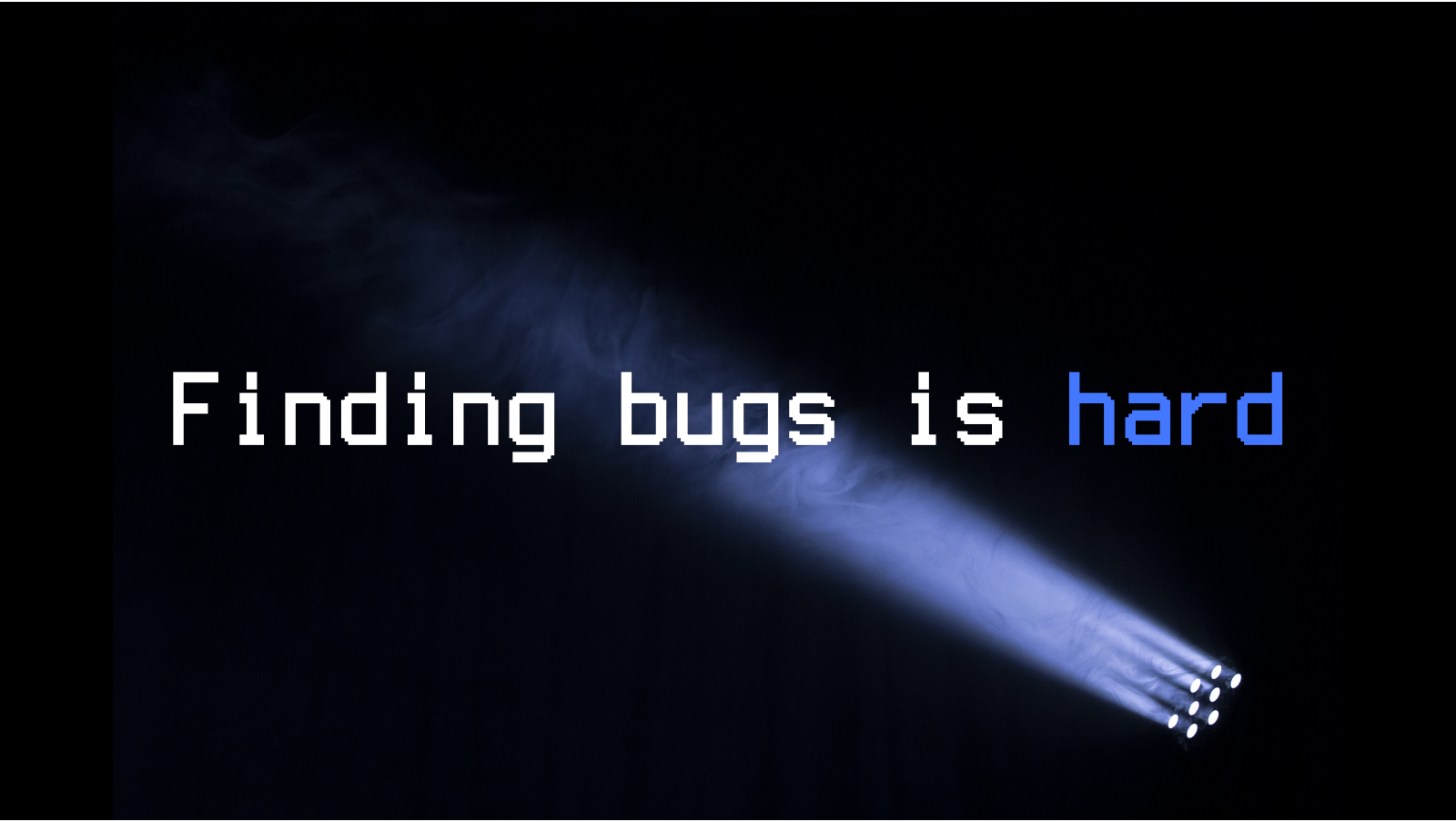
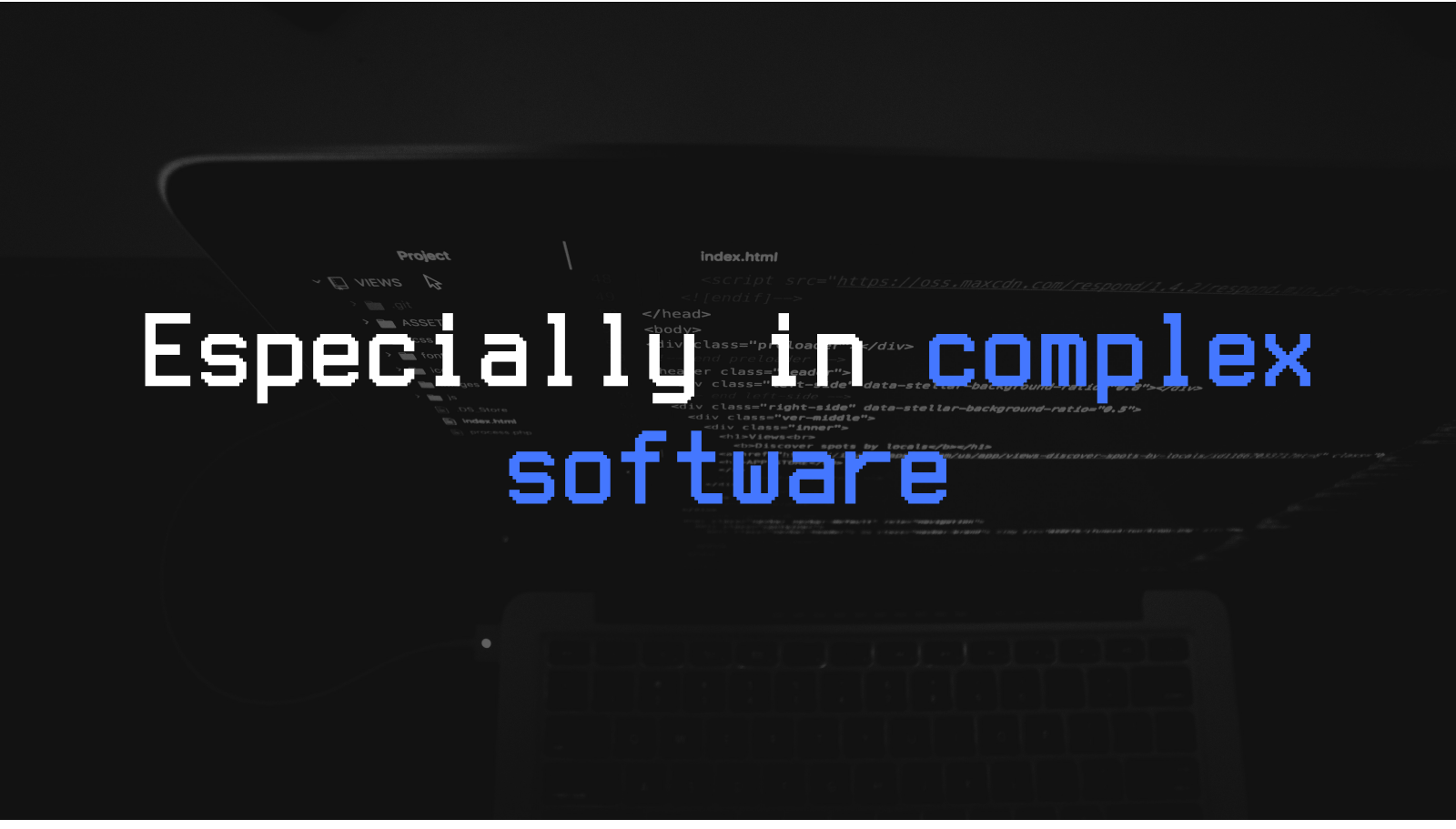
