Code-Analyse und Grammar-based Fuzzing

An Unholy Alliance

Hannes Sochor
hannes.sochor@scch.at
Software Competence Center Hagenberg GmbH

Josef Pichler
josef.pichler@fh-hagenberg.at
Fachhochschule Oberösterreich Campus Hagenberg

Rudolf Ramler
rudolf.ramler@scch.at
Software Competence Center Hagenberg GmbH

SCCH ist eine Initiative der JKU Johannes Kepler University Linz

FFG Forschung wirkt.

Bundesministerium Digitalisierung und Wirtschaftsstandort

Bundesministerium Verkehr, Innovation und Technologie

Regio 13

Member of UAR Innovation Network

SCCH ist Teil des softwarepark hagenberg

Software Competence Center Hagenberg

- Anwendungsorientiertes Forschungszentrum in Software Science und Data Science

- Partner für Industrie und Wirtschaft
- Internationale Forschung und Zusammenarbeit
- COMET-Kompetenzzentrum
- Gegründet 1999
- Ca. 70 Mitarbeiter
- Standort Hagenberg und Linz

JOHANNES KEPLER UNIVERSITÄT LINZ

softwarepark hagenberg
business research education

© Software Competence Center Hagenberg 2019

scch
software competence center hagenberg

scch
software competence center
hagenberg

Dissertationen und Masterarbeiten

Forschungs-projekte

Forschungskooperationen und multi-firm Projekte

4+Jahre, >20 Mio €

Technologie- und Wissenstransfer

Wissenschaftliche Partner

Unternehmens-partner

scch
software competence center
hagenberg

Warum eigentlich Fuzzing?

© Software Competence Center Hagenberg 2019

Bye-bye bugs

Over 16,000 bugs later, Google's fuzz tester is now open source

February 11, 2019 Sarah Schlothauer

Twitter Facebook Google+

Billions and Billions
Whitebox Fuzz Testi

Ella Bounimova
Microsoft Research, USA

Patrice Go
Microsoft Res

Abstract—We report experiences with constraint-based white-box fuzz testing in production across hundreds of large Windows applications and over 500 machine years of computation from 2007 to 2013. Whitebox fuzzing leverages symbol
on binary traces and constraint solvi
to a program. These i
solved with a *constrai*
to new i

acmqueue

SAGE: Whitebox Fuzzing for Security Testing

SAGE has had a remarkable impact at Microsoft.

Godefroid, Michael Y. Levin, David Molnar, Microsoft

m verification research" as mostly theoretical with
lines on a PC running some

ZDNet

Researchers find 36 new security flaws in LTE protocol

South Korean researchers apply fuzzing techniques to LTE protocol and find 51 vulnerabilities, of which 36 were new.

By Catalin Cimpanu for Zero Day | March 23, 2019 -- 08:00 GMT (08:00 GMT) | Topic: Security

MIDDLE EAST SCANDINAVIA AFRICA UK ITALY SPAIN MORE NEWSLETTERS ALL W

Security

Hot fuzz: Bug detectives whip up smarter version of classic AFL fuzzer to hunt code vulnerabilities

Flaw-spotting toolkit already has 42 zero-days to its name

By Shaun Nichols in San Francisco 28 Nov 2018 at 08:03 3 SHARE ▼

scch
software competence center
hagenberg

Fuzzing Success Stories

AFL	compile-time instrumentation, genetic algorithms	Vulnerabilities in Firefox, IE, Safari, Adobe Flash, sqlite, OpenSSL, LibreOffice, OpenSSH, PuTTY, tcpdump, JavaScript, Wireshark, Android, iOS, LLVM, Perl, VLC, Adobe Reader, Tor, MySQL, Linux Kernel, ... [1]
ClusterFuzz	Highly scalable (~25000 cores), Support for AFL and libFuzzer	~16,000 Bugs in Chrome, ~11,000 Bugs in Open Source Projects [2]
libFuzzer	coverage guided mutation	Vulnerabilities in SQLite, Python, OpenSLL, Linux Kernel, LLVM, Tensorflow, Wireshark, and more ... [3]
langfuzz	grammar-based code generation & code mutation	~4000 Bugs in JavaScript interpreter, 105 Vulnerabilities in Firefox [4]

[1] http://lcamtuf.coredump.cx/afl/
[2] https://google.github.io/clusterfuzz/
[3] http://llvm.org/docs/LibFuzzer.html
[4] https://issta2016.cispa.saarland/interview-with-christian-holler/

© Software Competence Center Hagenberg 2019

Was passiert eigentlich beim Fuzzing?

