QFuzz: Quantitative Fuzzing for Side Channels

Yannic Noller

Saeid Tizpaz-Niari

yannic.noller@acm.org
saeid@utep.edu
Detection / Quantification of Side-Channel Vulnerabilities

stringEquals (Original Jetty, v1)

```java
boolean stringEquals(String s1, String s2) {
    if (s1 == s2)
        return true;
    if (s1 == null || s2 == null ||
        s1.length() != s2.length())
        return false;
    for (int i = 0; i < s1.length(); ++i)
        if (s1.charAt(i) != s2.charAt(i))
            return false;
    return true;
}
```

conditional early return causes leakage

Side Channel Vulnerability

- leakage of secret data
- software side-channels
- observables (e.g., execution time)

Detection vs Quantification

Is there a vulnerability?
⇔
How much information can be leaked?
State of the Art

Challenges

How to go **beyond** non-interference?

How to avoid **expensive** symbolic execution?

How to **scale** to larger programs?

How to provide **guarantees** for vulnerability?

---

**Blazer**
Timos Antoniou, Paul Gazzillo, Michael Hicks¹, Eric Koskinen, Tachio Terauchi¹, and Shiyi Wei¹
Yale University ¹ University of Maryland ¹ JAIST

**Themis**
Jia Chen
University of Texas at Austin
jchen@cs.utexas.edu

Yu Feng
University of Texas at Austin
yufeng@cs.utexas.edu

Isil Dillig
University of Texas at Austin
islil@cs.utexas.edu

**DiffFuzz**
Shirin Nilizadeh*
University of Texas at Arlington
Arlington, TX, USA
shirin.nilizadeh@uta.edu

Yannic Noller*
Humboldt-Universität zu Berlin
Berlin, Germany
yannic.noller@hu-berlin.de

Corina S. Păsăreanu
Carnegie Mellon University Silicon Valley,
NASA Ames Research Center
Moffett Field, CA, USA
corina.s.pasaareanu@nasa.gov

**MaxLeak**
Corina S. Păsăreanu
Carnegie Mellon University, NASA Ames
Moffett Field, CA, USA
corina.s.pasaareanu@nasa.gov

Quoc-Sang Phan
Carnegie Mellon University
Moffett Field, CA, USA
sang.phan@sv.cmu.edu

Pasquale Malacaria
Queen Mary University of London
London, United Kingdom
p.malacaria@qmul.ac.uk

---

Decomposition Instead of Self-Composition for $\kappa$-Safety

Precise Detection of Side-Channel Vulnerabilities using Quantitative Cartesian Hoare Logic

DiffFuzz: Differential Fuzzing for Side-Channel Analysis

Multi-run side-channel analysis using Symbolic Execution and Max-SMT

---

QFuzz: Quantitative Fuzzing for Side Channels
Quantification

Information Leakage: min-entropy [Smith2009]
Assuming that the program $P$ is deterministic and the distribution over secret input $\Sigma$ is uniform, then the information leakage can be characterized $\log_2 k^*$ ($\epsilon=0$).

1. How to identify such inputs?
2. How to characterize observation classes?

Threat Model
Attacker can pick an ideal public input to compromise the secret value or some properties of it in one try.

Problem Statement
Find set of secret values $\Sigma$ and public value $y^*$ that characterize the maximum number of observation classes with the highest distance $\delta$.

Partitioning Algorithm
- $c(s_1, y)$
- $c(s_2, y)$

KDynamic & Greedy
- $c(s_3, y)$
- $c(s_4, y)$

QFuzz: Quantitative Fuzzing for Side Channels
Background: Greybox Fuzzing

1. initial seed files
2. queue
3. trim input
4. mutate repeatedly
5. mutant selection by input evaluation for the instrumented program P
6. mutated files that showed (new) interesting behavior

fuzzing driver

Check for new coverage or program crashes or timeouts

output

program coverage

parse input

execute program P

QFuzz: Quantitative Fuzzing for Side Channels

yannic.noller@acm.org
saeid@utep.edu
**QFuzz: Workflow**

Initial seed files → Queue → Trim input → Mutate repeatedly → Mutated files that showed (new) interesting behavior

- **Step 1:** Initial seed files
- **Step 2:** Queue
- **Step 3:** Trim input
- **Step 4:** Mutate repeatedly
- **Step 5:** Mutant selection by input evaluation for the instrumented program $P$
- **Step 6:** Mutated files that showed (new) interesting behavior

The diagram shows a fuzzing driver that takes initial seed files as input and produces mutated files that showed interesting behavior. The process involves trimming and mutating inputs repeatedly.