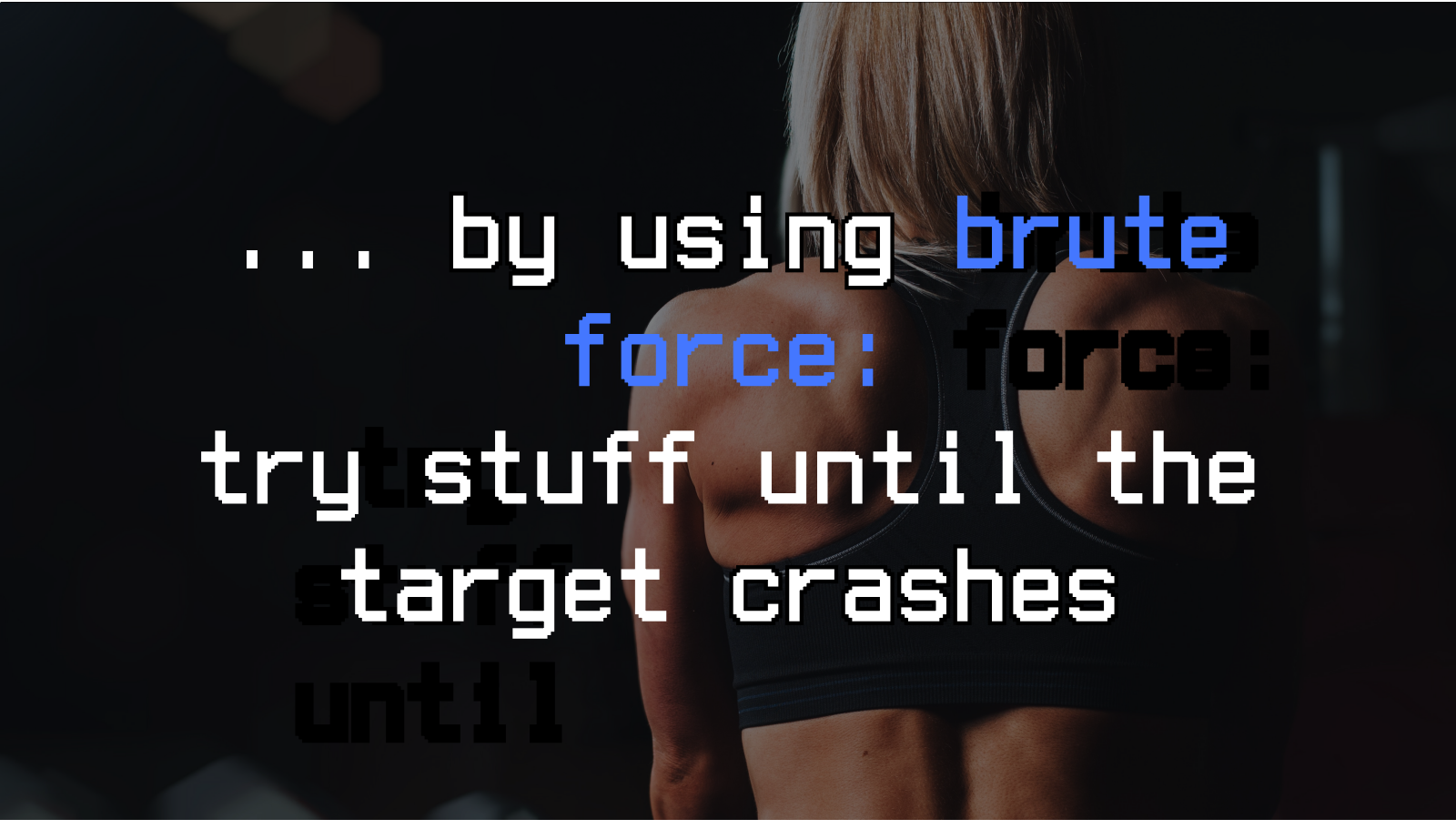
Fuzzing101