© Software Competence Center Hagenberg 2019

Beispiel: Expression Parser

Parser für einfache mathematische Ausdrücke

```
> expr 3
1
> expr 3 + 2
3
> expr (3 + 2) * 5
7
> expr (3 + 2) * 5 - 2)
-1
...
```

```java
public class ExprParser {

    public static int parse(char[] in) {

        int pos = expr(0, in);
        if (pos < in.length) {
            syntaxError("End of input", pos);
            pos = -1;
        }
        return pos;
    }

    public static int expr(int pos, char[] in) {
        pos = term(pos, in);
        if (pos == -1)
            return pos;
        if (pos == in.length)
            return pos;

        if (in[pos] == '+') {
            pos++;
            pos = term(pos, in);
        } else if (in[pos] == '-') {
            pos++;
            pos = term(pos, in);
        }

        return pos;
    }

    public static int term(int pos, char[] in) {
        pos = factor(pos, in);
        if (pos == -1)
            return pos;
        if (pos == in.length)
            return pos;

        if (in[pos] == '*') {
            pos++;
            pos = factor(pos, in);
        } else if (in[pos] == '/') {
            pos++;
            pos = factor(pos, in);
        }

        return pos;
    }
```

```java
    public static int factor(int pos, char[] in) {
        if (pos == in.length)
            return -1;

        if (in[pos] == '1')
            return pos + 1;
        if (in[pos] == '2')
            return pos + 1;
        if (in[pos] == '3')
            return pos + 1;
        if (in[pos] == '(') {
            pos++;
            pos = expr(pos, in);
            if (pos == -1)
                return pos;
            if (pos == in.length)
                return -1;
            if (in[pos] == ')') {
                pos++;
                return pos;
            } else {
                syntaxError("invalid factor ", pos);
            }
        }
        return -1;
    }

    public static void syntaxError(String msg, int pos) {
        final boolean log = false;
        if (log) {
            System.out.print("Syntax error ");
            System.out.print(msg);
            System.out.print(" col: ");
            System.out.println(pos);
        }
    }
}
```

© Software Competence Center Hagenberg 2019

Beispiel: Expression Parser

Random Fuzzing

Input	k3iou5j
Code Coverage	37,8%

```
5  public class ExprParser {
6
7      public static int parse(char[] in) {
8
9          int pos = expr(0, in);
10         if (pos < in.length) {
11             syntaxError("End of input", pos);
12             pos = -1;
13         }
14         return pos;
15     }
16
17     public static int expr(int pos, char[] in) {
18         pos = term(pos, in);
19         if (pos == -1)
20             return pos;
21         if (pos == in.length)
22             return pos;
23
24         if (in[pos] == '+') {
25             pos++;
26             pos = term(pos, in);
27         } else if (in[pos] == '-') {
28             pos++;
29             pos = term(pos, in);
30         }
31
32         return pos;
33     }
34
35     public static int term(int pos, char[] in) {
36         pos = factor(pos, in);
37         if (pos == -1)
38             return pos;
39         if (pos == in.length)
40             return pos;
41
42         if (in[pos] == '*') {
43             pos++;
44             pos = factor(pos, in);
45         } else if (in[pos] == '/') {
46             pos++;
47             pos = factor(pos, in);
48         }
49
50         return pos;
51     }
```

```
53     public static int factor(int pos, char[] in) {
54         if (pos == in.length)
55             return -1;
56
57         if (in[pos] == '1')
58             return pos + 1;
59         if (in[pos] == '2')
60             return pos + 1;
61         if (in[pos] == '3')
62             return pos + 1;
63         if (in[pos] == '(') {
64             pos++;
65             pos = expr(pos, in);
66             if (pos == -1)
67                 return pos;
68             if (pos == in.length)
69                 return -1;
70             if (in[pos] == ')') {
71                 pos++;
72                 return pos;
73             } else {
74                 syntaxError("invalid factor ", pos);
75             }
76         }
77         return -1;
78     }
79
80     public static void syntaxError(String msg, int pos) {
81         final boolean log = false;
82         if (log) {
83             System.out.print("Syntax error ");
84             System.out.print(msg);
85             System.out.print(" col: ");
86             System.out.println(pos);
87         }
88     }
89 }
```