Mathematically:

$$\max_{s_1,...,s_K,y} |Part_\epsilon (c(s_1, y), ..., c(s_K, y))| + (1 - e^{-0.1 \times \delta})$$

- **Maximize number of partitions**
- **Maximize the difference between the partitions**

**Variables:**
- $Part_\epsilon$: The number of partitions
- $c(s_i, y)$: Coverage of input
- $\epsilon$: A constant
- $\delta$: The distance between partitions

**Steps:**
- a) #partitions $k$
- b) Minimum distance $\delta$
- c) Program coverage
- d) Check for improved partitioning or coverage
KD\textit{ynamic vs Greedy Partitioning}

\[ K = 1 \]

\[ \varepsilon \]
**KDynamic vs Greedy Partitioning**

$$K = 2$$

Time

$$\epsilon$$

QFuzz: Quantitative Fuzzing for Side Channels
KDyamic vs Greedy Partitioning

\[ K = 3 \]

Time

Find valid partitions with max. delta

\[ \varepsilon, \delta \]

Find valid partitions with guarantees; simple and fast

\[ \leq \varepsilon, > \varepsilon \]
We report the maximum number of partitions (\(K\)) that are vulnerable to adaptive side channels where an attacker can use the side-channel observations related to the content of secret inputs. Each partition is at least \(\delta\) improved the security and reduced the strength of leaks as compared to the unconditional jump included a single extra bytecode instruction as compared to the conditional jump. With \(\varepsilon\) improved the security and reduced the strength of leaks as compared to the conditional jump. With

\[
\begin{align*}
\text{K=17} & \quad \delta=3 \\
\text{K=1} & \quad \delta=1 \\
\text{K=2} & \quad \delta=149 \\
\end{align*}
\]

Only leaks existence of special character.
Evaluation

Research Questions

RQ1 Which partitioning algorithm (KDynamic or Greedy) performs better in terms of correct number of partitions and time for partition computation?

RQ2 How does QFuzz compare with state-of-the-art SC detection techniques like Blazer, Themis, and DifFuzz?

RQ3 Can QFuzz be used for quantification of SC vulnerabilities in real-world Java applications and how does it compare with MaxLeak?

Subjects
- Micro-benchmark
- DARPA STAC
- GitHub projects

Tools/Techniques
- Blazer
- Themis
- DifFuzz
- MaxLeak

Our open-source tool **QFuzz** and all experimental subjects are publicly available:
- [https://github.com/yannicnoller/qfuzz](https://github.com/yannicnoller/qfuzz)
RQ1 KDynamic vs. Greedy

**Computational complexity of Greedy vs KDynamic in isolation**

**Eclipse Jetty (κ=1): temporal development Greedy and KDynamic with 5 different seed inputs combined**

(lines and bands show averages and 95% confidence intervals across 30 repetitions)
RQ2 Detection

