RUHR UNIVERSITÄT BOCHUM RUB

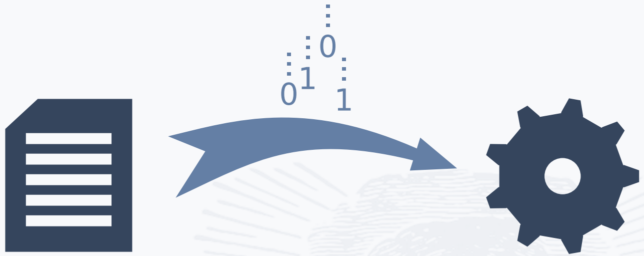
Nautilus

Fishing for Deep Bugs with Grammars

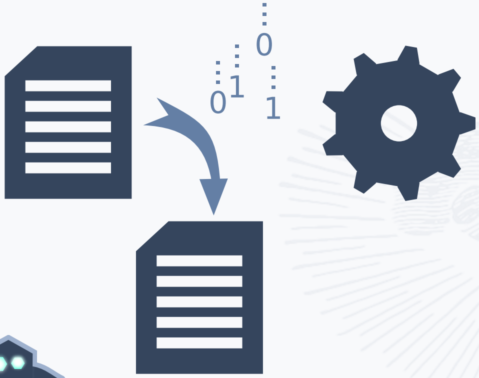
Cornelius Aschermann, Tommaso Frassetto, Thorsten Holz,
Patrick Jauernig, Ahmad-Reza Sadeghi and Daniel Teuchert
Ruhr-Universität Bochum & Technische Universität Darmstadt

