Fuzzing with Mutations

Security Testing

Andreas Zeller, Saarland University

Fuzz Input

[;x1-GPZ+wcckc];,N9J+?#6^6\e?]9lu2_%'4GX"0VUB[E/r
~fApu6b8<{%siq8Zh.6{V,hr?;{Ti.r3PIxMMMv6{xS^+'Hq!
AxB"YXRS@!Kd6;wtAMefFWM(`lJ_<1~o}z3K(CCzRH JIlvHz>_*.
\>JrlU32~eGP?IR=bF3+;y$3lodQ<B89!5"W2fK*vE7v{')KC-
i,c{<[~m!]o;{.'}Gj\(X}EtYetrpbY@aGZ1{P!AZU7x#4(Rtn!
q4nCwqol^y6}0lKo=*JK~;zMKV=9Nai:wxu{J&UV#HaU)*BiC<),`
+t*gka<W=Z.%T5WGHZpI30D<Pq>&]BS6R&j?#tP7iaV}-}`\?
[_[Z^LBMPG-FKj'\xwuZ1=Q`^`5,NQ@[!CuRzJ2DlvBy!
^zkhdf3C5PAkR?V hnl3='i2Qx]D
$qs4O`1@fevnG'2\11Vf3piU37@55ap\zIyl"'f,
$ee,J4Gw:cgNKLie3nx9(`efSlg6#[K"@WjhZ}r[Scun&sBCS,T[/
vY'pduwgzDlVNy7'rnzxNwI)(ynBa>%lb`;`9fG]P_0hdG~$@6
3]KAeEnQ7lU)3Pn,0)G/6N-wyzj/MTd#A;r

Fuzz Input

[;x1-GPZ+wcckc];,N9J+?#6^6\e?]9lu2_%'4GX"0VUB[E/r
~fApu6b8<{%siq8Zh.6{V,hr?;{Ti.r3PIxMMMv6{xS^+'Hq!
AxB"YXRS@!Kd6;wtAMefFWM(`lJ_<1~o}z3K(CCzRH JIlvHz>_*.
\>JrlU32~eGP?lB–bF3+;v$3lodQ<B89I5"W2fK*vE7v{')KC-
i,c{<[~m!]o;{.'}C n!
q4nCwqoI^y6}( )*BiC<),`
+t*gka<W=Z.% V}-}`\?
[_[Z^LBMPG-F By!
^zkhdf3C5PAk
$qs4O`1@fevnG 2\TTVI3plO37@55apvZIyF`1,
$ee,J4Gw:cgNKLie3nx9(`efSlg6#[K"@WjhZ}r[Scun&sBCS,T[/
vY'pduwgzDlVNy7'rnzxNwI)(ynBa>%lb`;`9fG]P_0hdG~$@6
3]KAeEnQ7lU)3Pn,0)G/6N-wyzj/MTd#A;r