https://dumpco.re/fuzz


Finding bugs is hard


Especially in complex
software


Fuzzing: automated
crashcrash
discovery


... by using brute force: try stuff until the target crashes

1) Send input
2) Did it crash?
3) repeat

Example target:

```
$ ./url-verifier https://bsideskbh.dk/schedule/
URL valid!
```

Example target:

```
$ ./url-verifier AAAAAAA
Error: that is not a URL
```

What should we use as input?

Dumb:
Generate random input

```
#!/bin/bash


            cat /dev/urandom
```

```
#!/bin/bash


cat /dev/urandom | head -c 100
```

```
#!/bin/bash



    # generate random input
    input=$(cat /dev/urandom | head -c 100)
```

```
#!/bin/bash



    # generate random input
    input=$(cat /dev/urandom | head -c 100)

    # send it to target
    ./url-verifier "$input"
```

```
#!/bin/bash



    # generate random input
    input=$(cat /dev/urandom | head -c 100)

    # send it to target
    ./url-verifier "$input"

    # did it crash?
    if [ $? -eq 139 ]; then
        # save testcase
        echo "$input" > crashes/crash

    fi
```

```
#!/bin/bash

counter=0
while [ 1 ]; do
    # generate random input
    input=$(cat /dev/urandom | head -c 100)

    # send it to target
    ./url-verifier "$input"

    # did it crash?
    if [ $? -eq 139 ]; then
        # save testcase
        echo "$input" > crashes/crash$counter
        let counter+=1
    fi
```

Mutation based:
Mutate existing good input

radamsa example001

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://user:pass@host.com:-8078/p/a/t
```

radamsa example002

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://u sfsser:pass@host.com:32766683506592233 7  03685477
```

radamsa example03

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://user:pa-ss@host.ꝋcom:8080/p/a/t
```

radamsa example004

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://user:pbss@host.com:-14643325729961986883/p/a/\x0d\x
```

radamsa example005

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://user:pass@host.co  m:8080/p/a/t
```

radamsa example006

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
http://user:pass@host.    com:$+%d"xcalc\x0d%p!xcalc"xcalc
```

radamsa example007

```
$ echo 'http://user:pass@host.com:8080/p/a/t' | ./radamsa
htp://user:pass@host.com:8080/p/a/t
```

```
#!/bin/bash

counter=0
while [ 1 ]; do
    # mutate sample input
    input=$(echo 'http://user:pass@host.com:8080/p/a/t' |

    # send it to target
    ./url-verifier "$input"

    # did it crash?
    if [ $? -eq 139 ]; then
        # save testcase
        echo "$input" > crashes/crash$counter
        let counter+=1
    fi
```

Generation based:
Generate input from
a data model

dharma example001

```
$ ./dharma.py -grammars url.dg
http://Y.f-.u9%B8.c%5E'*-.G:4739/++/%D1/T+-+-%272+?4'+-
```

dharma example002

```
$ ./dharma.py -grammars url.dg
http://a.F,'f.b/3?.+%3B5_"+%D9%0C,
```

dharma example003

```
$ ./dharma.py -grammars url.dg
prospero://m%14%76,S.U*53.o.H:7/'+%40@/+/++%008
```

dharma example004

```
$ ./dharma.py -grammars url.dg
ftp://_:-5@2.6.738.4:1/Q+w/k++
```

