
</table>

NEUZZZ achieves best performance with NN+Incremental learning
Key Takeaways of NEUZZ

● Use NN gradients to identify the critical locations of program inputs
● Focus mutations on the critical locations
● Minimize runtime overhead by using simple feed-forward neural networks
● Retrain the network incrementally to find new critical locations
NEUZZ is available at
https://github.com/Dongdongshe/neuzz
NEUZZ: Efficient Fuzzing with Neural Program Smoothing

Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, and Suman Jana
Columbia University