© Software Competence Center Hagenberg 2019

Beispiel: Expression Parser

Random Fuzzing

Input	k3iou5j
Code Coverage	37,8%

Grammar-based Fuzzing

S	=	**Expr**
Expr	=	**Term** [("+" | "-") **Term**]
Term	=	**Factor** [("*" | "/") **Factor**]
Factor	=	"*1*" | "*2*" | "*3*" | "*(*" **Expr** "*)*"

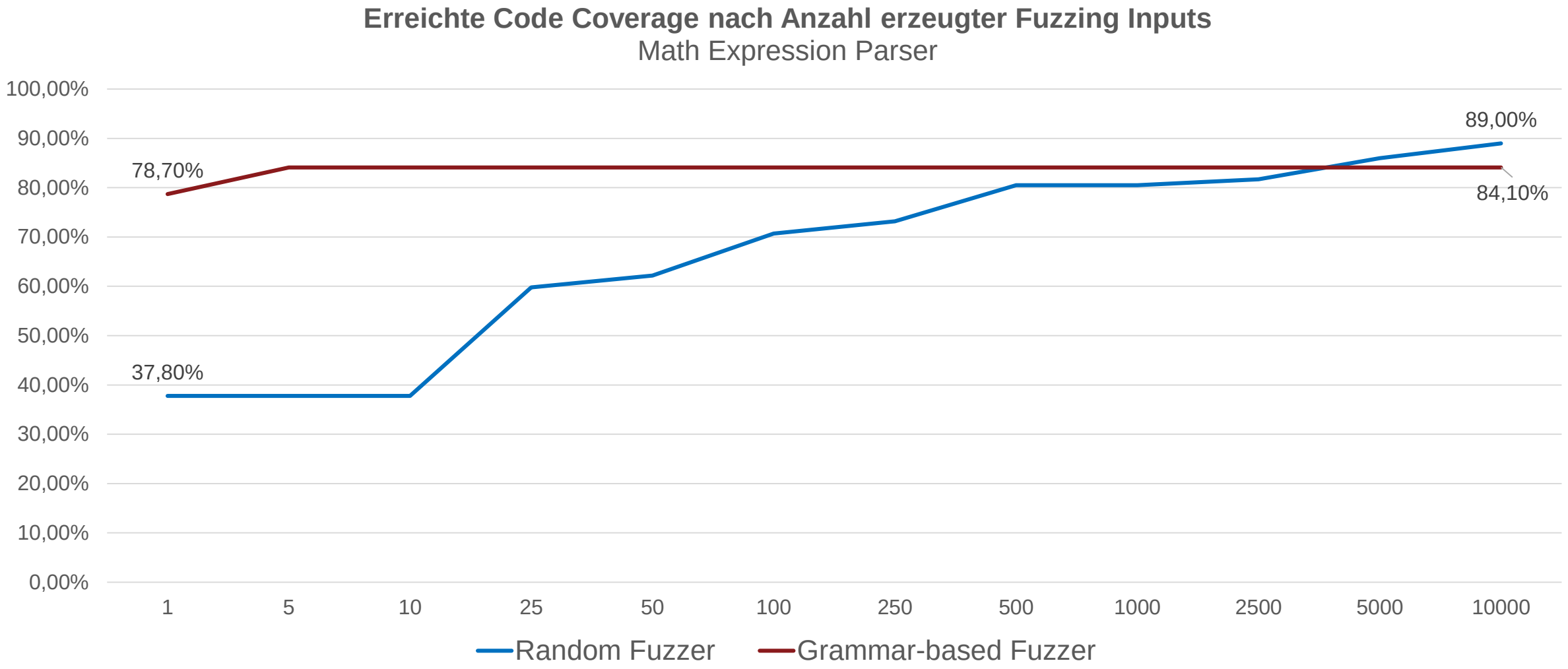
Input	(1+3)/2
Code Coverage	78,7%

```
5  public class ExprParser {
6
7      public static int parse(char[] in) {
8
9          int pos = expr(0, in);
10         if (pos < in.length) {
11             syntaxError("End of input", pos);
12             pos = -1;
13         }
14         return pos;
15     }
16
17     public static int expr(int pos, char[] in) {
18         pos = term(pos, in);
19         if (pos == -1)
20             return pos;
21         if (pos == in.length)
22             return pos;
23
24         if (in[pos] == '+') {
25             pos++;
26             pos = term(pos, in);
27         } else if (in[pos] == '-') {
28             pos++;
29             pos = term(pos, in);
30         }
31
32         return pos;
33     }
34
35     public static int term(int pos, char[] in) {
36         pos = factor(pos, in);
37         if (pos == -1)
38             return pos;
39         if (pos == in.length)
40             return pos;
41
42         if (in[pos] == '*') {
43             pos++;
44             pos = factor(pos, in);
45         } else if (in[pos] == '/') {
46             pos++;
47             pos = factor(pos, in);
48         }
49
50         return pos;
51     }
```

```
53     public static int factor(int pos, char[] in) {
54         if (pos == in.length)
55             return -1;
56
57         if (in[pos] == '1')
58             return pos + 1;
59         if (in[pos] == '2')
60             return pos + 1;
61         if (in[pos] == '3')
62             return pos + 1;
63         if (in[pos] == '(') {
64             pos++;
65             pos = expr(pos, in);
66             if (pos == -1)
67                 return pos;
68             if (pos == in.length)
69                 return -1;
70             if (in[pos] == ')') {
71                 pos++;
72                 return pos;
73             } else {
74                 syntaxError("invalid factor ", pos);
75             }
76         }
77         return -1;
78     }
79
80     public static void syntaxError(String msg, int pos) {
81         final boolean log = false;
82         if (log) {
83             System.out.print("Syntax error ");
84             System.out.print(msg);
85             System.out.print(" col: ");
86             System.out.println(pos);
87         }
88     }
89 }
```