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Version</th>
<th>QFuzz</th>
<th>DirFuzz</th>
<th>Time (s)</th>
<th>QFuzz, $\bar{p} &gt; 1$</th>
<th>DirFuzz, $\bar{d} &gt; 0$</th>
<th>Blazer</th>
<th>Themis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Safe</td>
<td></td>
<td>1</td>
<td>$\delta_{\text{max}}$</td>
<td>-</td>
<td>7.40 (+/- 1.21)</td>
<td>1.60</td>
<td>0.28</td>
</tr>
<tr>
<td>Array</td>
<td>Unsafe</td>
<td>2</td>
<td>192</td>
<td>$\delta_{\text{max}}$</td>
<td>195</td>
<td>5.70 (+/- 0.21)</td>
<td>7.40 (+/- 0.93)</td>
<td>0.16</td>
</tr>
<tr>
<td>LoopAndbranch</td>
<td>Safe</td>
<td>2</td>
<td>4</td>
<td>$\delta_{\text{max}}$</td>
<td>4,278,268,702</td>
<td>1045.33 (+/- 43.51)</td>
<td>18.60 (+/- 6.40)</td>
<td>0.23</td>
</tr>
<tr>
<td>LoopAndbranch</td>
<td>Unsafe</td>
<td>2</td>
<td>4</td>
<td>$\delta_{\text{max}}$</td>
<td>4,294,838,782</td>
<td>1078.63 (+/- 61.04)</td>
<td>10.60 (+/- 2.62)</td>
<td>0.65</td>
</tr>
<tr>
<td>Sanity</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0.63</td>
</tr>
<tr>
<td>Sanity</td>
<td>Unsafe</td>
<td>2</td>
<td>3,537,954,359</td>
<td>$\delta_{\text{max}}$</td>
<td>4,290,510,883</td>
<td>1414.13 (+/- 102.27)</td>
<td>163 (+/- 40.63)</td>
<td>0.30</td>
</tr>
<tr>
<td>Straightline</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0.21</td>
</tr>
<tr>
<td>Straightline</td>
<td>Unsafe</td>
<td>2</td>
<td>8</td>
<td>$\delta_{\text{max}}$</td>
<td>8</td>
<td>7.47 (+/- 0.18)</td>
<td>14.60 (+/- 6.53)</td>
<td>22.20</td>
</tr>
<tr>
<td>unixlogin</td>
<td>Safe</td>
<td>-</td>
<td>-</td>
<td>$\delta_{\text{max}}$</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>510 (+/- 91.18)</td>
</tr>
<tr>
<td>unixlogin</td>
<td>Unsafe</td>
<td>2</td>
<td>6,400,000,008</td>
<td>$\delta_{\text{max}}$</td>
<td>3,200,000,008</td>
<td>1784.47 (+/- 21.27)</td>
<td>464.20 (+/- 64.61)</td>
<td>0.77</td>
</tr>
<tr>
<td>modPow1</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>1.47</td>
</tr>
<tr>
<td>modPow1</td>
<td>Unsafe</td>
<td>22</td>
<td>117</td>
<td>$\delta_{\text{max}}$</td>
<td>3,068</td>
<td>4.73 (+/- 0.16)</td>
<td>4.80 (+/- 1.11)</td>
<td>218.54</td>
</tr>
<tr>
<td>modPow2</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>1.62</td>
</tr>
<tr>
<td>modPow2</td>
<td>Unsafe</td>
<td>31</td>
<td>1</td>
<td>$\delta_{\text{max}}$</td>
<td>5,206</td>
<td>294.70 (+/- 104.66)</td>
<td>23.00 (+/- 3.48)</td>
<td>7813.68</td>
</tr>
<tr>
<td>passwordEq</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>0.00</td>
<td>-</td>
<td>-</td>
<td>2.70</td>
</tr>
<tr>
<td>passwordEq</td>
<td>Unsafe</td>
<td>93</td>
<td>2</td>
<td>$\delta_{\text{max}}$</td>
<td>127</td>
<td>4.57 (+/- 0.22)</td>
<td>8.60 (+/- 2.11)</td>
<td>1.30</td>
</tr>
<tr>
<td>k96</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{\text{max}}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0.70</td>
</tr>
<tr>
<td>k96</td>
<td>Unsafe</td>
<td>93</td>
<td>2</td>
<td>$\delta_{\text{max}}$</td>
<td>3,087,339</td>
<td>4.57 (+/- 0.22)</td>
<td>3.40 (+/- 0.98)</td>
<td>1.29</td>
</tr>
<tr>
<td>gpt14</td>
<td>Safe</td>
<td>12</td>
<td>1</td>
<td>$\delta_{min}$</td>
<td>517</td>
<td>5.00 (+/- 0.00)</td>
<td>4.20 (+/- 0.80)</td>
<td>1.43</td>
</tr>
<tr>
<td>gpt14</td>
<td>Unsafe</td>
<td>92</td>
<td>2</td>
<td>$\delta_{min}$</td>
<td>12,965,890</td>
<td>5.87 (+/- 0.12)</td>
<td>4.40 (+/- 1.03)</td>
<td>219.30</td>
</tr>
<tr>
<td>login</td>
<td>Safe</td>
<td>1</td>
<td>0</td>
<td>$\delta_{min}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>1.77</td>
</tr>
<tr>
<td>login</td>
<td>Unsafe</td>
<td>17</td>
<td>2</td>
<td>$\delta_{min}$</td>
<td>62</td>
<td>7.77 (+/- 0.69)</td>
<td>10.00 (+/- 2.92)</td>
<td>1.79</td>
</tr>
</tbody>
</table>

QFuzz: Quantitative Fuzzing for Side Channels

- same vulnerabilities detected
- additional information about the strength of leaks and the exploitability of vulnerabilities
- large values for $K$ may slow down QFuzz, but eventually, enable the exploration of many partitions
RQ3 Quantification