dharma example005

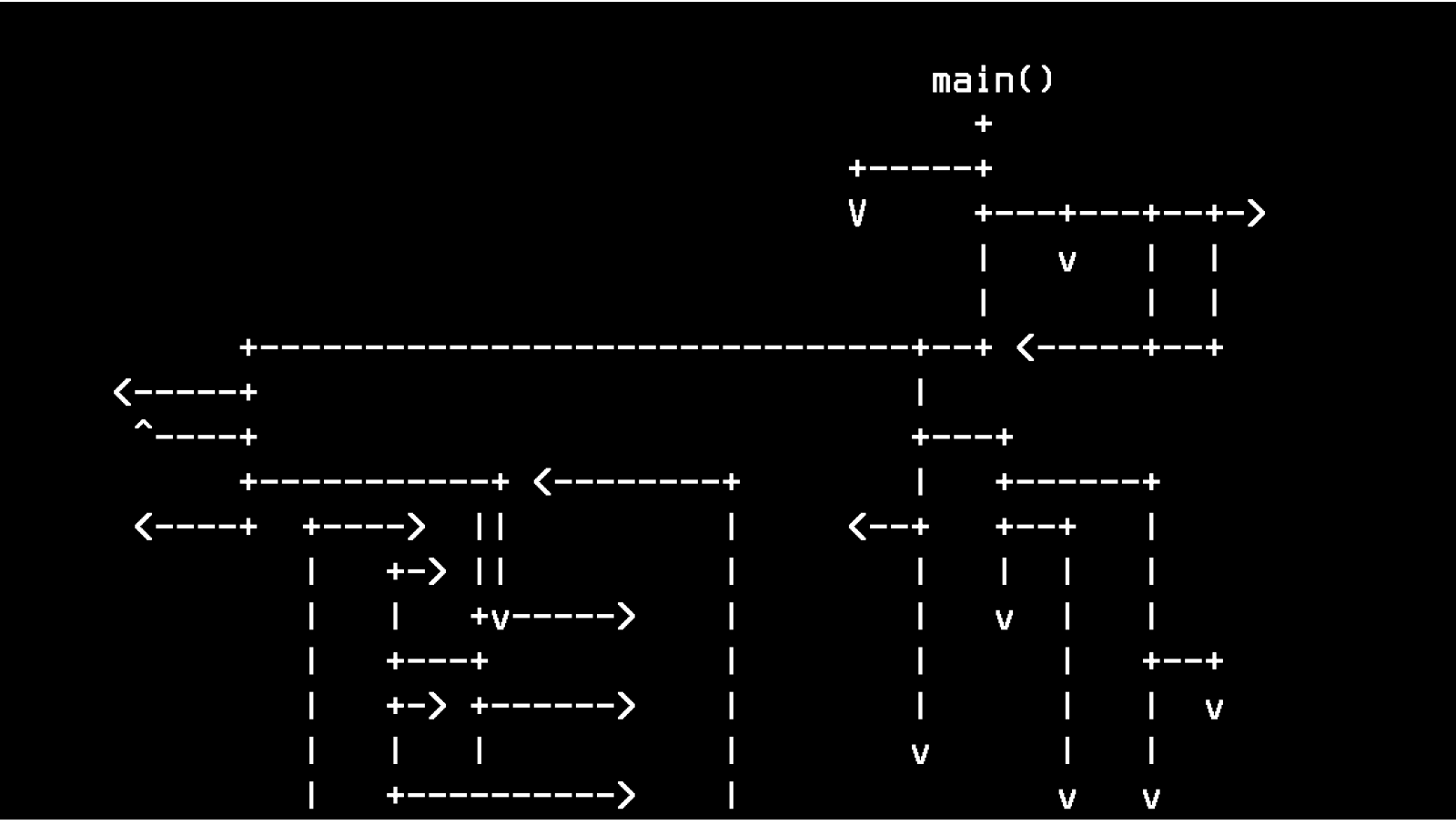
```
$ ./dharma.py -grammars url.dg
gopher://53.7.5915141.14:8/'+%E4/-+P
```

dharma example006

```
$ ./dharma.py -grammars url.dg
gopher://n.qx/0f/(&
```

```
$ cat url.dg
url :=
    +httpaddress+
    +ftpaddress+
    +telnetaddress+

httpaddress :=
    http://+hostport+
    http://+hostport+/+path+
    http://+hostport+/+path+?+search+

ftpaddress :=
    ftp://+login+/+path+
    ftp://+login+/+path+;+ftptype+