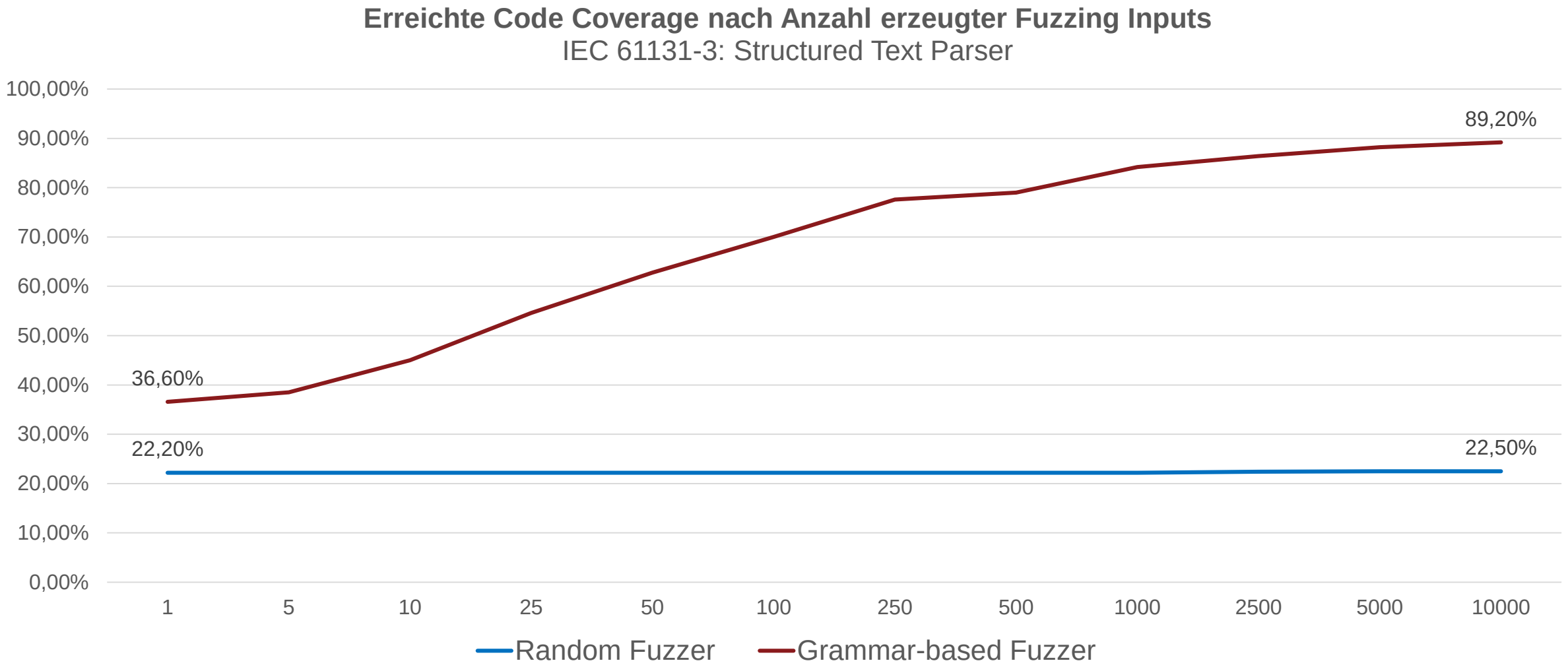
© Software Competence Center Hagenberg 2019