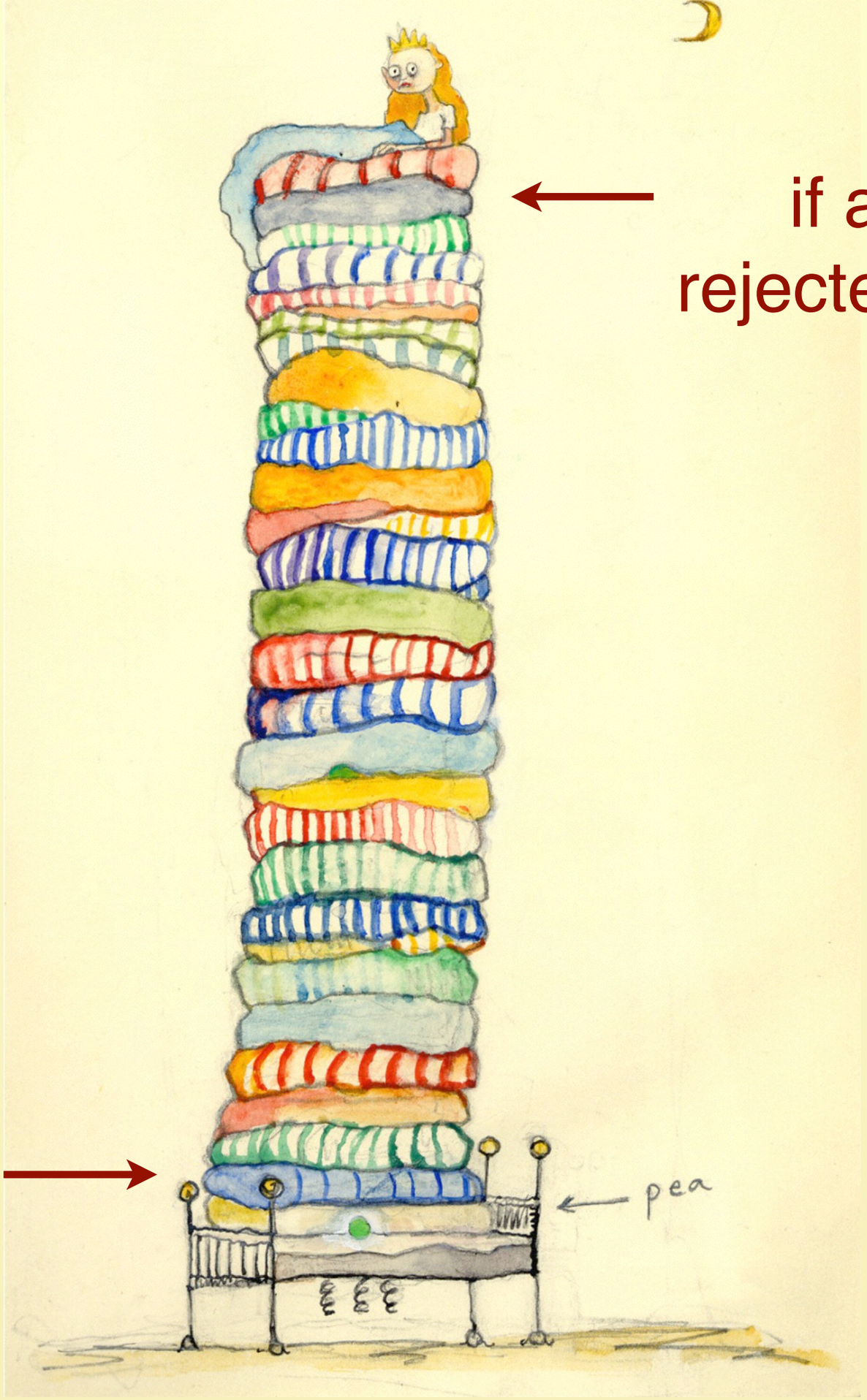
Most programs *reject* invalid inputs

(especially after having been fuzzed)


pea

if all input is rejected here…

…we will never get to the pea



The Challenge

How do we get a fuzzer to produce *valid* inputs?

(and find the pea)

The Idea

Leverage
existing **valid inputs!**

american fuzzy lop

american fuzzy lop 1.86b (test)

process timing
run time : 0 days, 0 hrs, 0 min, 2 sec
last new path : none seen yet
last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
last uniq hang : none seen yet

overall results
cycles done : 0
total paths : 1
uniq crashes : 1
uniq hangs : 0

cycle progress
now processing : 0 (0.00%)
paths timed out : 0 (0.00%)

map coverage
map density : 2 (0.00%)
count coverage : 1.00 bits/tuple

stage progress
now trying : havoc
stage execs : 1464/5000 (29.28%)
total execs : 1697
exec speed : 626.5/sec

findings in depth
favored paths : 1 (100.00%)
new edges on : 1 (100.00%)
total crashes : 39 (1 unique)
total hangs : 0 (0 unique)

fuzzing strategy yields
bit flips : 0/16, 1/15, 0/13
byte flips : 0/2, 0/1, 0/0
arithmetics : 0/112, 0/25, 0/0
known ints : 0/10, 0/28, 0/0
dictionary : 0/0, 0/0, 0/0
havoc : 0/0, 0/0
trim : n/a, 0.00%

path geometry
levels : 1
pending : 1
pend fav : 1
own finds : 0
imported : n/a
variable : 0

[cpu: 92%]

