CONFETTI: Amplifying Concolic Guidance for Fuzzers

James Kukucka, Luís Pina, Paul Ammann, Jonathan Bell
Motivation - CVE-2021-45105 log4j DoS Vulnerability

CVE-2021-44832: New Vulnerability Found in Apache Log4j

Summary

A new vulnerability was discovered in the Apache Log4j library. Tracked as CVE-2021-44832, this bug may allow arbitrary code execution in compromised systems when the attacker has permissions to modify the logging configuration file.

CVE-2021-44832 has received a CVSS score of 6.6 out of 10, and it affects all versions of Log4j from 2.0-alpha7 to 2.17.0, excluding 2.3.2 and 2.12.4. This is the fourth Log4j vulnerability addressed by Apache in December 2021, followed by:

- CVE-2021-45105: Vulnerability that could allow DoS attacks (CVSS 5.9)
- CVE-2021-45046: Vulnerability that could allow Remote Code Execution (CVSS 9.0)
- CVE-2021-44228: Vulnerability that could allow Remote Code Execution (CVSS 10.0)

```java
protected boolean substitute(final LogEvent event, final StringBuilder buf, final int offset, final int length) {
    return substitute(event, buf, offset, length, null) > 0;
}
```

Introduction: Parametric Fuzzers vs Greybox Fuzzers

Seed: <xml></xml>

Guidance

Coverage Instrumentation

System Under Test

Application Logic

Syntactic Parser

Mutator

Concrete Input: <xml>^/xml>

Biases
State-of-the-art

Seed: 0001

Coverage Instrumentation

Guidance

Bias

Parametric Input: 0011

Mutator

System Under Test

Application Logic

Syntactic Parser

Concrete Input:

<xml>
  <name>
    value
  </name>
</xml>

String generateXML() {
    String tagName = generateString();
    if (generateBoolean())
        return "<" + tagName + " +
            generateString() + ":=" +
            generateString() + "</" + tagName + ">");
    return "<" + tagName + ">" + "</" + tagName + ">";
}
Our Solution, CONFETTI, leverages state-of-the-art parametric fuzzing and novel hinting

CONFETTI (CONcolic Fuzzer Employing Taint Tracking Information)
Our Solution, CONFETTI, leverages state-of-the-art parametric fuzzing and novel hinting
Our Solution, CONFETTI, leverages state-of-the-art parametric fuzzing and novel hinting

CONFETTI (CONcolic Fuzzer Employing Taint Tracking Information)

- Targeted Hinting
  - Taint Tracking
  - Concolic Execution
- Parametric Fuzzing
Our Solution, CONFETTI, leverages state-of-the-art parametric fuzzing and novel hinting.
Example of applying targeted hints

```java
String generateXML() {
    String tagName = generateString();
    if (generateBoolean())
        return "" + tagName + "=" +
            generateString() + ">
            + generateString() + "";
    return "" + tagName + "";
}
```

(a) Simple parametric XML tag generator that uses 4 random choices

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Parametric Input</th>
<th>Generator Choices</th>
<th>New Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>seed</td>
<td>01011010</td>
<td>&quot;groupId&quot;</td>
<td>False</td>
</tr>
</tbody>
</table>

```java
byte[] input = generateXML();
XMLDocument doc = parse(input);
if (!"expected".equals(doc.getElementById(0).getName()))
    throw new Error();
String v = doc.getElementById(0).getAttribute("version");
if (v == null) throw new Error();
```
Example of applying targeted hints

```java
String generateXML(){
    String tagName = generateString();
    if(generateBoolean())
        return "<"+tagName+" "+
            generateString() +"="+
            generateString() +"</"+tagName+">");
    return "<"+tagName+"></"+tagName+">");
}
```

(a) Simple parametric XML tag generator that uses 4 random choices

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Parametric Input</th>
<th>Generator Choices</th>
<th>New Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>seed</td>
<td>01011010</td>
<td>&quot;groupId&quot;False</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>mutation</td>
<td>01011110</td>
<td>&quot;package&quot;False</td>
<td>-</td>
</tr>
</tbody>
</table>
Example of applying targeted hints

```java
String generateXML(){
    String tagName = generateString();
    if(generateBoolean())
        return "<"+tagName+" "+
            generateString() +"="+
            generateString() +" ></"+tagName+"";
    return "<"+tagName+" /></"+tagName+"";
}
```

(a) Simple parametric XML tag generator that uses 4 random choices

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Parametric Input</th>
<th>Generator Choices</th>
<th>New Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>seed</td>
<td>01011010</td>
<td>&quot;groupId&quot;false</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>mutation</td>
<td>01011110</td>
<td>&quot;package&quot;false</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>hint</td>
<td>10010110</td>
<td>&quot;expected&quot;false</td>
<td>6</td>
</tr>
</tbody>
</table>

```java
byte[] input = generateXML();
XMLElement doc = parse(input);
if (!"expected".equals(doc.getElementById(0).getName()))
    throw new Error();
String v = doc.getElementById(0).getAttribute("version");
if (v == null) throw new Error();
```
Example of applying targeted hints

```java
String generateXML()
{
    String tagName = generateString();
    if(generateBoolean())
        return "<"+tagName+ "=" +
                generateString() +"=" +
                generateString() +"></"+tagName+">";
    return "<"+tagName+"></"+tagName+">
;
}
```