<table>
<thead>
<tr>
<th>Modulo</th>
<th>Len</th>
<th>#Partitions</th>
<th>$\bar{T}$</th>
<th>$P_{max}$</th>
<th>Time (s): $P_{max}$</th>
<th>$t_{min}$</th>
<th>MaxLeak (default)</th>
<th>#Obs</th>
<th>Time (s)</th>
<th>MaxLeak (No solver)</th>
<th>#Obs</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1717</td>
<td>5</td>
<td>7</td>
<td>7.00 (+/- 0.00)</td>
<td>7</td>
<td>1.00 (+/- 0.00)</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>20.892</td>
<td>12</td>
<td>1.370</td>
<td>1.370</td>
</tr>
<tr>
<td>1717</td>
<td>4</td>
<td>10</td>
<td>10.00 (+/- 0.00)</td>
<td>10</td>
<td>7.43 (+/- 0.45)</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>152.332</td>
<td>12</td>
<td>1.370</td>
<td>1.370</td>
</tr>
<tr>
<td>1717</td>
<td>5</td>
<td>13</td>
<td>13.00 (+/- 0.00)</td>
<td>13</td>
<td>20.40 (+/- 3.87)</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>839.788</td>
<td>15</td>
<td>2.916</td>
<td>2.916</td>
</tr>
<tr>
<td>1717</td>
<td>6</td>
<td>16</td>
<td>16.00 (+/- 0.00)</td>
<td>16</td>
<td>294.60 (+/- 53.17)</td>
<td>22</td>
<td>15</td>
<td>18</td>
<td>3731.328</td>
<td>18</td>
<td>8.006</td>
<td>8.006</td>
</tr>
<tr>
<td>1717</td>
<td>7</td>
<td>19</td>
<td>18.37 (+/- 0.25)</td>
<td>19</td>
<td>2484.30 (+/- 451.42)</td>
<td>385</td>
<td>&gt; 4 h</td>
<td>21</td>
<td>19.241</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
</tr>
<tr>
<td>1717</td>
<td>8</td>
<td>22</td>
<td>20.43 (+/- 0.45)</td>
<td>22</td>
<td>3168.07 (+/- 303.47)</td>
<td>508</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
</tr>
<tr>
<td>1717</td>
<td>9</td>
<td>25</td>
<td>22.20 (+/- 0.36)</td>
<td>24</td>
<td>3489.03 (+/- 169.19)</td>
<td>1009</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
</tr>
<tr>
<td>1717</td>
<td>10</td>
<td>28</td>
<td>24.40 (+/- 0.49)</td>
<td>27</td>
<td>3548.63 (+/- 57.73)</td>
<td>2929</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>91.821</td>
</tr>
<tr>
<td>834443</td>
<td>3</td>
<td>7</td>
<td>7.00 (+/- 0.00)</td>
<td>7</td>
<td>13.40 (+/- 1.96)</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td>7.416</td>
<td>12</td>
<td>1.188</td>
<td>1.188</td>
</tr>
<tr>
<td>834443</td>
<td>4</td>
<td>10</td>
<td>10.00 (+/- 0.00)</td>
<td>10</td>
<td>40.33 (+/- 12.14)</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>42.684</td>
<td>12</td>
<td>1.385</td>
<td>1.385</td>
</tr>
<tr>
<td>834443</td>
<td>5</td>
<td>13</td>
<td>12.93 (+/- 0.09)</td>
<td>13</td>
<td>645.70 (+/- 329.43)</td>
<td>74</td>
<td>12</td>
<td>15</td>
<td>215.929</td>
<td>15</td>
<td>2.953</td>
<td>2.953</td>
</tr>
<tr>
<td>834443</td>
<td>6</td>
<td>16</td>
<td>15.40 (+/- 0.20)</td>
<td>16</td>
<td>2711.87 (+/- 433.23)</td>
<td>271</td>
<td>15</td>
<td>18</td>
<td>936.921</td>
<td>18</td>
<td>7.511</td>
<td>7.511</td>
</tr>
<tr>
<td>834443</td>
<td>7</td>
<td>19</td>
<td>16.80 (+/- 0.33)</td>
<td>18</td>
<td>3227.60 (+/- 275.29)</td>
<td>952</td>
<td>18</td>
<td>21</td>
<td>4021.150</td>
<td>21</td>
<td>19.068</td>
<td>19.068</td>
</tr>
<tr>
<td>834443</td>
<td>8</td>
<td>22</td>
<td>17.93 (+/- 0.54)</td>
<td>22</td>
<td>3556.70 (+/- 83.44)</td>
<td>2301</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
</tr>
<tr>
<td>834443</td>
<td>9</td>
<td>25</td>
<td>20.13 (+/- 0.59)</td>
<td>24</td>
<td>3572.83 (+/- 37.16)</td>
<td>3110</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
</tr>
<tr>
<td>834443</td>
<td>10</td>
<td>28</td>
<td>21.83 (+/- 0.46)</td>
<td>24</td>
<td>3504.13 (+/- 121.70)</td>
<td>1845</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>96.360</td>
</tr>
<tr>
<td>1964903306</td>
<td>3</td>
<td>7</td>
<td>6.47 (+/- 0.18)</td>
<td>7</td>
<td>2228.30 (+/- 542.13)</td>
<td>119</td>
<td>6</td>
<td>9</td>
<td>12.167</td>
<td>9</td>
<td>1.085</td>
<td>1.085</td>
</tr>
<tr>
<td>1964903306</td>
<td>4</td>
<td>10</td>
<td>8.67 (+/- 0.19)</td>
<td>10</td>
<td>3494.30 (+/- 203.69)</td>
<td>429</td>
<td>9</td>
<td>12</td>
<td>70.805</td>
<td>12</td>
<td>1.535</td>
<td>1.535</td>
</tr>
<tr>
<td>1964903306</td>
<td>5</td>
<td>13</td>
<td>10.70 (+/- 0.19)</td>
<td>12</td>
<td>3594.00 (+/- 11.56)</td>
<td>3420</td>
<td>12</td>
<td>15</td>
<td>2306.261</td>
<td>15</td>
<td>3.391</td>
<td>3.391</td>
</tr>
<tr>
<td>1964903306</td>
<td>6</td>
<td>16</td>
<td>12.90 (+/- 0.11)</td>
<td>13</td>
<td>1337.90 (+/- 443.89)</td>
<td>206</td>
<td>&gt; 4 h</td>
<td>18</td>
<td>7.506</td>
<td>&gt; 4 h</td>
<td>18</td>
<td>7.506</td>
</tr>
<tr>
<td>1964903306</td>
<td>7</td>
<td>19</td>
<td>14.10 (+/- 0.27)</td>
<td>15</td>
<td>2984.67 (+/- 362.05)</td>
<td>503</td>
<td>&gt; 4 h</td>
<td>21</td>
<td>19.486</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
</tr>
<tr>
<td>1964903306</td>
<td>8</td>
<td>22</td>
<td>15.33 (+/- 0.36)</td>
<td>17</td>
<td>3398.37 (+/- 204.45)</td>
<td>1411</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
</tr>
<tr>
<td>1964903306</td>
<td>9</td>
<td>25</td>
<td>16.30 (+/- 0.51)</td>
<td>19</td>
<td>3562.33 (+/- 54.24)</td>
<td>2819</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
</tr>
<tr>
<td>1964903306</td>
<td>10</td>
<td>28</td>
<td>17.30 (+/- 0.48)</td>
<td>20</td>
<td>3559.67 (+/- 77.72)</td>
<td>2390</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
<td>&gt; 4 h</td>
<td>24</td>
<td>98.325</td>
</tr>
</tbody>
</table>