Random Fuzzing VS Grammar-based Fuzzing



© Software Competence Center Hagenberg 2019

Random Fuzzing VS Grammar-based Fuzzing



© Software Competence Center Hagenberg 2019

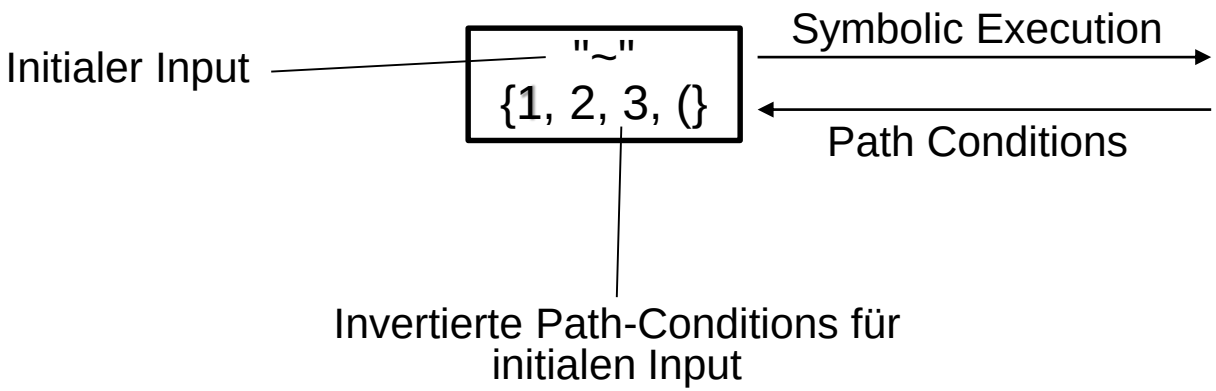
Alles schön und gut, aber woher kommt die Grammatik?

© Software Competence Center Hagenberg 2019

Grammar Mining

© Software Competence Center Hagenberg 2019

Grammar Mining – Schritt 1: Suche nach gültigen Inputs




```java
public class ExprParser {

    public static int parse(char[] in) {

        int pos = expr(0, in);
        if (pos < in.length) {
            syntaxError("End of input", pos);
            pos = -1;
        }
        return pos;
    }

    public static int expr(int pos, char[] in) {
        pos = term(pos, in);
        if (pos == -1)
            return pos;
        if (pos == in.length)
            return pos;

        if (in[pos] == '+') {
            pos++;
            pos = term(pos, in);
        } else if (in[pos] == '-') {
            pos++;
            pos = term(pos, in);
        }

        return pos;
    }

    public static int term(int pos, char[] in) {
        pos = factor(pos, in);
        if (pos == -1)
            return pos;
        if (pos == in.length)
            return pos;

        if (in[pos] == '*') {
            pos++;
            pos = factor(pos, in);
        } else if (in[pos] == '/') {
            pos++;
            pos = factor(pos, in);
        }

        return pos;
    }
```