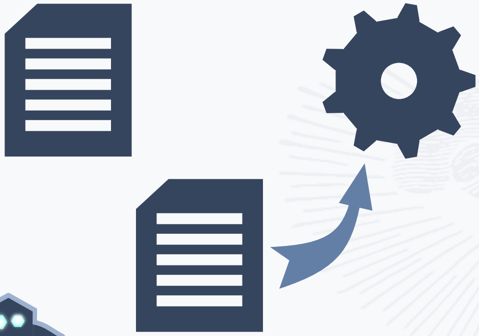
Fuzzing

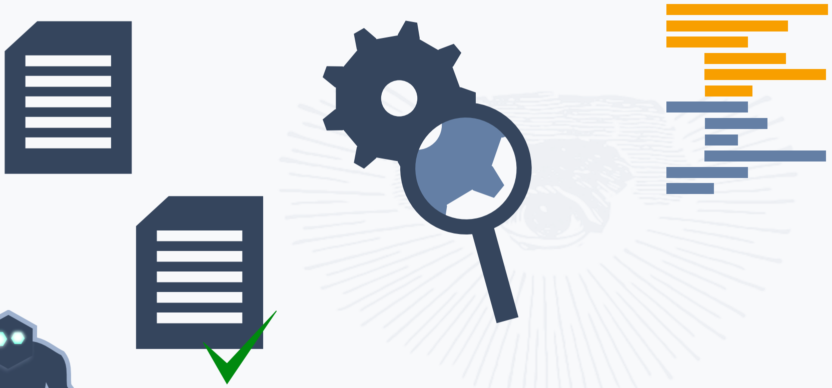
Daniel Teuchert

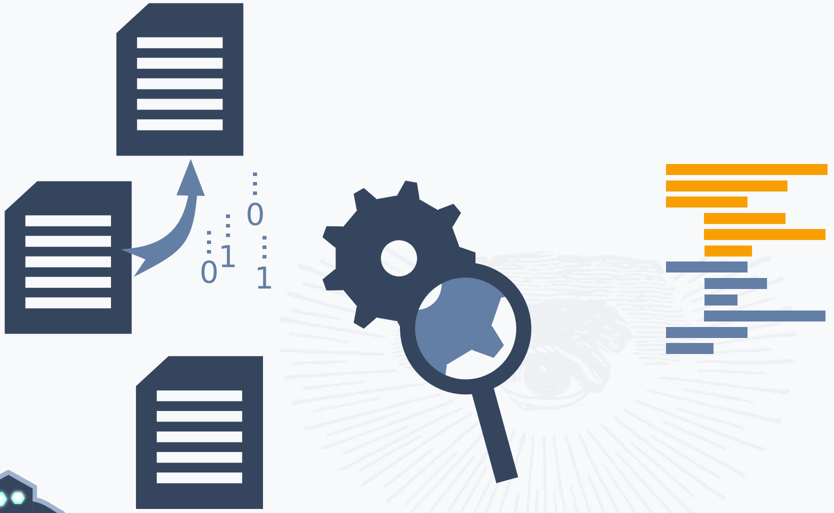

0
1
0
1

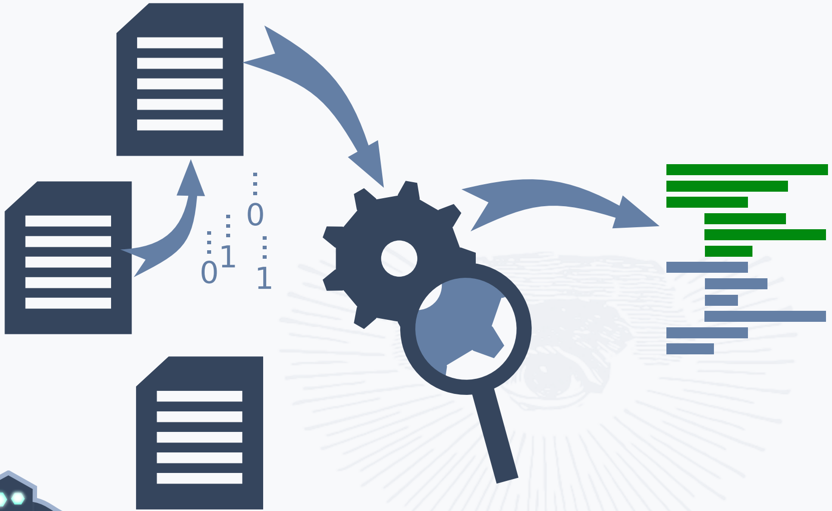
AFL

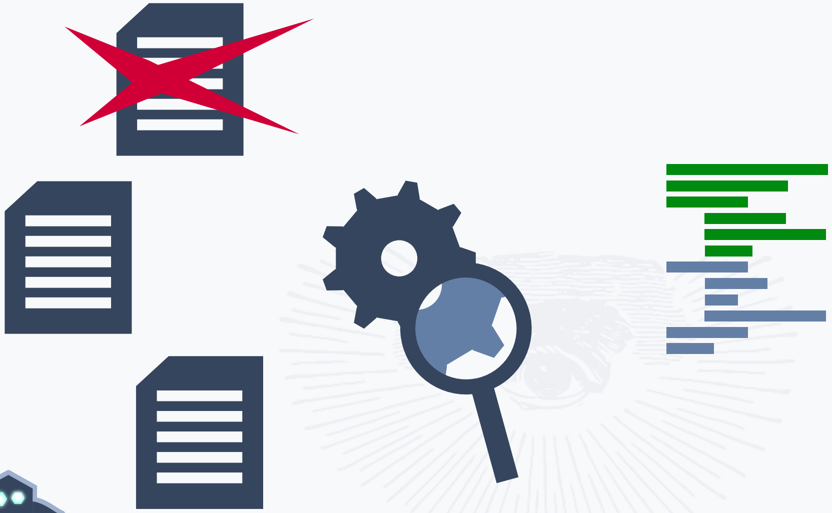

0
1
0
1

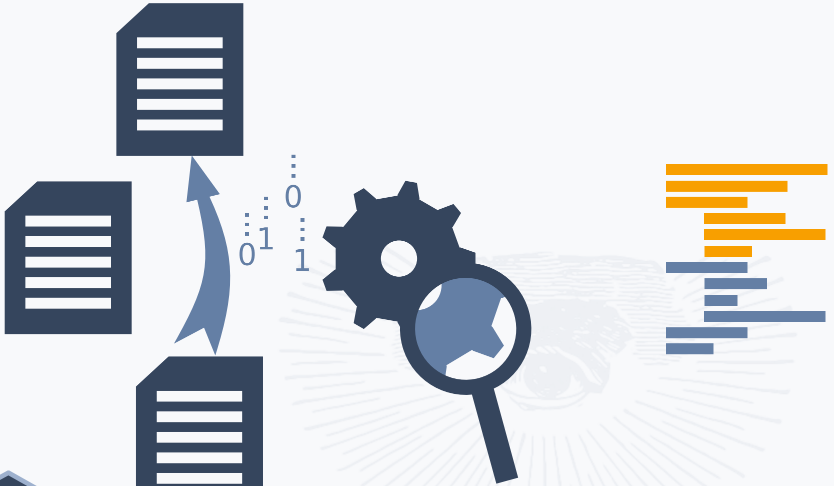


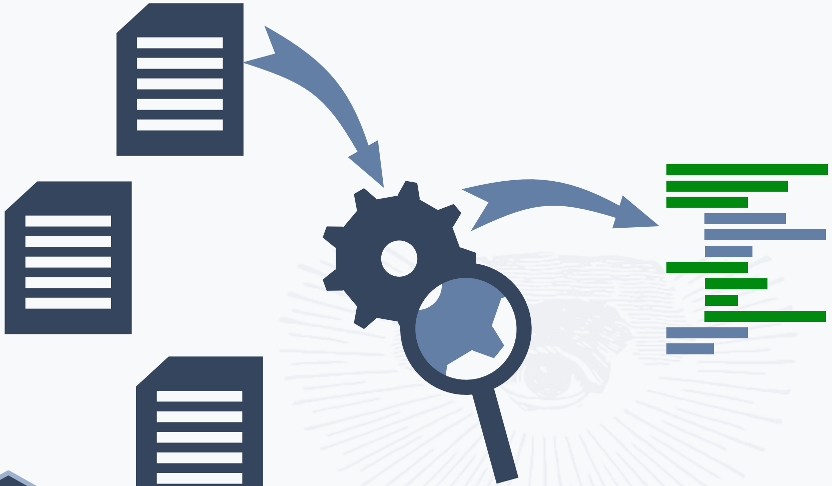


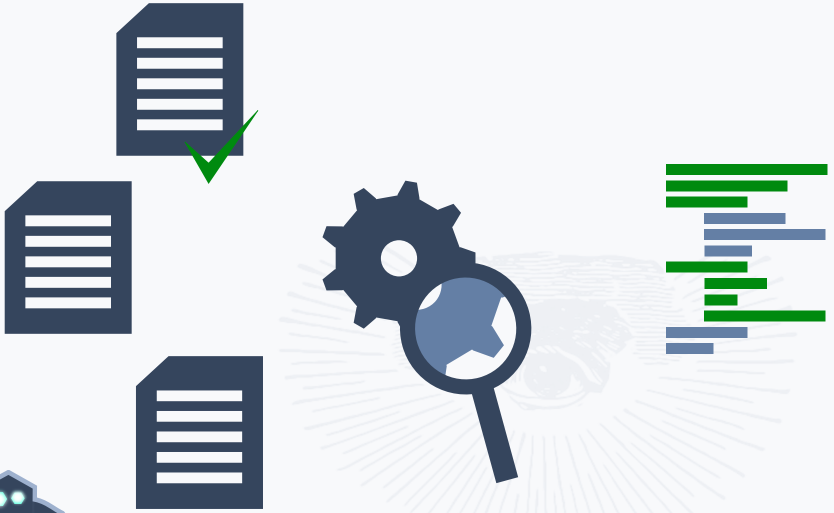

0
1
0
1

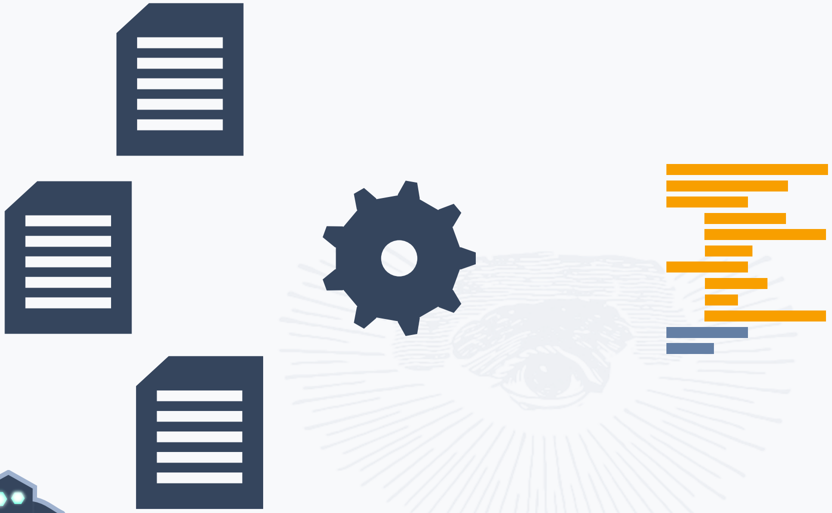

0
1
0
1




0
1
0
1



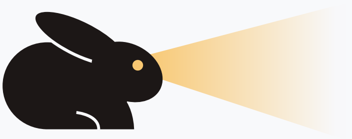




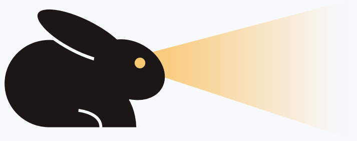
```
if !input.parse() {
    exit()
}
```

```