(a) Simple parametric XML tag generator that uses 4 random choices

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Parametric Input</th>
<th>Generator Choices</th>
<th>New Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>seed</td>
<td>01011010</td>
<td>&quot;groupId&quot;false</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>mutation</td>
<td>01011110</td>
<td>&quot;package&quot;false</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>hint</td>
<td>00010110</td>
<td>&quot;expected&quot;false</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>mutation</td>
<td>001011110110101101</td>
<td>&quot;expected&quot;true &quot;groupId&quot; &quot;A&quot;</td>
<td>7, 8</td>
</tr>
</tbody>
</table>

```java
1byte[] input = generateXML();
2XMLDocument doc = parse(input);
3
4if (!"expected".equals(doc.getElementById(0).getName()))
5    throw new Error();
6
7String v = doc.getElementById(0).getAttribute("version");
8if (v == null) throw new Error();
9```
Example of applying targeted hints

```java
String generateXML(){
    String tagName = generateString();
    if(generateBoolean())
        return ""+tagName+"="+
            generateString()+""+
            generateString()+""+""+""+"";
    return ""+tagName+""+""+"";
}
```

(a) Simple parametric XML tag generator that uses 4 random choices

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Parametric Input</th>
<th>Generator Choices</th>
<th>New Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>seed</td>
<td>01011010</td>
<td>&quot;groupId&quot;false</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>mutation</td>
<td>01011110</td>
<td>&quot;package&quot;false</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>hint</td>
<td>00010110</td>
<td>&quot;expected&quot;false</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>mutation</td>
<td>000101110110101101</td>
<td>&quot;expected&quot;true &quot;groupId&quot;&quot;A&quot;</td>
<td>7, 8</td>
</tr>
<tr>
<td>5</td>
<td>hint</td>
<td>000101110111101101</td>
<td>&quot;expected&quot;true &quot;version&quot;&quot;A&quot;</td>
<td>9</td>
</tr>
</tbody>
</table>

```java
byte[] input = generateXML();
XMLDocument doc = parse(input);
if (!"expected".equals(doc.getElementById(0).getName()))
    throw new Error();
String v = doc.getElementById(0).getAttribute("version");
if (v == null) throw new Error();
```
CONFETTI uses a non-blocking Architecture
CONFETTI uses a non-blocking architecture

1. Generate and Execute Inputs
2. Interesting input
3. Analyze new input
4. Constraints + taint flows
5. Solve negated constraints
6. New inputs
7. New input(s) & hints for original input

Whitebox Analysis (Knarr)
CONFETTI Coordinator
SMT Solver
Greybox Fuzzer
KNARR builds path conditions by propagating taint tags from parametric bytes to concrete input

- KNARR is able to taint parametric input bytes and propagate taint tags with minimal changes to underlying generators.
- Strings are tainted at the character level, and operations such as `equals()` and `startsWith()` are instrumented.
- KNARR extends the taint engine to create an abstract expression as part of the taint tag, building it as taints are propagated to new variables.
- When a tainted input reaches a branch, the taint tag of the branch is the complete symbolic expression from the parametric input.
- KNARR facilitates concolic execution in this way, as opposed to pure symbolic execution.
The CONFETTI coordinator ingests constraints from KNARR to attempt to discover new branches

- In the style of concolic execution, the CONFETTI Coordinator targets branches based on whether they are uncovered and whether their branch predicate contains some part of the input
- Branch is negated and all other constraints are dropped, then it is passed to Z3.
- Helpful to cover branches the fuzzer got stuck on.
- User-configurable parameters to cut down on wasted solving time.
Taint Tracking Doesn’t Capture Relationships Through Control Flow

```java
public void magic(String s1, String s2){
    boolean v1 = s1.equals("abc");
    boolean v2 = s2.equals(s1.concat("def"));
    if(v1 && v2)
        throw new IllegalStateException(); // Bug
}```
Global hinting allows CONFETTI to explore branches it could not otherwise.

```java
public void magic(String s1, String s2){
    boolean v1 = s1.equals("abc");
    boolean v2 = s2.equals(s1.concat("def"));
    if(v1 && v2)
        throw new IllegalStateException(); // Bug
}
```
Global hinting allows CONFETTI to explore branches it could not otherwise.

```java
public void magic(String s1, String s2) {
    boolean v1 = s1.equals("abc");
    boolean v2 = s2.equals(s1.concat("def"));
    if (v1 && v2)
        throw new IllegalStateException(); // Bug
}
```

```java
public String generateString(ParametricInputArray r) {
    if (r.nextBoolean())
        return static_dict[r.nextInt()];
    return global_hints[r.nextInt()];
}
```

s1 = generateString(r); // picks randomly from static dictionary to yield “abc”
s2 = generateString(r); // picks randomly from global hints to yield “abcdef”
CONFETTI leverages both targeted and global hints in guiding the fuzzer