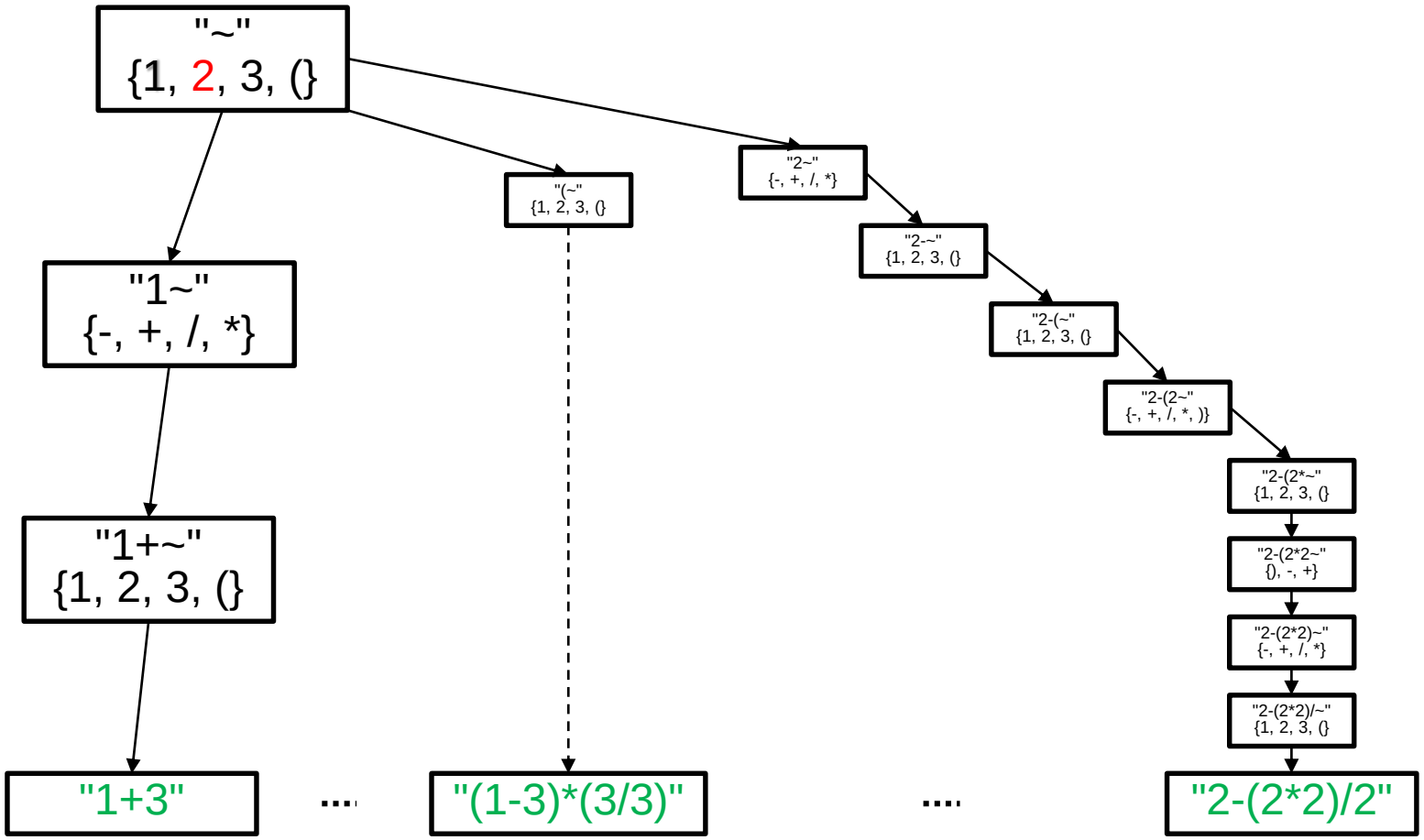
```java
    public static int factor(int pos, char[] in) {
        if (pos == in.length)
            return -1;

        if (in[pos] == '1')
            return pos + 1;
        if (in[pos] == '2')
            return pos + 1;
        if (in[pos] == '3')
            return pos + 1;
        if (in[pos] == '(') {
            pos++;
            pos = expr(pos, in);
            if (pos == -1)
                return pos;
            if (pos == in.length)
                return -1;
            if (in[pos] == ')') {
                pos++;
                return pos;
            } else {
                syntaxError("invalid factor ", pos);
            }
        }
        return -1;
    }

    public static void syntaxError(String msg, int pos) {
        final boolean log = false;
        if (log) {
            System.out.print("Syntax error ");
            System.out.print(msg);
            System.out.print(" col: ");
            System.out.println(pos);
        }
    }
}
```