if !input.parse() {
      exit()
}

if !input.check() {
    exit()
}
```

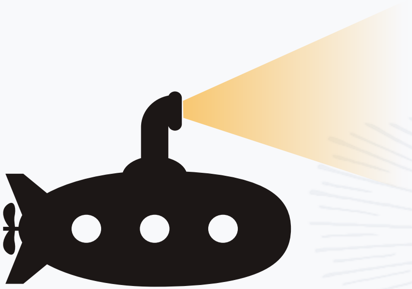
```
if !input.parse() {
      exit()
}

if !input.check() {
    exit()
}

do_stuff()
```



```
if !input.parse() {
      exit()
}

if !input.check() {
    exit()
}

do_stuff()
```



```
if !input.parse() {
      exit()
}

if !input.check() {
    exit()
}

do_stuff()
```



Grammars
+
Feedback

Context-Free Grammars

Context-Free Grammars

PROG	→	STMT
PROG	→	STMT ; PROG
STMT	→	return 1
STMT	→	VAR = EXPR
VAR	→	a
EXPR	→	NUM
EXPR	→	EXPR + EXPR
NUM	→	1 | 2

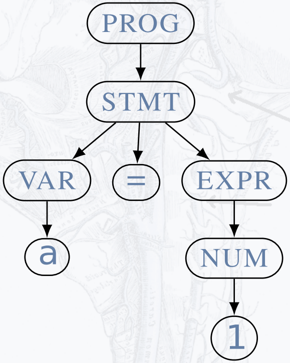
RUHR UNIVERSITÄT BOCHUM RUB

Context-Free Grammars

PROG	→	STMT
PROG	→	STMT ; PROG
STMT	→	return 1
STMT	→	VAR = EXPR
VAR	→	a
EXPR	→	NUM
EXPR	→	EXPR + EXPR
NUM	→	1 | 2

Context-Free Grammars

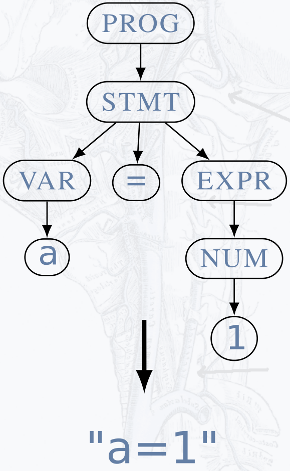
PROG	→	STMT
PROG	→	STMT ; PROG
STMT	→	return 1
STMT	→	VAR = EXPR
VAR	→	a
EXPR	→	NUM
EXPR	→	EXPR + EXPR
NUM	→	1 | 2



$$\begin{tikzcd}
 & \text{PROG} \arrow[d] & \\
 & \text{STMT} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{VAR} \arrow[d] & = & \text{EXPR} \arrow[d] \\
a & & \text{NUM} \arrow[d] \\
 & & 1
\end{tikzcd}$$