due to its dynamic analysis, **QFuzz is more scalable than MaxLeak**

**QFuzz has precision comparable to MaxLeak that uses symbolic execution with model counting**

even for complex scenarios **QFuzz provides reasonable lower-bound guarantees**
QFuzz: Quantitative Fuzzing for Side Channels

stringEquals (Original Jetty, v1)

```java
boolean stringEquals(String s1, String s2) {
    if (s1 == s2) {
        return true;
    }
    if (s1 == null || s2 == null || s1.length() != s2.length()) {
        return false;
    }
    for (int i = 0; i < s1.length(); ++i) {
        if (s1.charAt(i) != s2.charAt(i)) {
            return false;
        }
    }
    return true;
}
```

conditional early return causes leakage

Detection vs Quantification

- Is there a vulnerability?
- How much information can be leaked?

Challenges

- How to go beyond non-interference?
- How to avoid expensive symbolic execution?
- How to scale to larger programs?
- How to provide guarantees for vulnerability?

Maximize number of partitions
Maximize the difference between the partitions

mutated files that showed (new) interesting behavior

mutant selection by input evaluation for the instrumented program P

additional information about the strength of leaks and the exploitability of vulnerabilities
due to its dynamic analysis, QFuzz is more scalable than MaxLeak

QFuzz has precision comparable to MaxLeak that uses symbolic execution with model counting
even for complex scenarios QFuzz provides reasonable lower-bound guarantees