telnetaddress :=
```

```
#!/bin/bash

counter=0
while [ 1 ]; do
    # generate input with Dharma
    input=$(./dharma.py -grammars url.dg)

    # send it to target
    ./url-verifier "$input"

    # did it crash?
    if [ $? -eq 139 ]; then
        # save testcase
        echo "$input" > crashes/crash$counter
        let counter+=1
    fi
```


main()

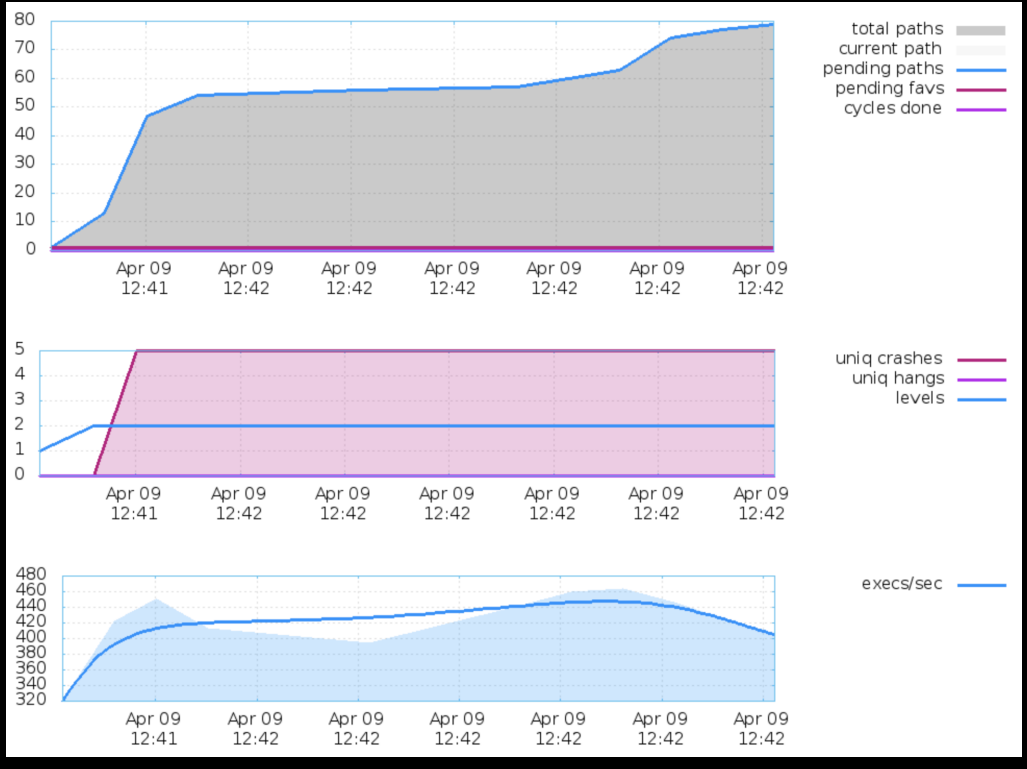
Feedback-driven fuzzing (code coverage guided)