Context-Free Grammars

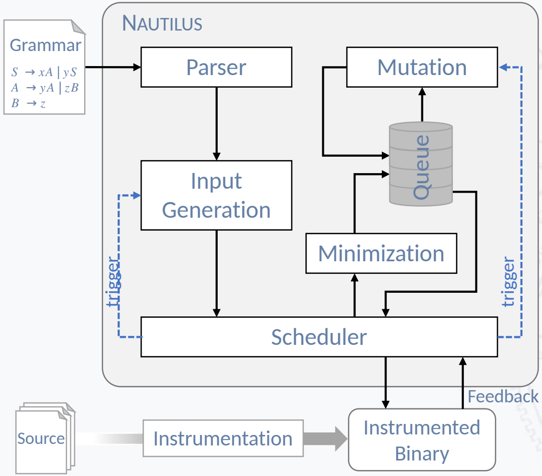
PROG	→	STMT
PROG	→	STMT ; PROG
STMT	→	return 1
STMT	→	VAR = EXPR
VAR	→	a
EXPR	→	NUM
EXPR	→	EXPR + EXPR
NUM	→	1 | 2

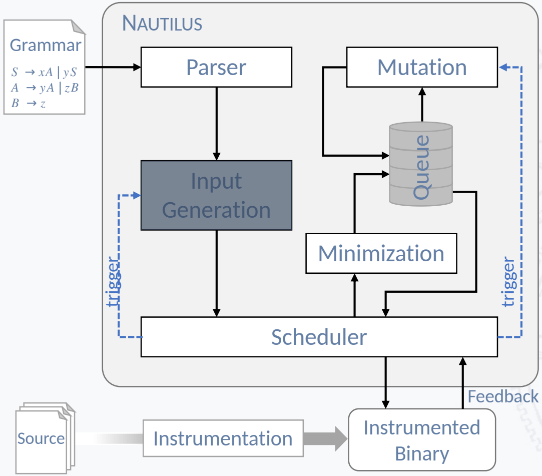


$$\begin{tikzcd}
 & \boxed{\text{PROG}} \arrow[d] & \\
 & \boxed{\text{STMT}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{VAR}} \arrow[d] & \boxed{=} & \boxed{\text{EXPR}} \arrow[d] \\
\boxed{\text{a}} & & \boxed{\text{NUM}} \arrow[d] \\
 & \arrow[d] & \boxed{1} \\
 & \text{"a=1"} &
\end{tikzcd}$$

Design of Nautilus

Design of Nautilus




Grammar
$S \rightarrow xA \mid yS$
$A \rightarrow yA \mid zB$
$B \rightarrow z$
NAUTILUS
Parser
Mutation
Input Generation
Queue
Minimization
trigger
trigger
Scheduler
Feedback
Source
Instrumentation
Instrumented Binary

Generation:

Generation:

-Naive Generation

Generation:

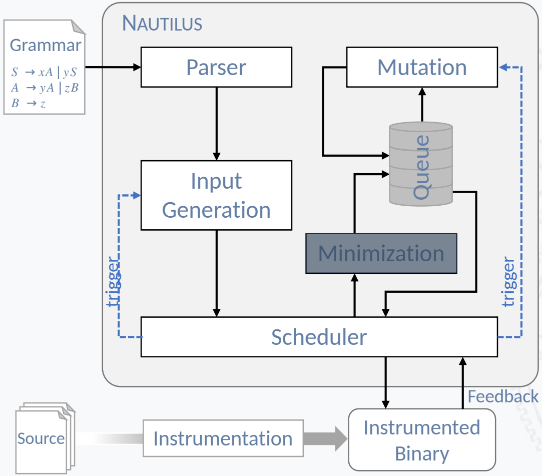
-Naive Generation

PROG	→	STMT
PROG	→	STMT ; PROG
STMT	→	return 1
STMT	→	VAR = EXPR
VAR	→	a
EXPR	→	NUM
EXPR	→	EXPR + EXPR
NUM	→	1 | 2

Generation:

-Naive Generation

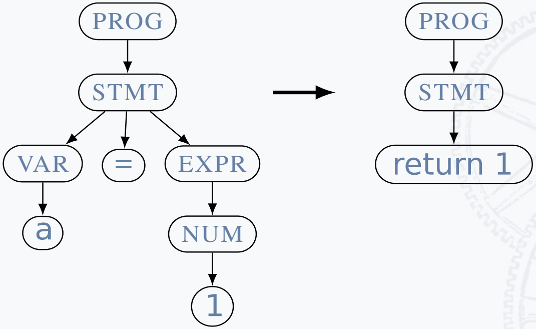
-Uniform Generation


Grammar
$S \rightarrow xA \mid yS$
$A \rightarrow yA \mid zB$
$B \rightarrow z$
NAUTILUS
Parser
Input Generation
Mutation
Queue
Minimization
Scheduler
trigger
trigger
Feedback
Source
Instrumentation
Instrumented Binary

Minimization:

Minimization:

-Subtree Minimization

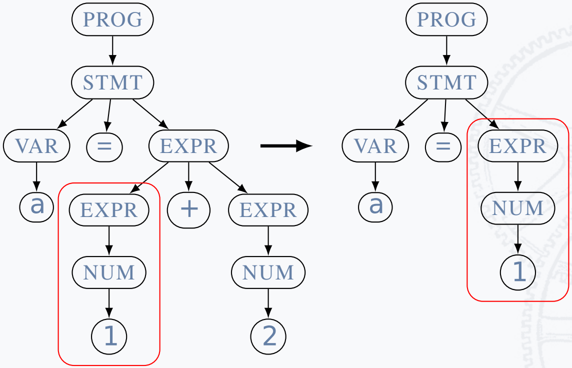


Subtree Minimization

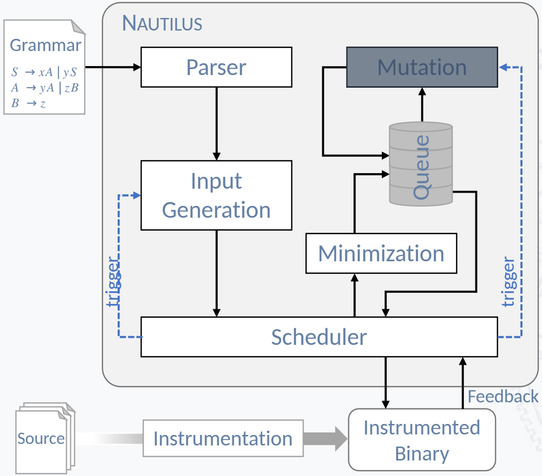
Minimization:

-Subtree Minimization

-Recursion Minimization



Recursive Minimization


NAUTILUS
Grammar
$S \to xA \mid yS$
$A \to yA \mid zB$
$B \to z$
Parser
Input Generation
Mutation
Queue
Minimization
Scheduler
trigger
trigger
Feedback
Source
Instrumentation
Instrumented Binary

Mutation:

Mutation:

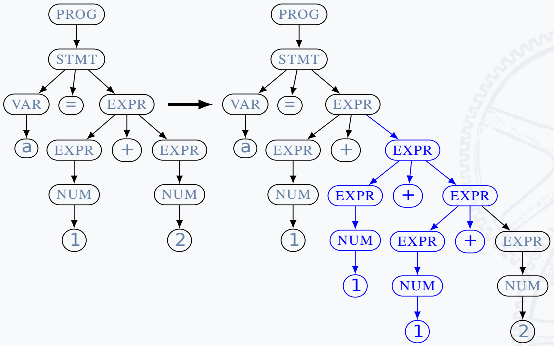
-Random

Mutation:

- -Random
- -Rules

Mutation:

- Random
- Rules
- Random Recursive



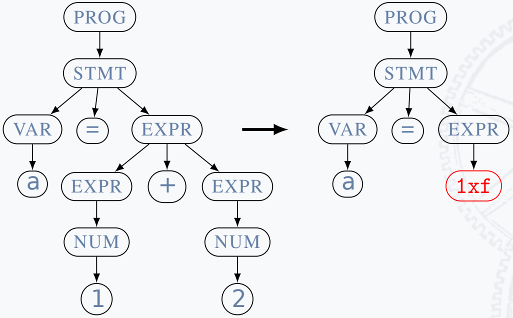
Random Recursive Mutation

Mutation:

- -Random
- -Rules
- -Random Recursive
- -Splicing

Mutation:

- -Random
- -Rules
- -Random Recursive
- -Splicing
- -AFL



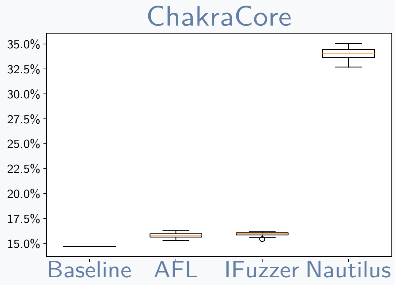
AFL Mutation

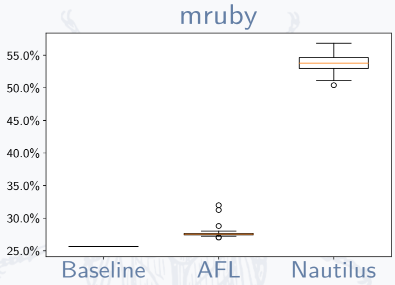
Evaluation

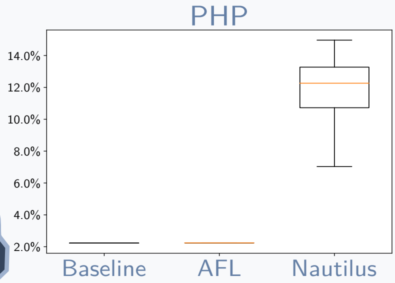
Targets:

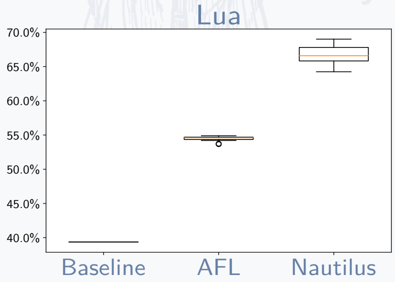
-mruby
-PHP
-lua
-ChackraCore

vs. AFL / IFuzzer

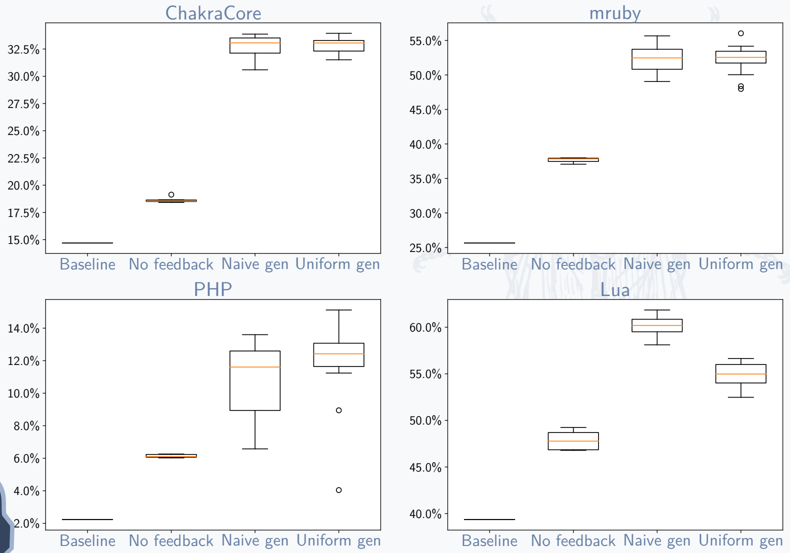


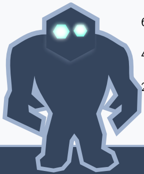


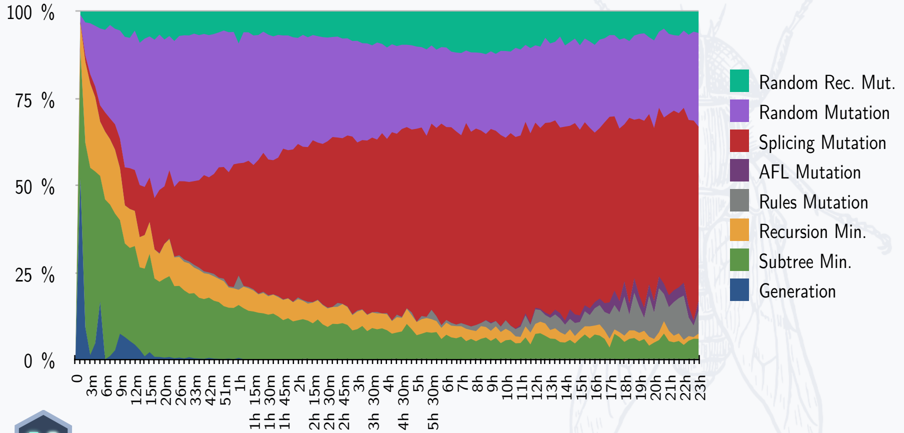




Configurations







100 %
75 %
50 %
25 %
0 %
0
3m
6m
9m
12m
15m
20m
26m
33m
42m
51m
1h
1h 15m
1h 30m
1h 45m
2h
2h 15m
2h 30m
2h 45m
3h
3h 30m
4h
4h 30m
5h
5h 30m
6h
7h
8h
9h
10h
11h
12h
13h
14h
15h
16h
17h
18h
19h
20h
21h
22h
23h
Random Rec. Mut.
Random Mutation
Splicing Mutation
AFL Mutation
Rules Mutation
Recursion Min.
Subtree Min.
Generation

```
ObjectSpace.each do |a|
    begin
        a.method(...)
    rescue
    end
end
```

Bugs?

mruby:

- CVE-2018-10191: UAF
- CVE-2018-10199: UAF
- CVE-2018-11743: Use of Uninitialized Pointer
- CVE-2018-12249: SEGV
- CVE-2018-12247: SEGV
- CVE-2018-12248: Heap Buffer Overflow
- Stack Overflow

PHP:

Division by Zero

SEGV

Stack Overflow

PHP:

- Division by Zero
- SEGV
- Stack Overflow

lua:

- UAF

PHP:

- Division by Zero
- SEGV
- Stack Overflow

lua:

- UAF

ChakraCore:

- OOM Crash

Conclusion

Conclusion

- Grammars & Feedback ++

Conclusion

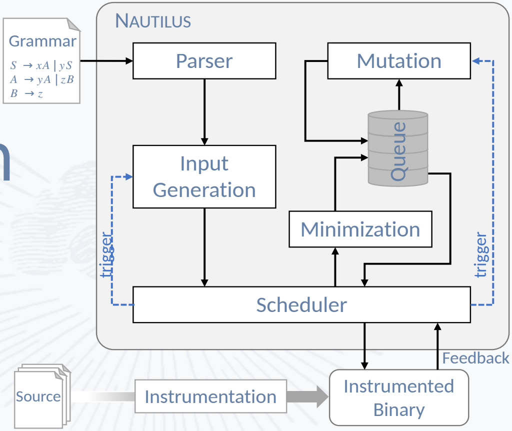
- Grammars & Feedback ++
- Splicing is important!

Overview

Generation

Minimization

Mutations


Grammar
$S \rightarrow xA \mid yS$
$A \rightarrow yA \mid zB$
$B \rightarrow z$
NAUTILUS
Parser
Mutation
Queue
Input Generation
Minimization
trigger
trigger
Scheduler
Feedback
Source
Instrumentation
Instrumented Binary