© Software Competence Center Hagenberg 2019

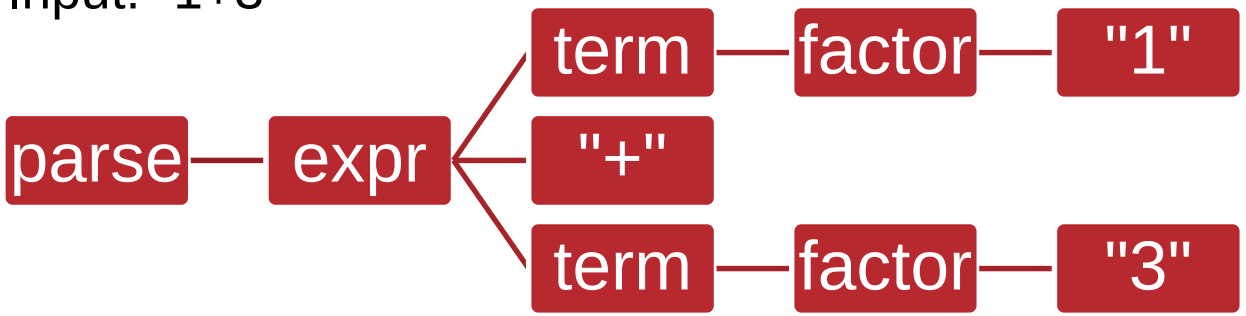
Grammar Mining – Schritt 1: Suche nach gültigen Inputs



© Software Competence Center Hagenberg 2019

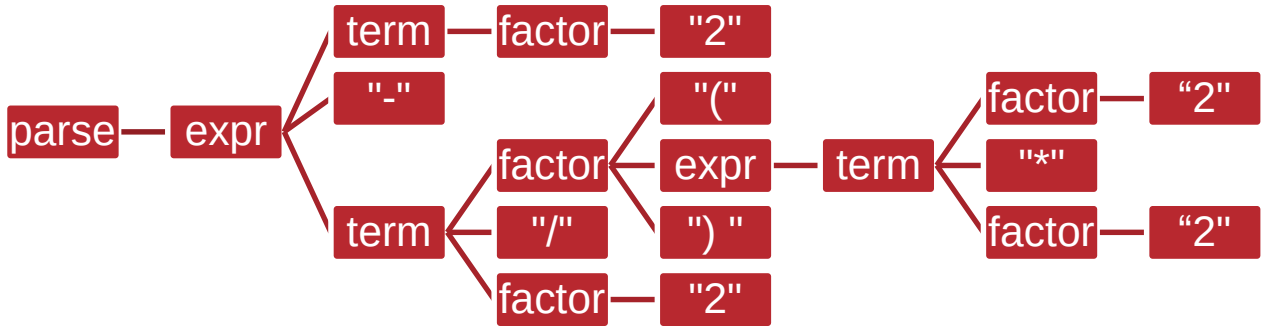
Grammar Mining – Schritt 2: Ableiten der Parse-Trees

Input: "1+3"



parse	=	**expr**
expr	=	**term** "+" **term**
term	=	**factor**
factor	=	"*1*" | "*3*"

Input: "2-(2*2)/2"



parse	=	**expr**
expr	=	**term** "-" **term** | **term**
term	=	**factor** | **factor** "*" **factor** | **factor** "/" **factor**
factor	=	"*2*" | "*(*" **expr** ")"

© Software Competence Center Hagenberg 2019

Grammar Mining – Schritt 3: Zusammenfügen der Grammatik

```
parse   =  expr
expr    =  term "+" term
term    =  factor
factor  =  "1" | "3"
```



```
parse   =  expr
expr    =  term "-" term | term
term    =  factor | factor "*" factor | factor "/" factor
factor  =  "2" | "(" expr ")"
```