1) Send input
2) Did it crash?
3)
4) repeat

1) Send input
2) Did it crash?
3) Did it reach new code?
4) repeat

1) Compile target with instrumentation
2) Fuzz

demo


80
70
60
50
40
30
20
10
0
Apr 09
12:41
Apr 09
12:42
total paths
current path
pending paths
pending favs
cycles done
5
4
3
2
1
0
uniq crashes
uniq hangs
levels
480
460
440
420
400
380
360
340
320
execs/sec

```
$ tree crashes/
crashes/
    crash0
    crash1
    crash2
    crash3
    crash4
    crash5

0 directories, 6 files

$ cat crashes/crash0
http://xx.aY.f-.u9%B8.c%5E-.G:4739/p

$ cat crashes/crash1
http://a29t43t432./fwaofew//AAAAAAAAAAAA
```

crash == bug

crash == vulnerability
?

crash analysis001

```
$ gdb -q url-verifier
Reading symbols from url-verifier (no debugging symbols fo
(gdb) r 'http://xx.aY.f-.u9%B8.c%5E-.G:4739/p'
Starting program: url-verifier 'http://xx.aY.f-.u9%B8.c%5E

Program received signal SIGSEGV, Segmentation fault.
0x004004c6 in ?? ()
(gdb) p $_siginfo._sifields._sigfault.si_addr
$1 = (void *) 0x0
(gdb)
```

crash analysis002

```
$ gdb -q url-verifier
Reading symbols from url-verifier (no debugging symbols fo
(gdb) r 'http://a29t43t432./fwaofew//AAAAAAAAAAAA'
Starting program: url-verifier 'http://a29t43t432./fwaofew

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb) p/x $eip
$1 = 0x41414141
(gdb)
```

Fuzzing over networks

Typical server code

```
init();

while (keep_running) {

    waitForData(); // blocking

    readInput();
    parseInput();

    housekeeping();

}

cleanup();
```

Server fuzzing concept

```
init();

while (keep_running) {

    input = readFromFile();
    sendToItself(input);

    waitForData(); // blocking

    readInput();
    parseInput();

    housekeeping();

}
```

Findings:

- ntpd: 2 bugs
- openslp: 3 bugs
- net-snmp: 2 bugs
- ntpsec: 4 bugs

details on https://dumpco.re

Is fuzzing worth it?

https://github.com/google/honggfuzz
https://llvm.org/docs/LibFuzzer.html#trophies
http://lcamtuf.coredump.cx/afl/

Fuzzers find crashes...

...convert whatever you want to find into a crash

```
abort()
```

Logic flaws

```
// read input from fuzzer
input = readInput();

// give it to both libraries
resA = libA(input);
resB = libB(input);

// if they don't agree on the result, it's a logical flaw
if (resA != resB) {
    abort();
}
```

Authentication bypass

```
if (authenticated == true) {
    abort();
}
```

sandbox escape?
side channels?
degradation of
service?
mess up audit trail?

If you can make it crash upon a special condition, then it can be fuzzed

@magnusstubman