fuzzing strategy

bit flips : 0/16

byte flips : 0/2,

arithmetics : 0/11

known ints : 0/10

dictionary : 0/0,

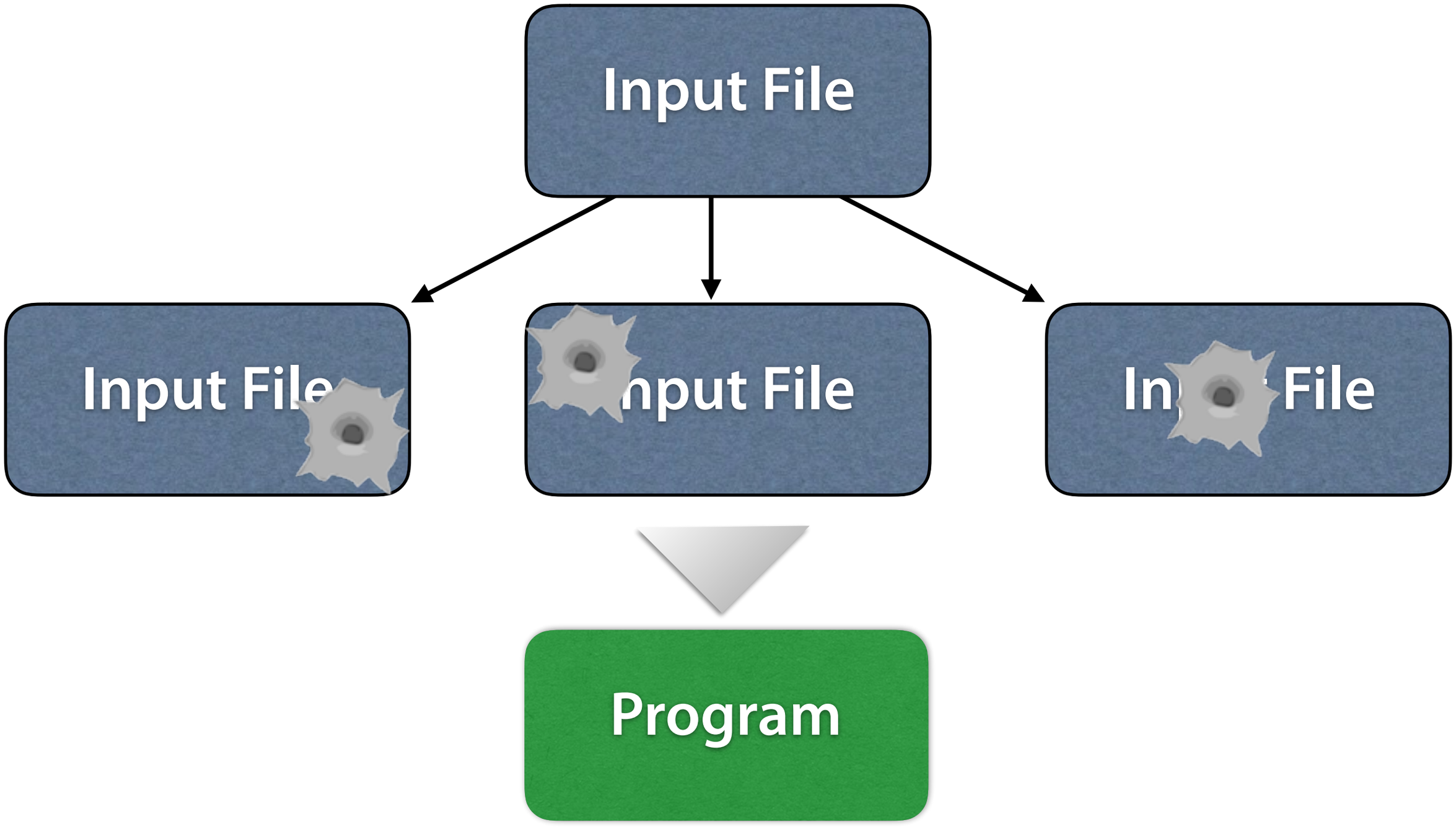
havoc : 0/0,

trim : n/a,

American Fuzzy Lop

Input File

American Fuzzy Lop



Mutations

- **Bit Flip**: Change a random bit in the input
- **Byte Flip**: Swap two bytes in the input
- **Arithmetic**: Increase/decrease number
- **Trim**: Remove input tail
- … and more …

American Fuzzy Lop

Wikipedia

Demo

Trimming

```
# Trim the last characters of an input
def trim(s):
    max_len = random.randint(0, len(s) - 1)
    sys.stderr.write("Trimming at " + repr(max_len) + "\n")
    return s[:max_len]
```

Flipping

```
# Flip a random bit at a random position
def flip(s):
    pos = random.randint(0, len(s) - 1)
    bit = random.randint(0, 6)
    sys.stderr.write("Flipping bit " + repr(bit) + " at " +
repr(pos) + "\n")
    flipped = chr(ord(s[pos]) ^ (1 << bit))
    return s[:pos] + flipped + s[pos + 1:]
```

Swapping

```
# Swap two bytes at a random position
def swap(s):
    pos = random.randint(0, len(s) - 2)
    sys.stderr.write("Swapping at " + repr(pos) + "\n")
    return s[:pos] + s[pos + 1] + s[pos] + s[pos + 2:]
```

Mutating

```
def process(f):
    # Read specified file
    s = f.read()

    # Choose a mutator (randomly)...
    mutations = [trim, flip, swap]
    mutator = mutations[int(random.random() * len(mutations))]

    # ...and output its result.
    mutated = mutator(s)
    sys.stdout.write(mutated)
```

Main

```
# Main function
def main():
    opts, args = getopt.getopt(sys.argv[1:], "h")
    for opt, a in opts:
        if opt == "-h":
            help()
        else:
            assert False, "unhandled option"

if len(args) == 0:
    # No arguments: read standard input
    process(sys.stdin)
    sys.exit(0)

# Process one argument after another
for arg in args:
    # Read specified file
    process(open(arg))
```

Things To Do

- Make configurable (more options)
- Integrate subprocess communication (invoking program again and again)
- Add further mutators (e.g. arithmetic)
- Try out on various binary formats

mozilla

SECURITY TESTING AT MOZILLA

CHRISTIAN HOLLER, M.SC.
STAFF SECURITY ENGINEER