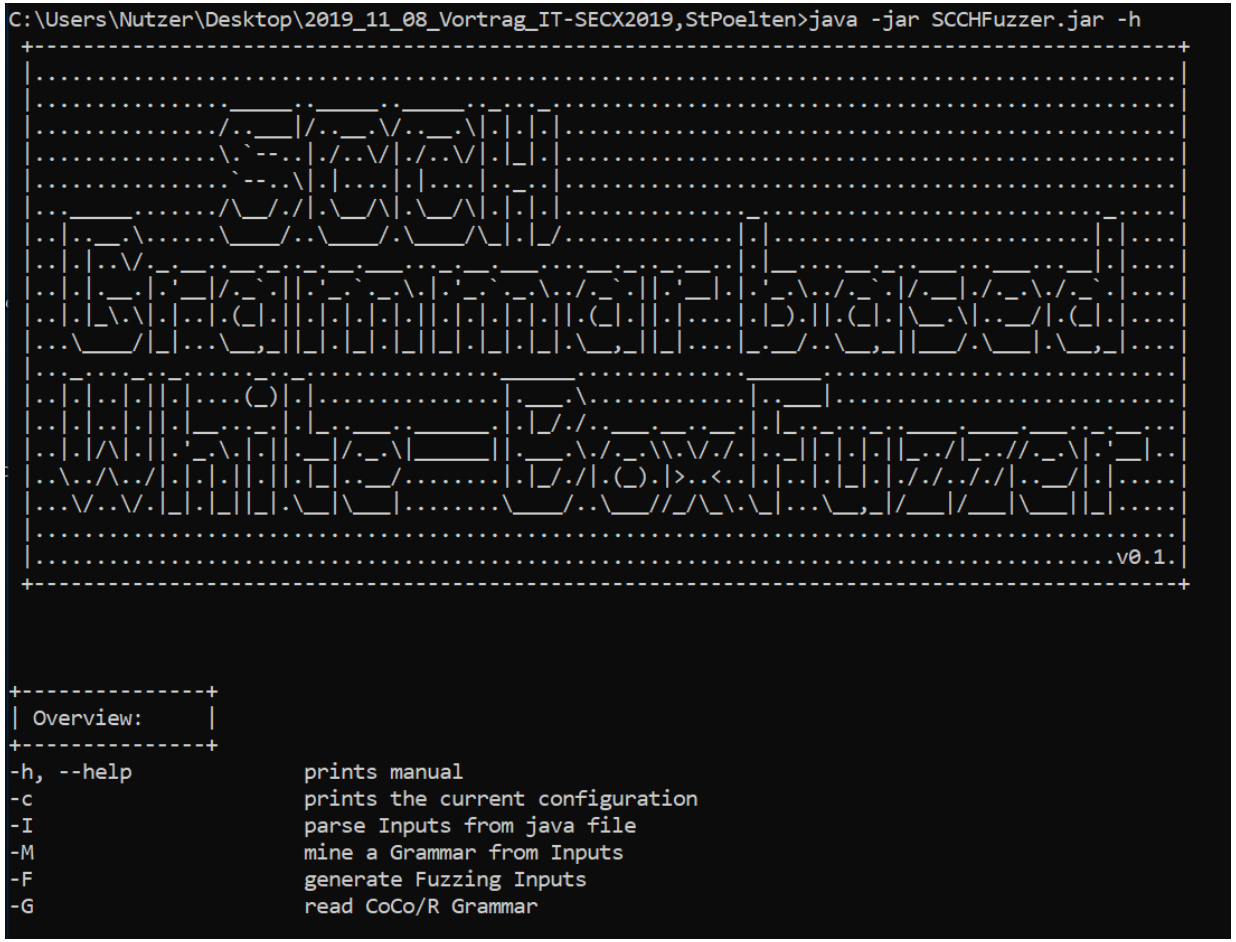
```
parse   =  expr
expr    =  term "+" term | term "-" term | term
term    =  factor | factor "*" factor | factor "/" factor
factor  =  "1" | "3" | "2" | "(" expr ")"
```

```
S       =  Expr
Expr    =  Term [ ( "+" | "-" ) Term ]
Term    =  Factor [ ( "*" | "/" ) Factor ]
Factor  =  "1" | "2" | "3" | "(" Expr ")"
```

© Software Competence Center Hagenberg 2019

Und wie sieht das jetzt live aus?

Demo


C:\Users\Nutzer\Desktop\2019_11_08_Vortrag_IT-SECX2019,StPoelten>java -jar SCCHFuzzer.jar -h
v0.1.
+---------------+
| Overview: |
+---------------+
-h, --help prints manual
-c prints the current configuration
-I parse Inputs from java file
-M mine a Grammar from Inputs
-F generate Fuzzing Inputs
-G read CoCo/R Grammar

Vielen Dank!

Hannes Sochor MSc

Researcher
Secure Software Analytics

Software Competence Center Hagenberg GmbH
Softwarepark 21, A-4232 Hagenberg, Austria
eMail: hannes.sochor@scch.at
phone: +43 50 343 851
http://www.scch.at

© Software Competence Center Hagenberg 2019