● CONFETTI does not seek to purely perform whitebox analysis, but to *guide* the fuzzing process so that it maximizes the efficiency of greybox fuzzing.
● It does this by leveraging several choices when mutating an input:
  ○ Apply a single targeted hint
  ○ Apply multiple targeted hints
  ○ Mutate, which may or may not apply global hints
● Hints are *inheritable* meaning they are preserved in future generations (if an input reveals new coverage and is fuzzed again).
● Stacking hints allows for more complex inputs that may reveal new coverage.
On most benchmark programs, the use of CONFETTI’s global hinting with targeted hinting resulted in higher branch coverage and more bugs found.

<table>
<thead>
<tr>
<th>Program</th>
<th>Total Branches</th>
<th>Total Branch Coverage</th>
<th>Bugs Found</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Zest</td>
<td>CONFETTI_tgt</td>
</tr>
<tr>
<td>ant</td>
<td>23,361</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>859</td>
<td>871</td>
</tr>
<tr>
<td>bcel</td>
<td>6,220</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1361</td>
<td>1423</td>
</tr>
<tr>
<td>closure</td>
<td>49,602</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10,545</td>
<td>10,640</td>
</tr>
<tr>
<td>maven</td>
<td>5,858</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>821</td>
<td>853</td>
</tr>
<tr>
<td>rhino</td>
<td>25,035</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,757</td>
<td>3,534</td>
</tr>
</tbody>
</table>
CONFETTI finds more bugs, including bugs that the baseline fuzzer cannot

<table>
<thead>
<tr>
<th>Issue #</th>
<th>JQF-Zest</th>
<th>CONFETTI_tgt</th>
<th>CONFETTI</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>B1</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>B3</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>B4</td>
<td>0</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>B5</td>
<td>0</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>B6</td>
<td>0</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>R1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>R2</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>R3</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>R4</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue #</th>
<th>JQF-Zest</th>
<th>CONFETTI_tgt</th>
<th>CONFETTI</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>C2</td>
<td>90</td>
<td>85</td>
<td>5</td>
</tr>
<tr>
<td>C3</td>
<td>80</td>
<td>70</td>
<td>45</td>
</tr>
<tr>
<td>C4</td>
<td>0</td>
<td>45</td>
<td>95</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
<td>15</td>
<td>90</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C7</td>
<td>0</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>C8</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>C9</td>
<td>15</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>C10</td>
<td>0</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>C11</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>C12</td>
<td>0</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>C13</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>C14</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C15</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
Our evaluation, all data and CONFETTI are archived and open-source

https://doi.org/10.6084/m9.figshare.16563776

https://github.com/neu-se/confetti
Continuous Integration workflow allows for easy evaluation
Our Solution, CONFETTI, leverages state-of-the-art parametric fuzzing and novel hinting.

Confetti (CONcolic Fuzzer Employing Taint Tracking Information)

**Targeted Hinting**

**Global Hinting**

**Parametric Fuzzing**

**Taint Tracking**

**Concolic Execution**

Global hinting allows CONFETTI to explore branches it could not otherwise.

```java
public void magic(String st, String s2) {
  boolean v1 = s1.equals("abc");
  boolean v2 = s2.equals(s1.concat("def"));
  if (v1 && v2)
    throw new IllegalStateException(); // Bug
}
```

public String generateString(ParametricInputArray r) {
  if (r.nextBoolean())
    return static_dic[r.nextInt()];
  return global_hints[r.nextInt()];
}

s1 = generateString(); // picks randomly from static dictionary to yield "abc"
s2 = generateString(); // picks randomly from global hints to yield "abcdef"

CONFETTI finds more bugs, including bugs that the baseline fuzzer cannot.

<table>
<thead>
<tr>
<th>Issue #</th>
<th>JQF-Zest</th>
<th>CONFETTI, tgt</th>
<th>CONFETTI</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>B1</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>B3</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>B4</td>
<td>0</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>B5</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>B6</td>
<td>0</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue #</th>
<th>JQF-Zest</th>
<th>CONFETTI, tgt</th>
<th>CONFETTI</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>C2</td>
<td>90</td>
<td>85</td>
<td>5</td>
</tr>
<tr>
<td>C3</td>
<td>80</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>C4</td>
<td>90</td>
<td>45</td>
<td>90</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
<td>15</td>
<td>90</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C7</td>
<td>0</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>C8</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>C9</td>
<td>15</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>C10</td>
<td>0</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>C11</td>
<td>0</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>C12</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>C13</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C14</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C15</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Our evaluation, all data and CONFETTI are archived and open-source.

https://doi.org/10.6084/m9.figshare.16563776

https://github.com/neu-sel/confetti