Binary Fuzzing

Jon Crussell

UC Davis

February 21th, 2014

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.

Two main techniques:

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.
Two main techniques:

- Mutation based: randomly mutate known (valid) inputs

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.
Two main techniques:

- Mutation based: randomly mutate known (valid) inputs
- Generation based: create input from scratch based on specification

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.
Two main techniques:

- Mutation based: randomly mutate known (valid) inputs
- Generation based: create input from scratch based on specification

Smart vs. Dumb

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.
Two main techniques:

- Mutation based: randomly mutate known (valid) inputs
- Generation based: create input from scratch based on specification

Smart vs. Dumb

- Dumb: Knows nothing about input format

What is fuzzing?

(Semi-)Automated technique that involves sending unexpected input to programs.

Two main techniques:

- Mutation based: randomly mutate known (valid) inputs
- Generation based: create input from scratch based on specification

Smart vs. Dumb

- Dumb: Knows nothing about input format
- Smart: Knows more than nothing about input format

Examples

Examples

Dumb mutative:

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

- Take known inputs, randomly flip bits

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

- Take known inputs, randomly flip bits
- Fix checksums

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

- Take known inputs, randomly flip bits
- Fix checksums

Generative:

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

- Take known inputs, randomly flip bits
- Fix checksums

Generative:

- All generative fuzzers should be “smart”

Examples

Dumb mutative:

- Take known inputs, randomly flip bits

Smart mutative:

- Take known inputs, randomly flip bits
- Fix checksums

Generative:

- All generative fuzzers should be “smart”
- Take HTTP specification, generate HTTP requests

Why fuzzing?

Why fuzzing?

- Find security vulnerabilities

Why fuzzing?

- Find security vulnerabilities
- Find memory leaks

Why fuzzing?

- Find security vulnerabilities
- Find memory leaks
- Find uncaught exceptions

Why fuzzing?

- Find security vulnerabilities
- Find memory leaks
- Find uncaught exceptions
- Find differences between program versions

Why fuzzing?

- Find security vulnerabilities
- Find memory leaks
- Find uncaught exceptions
- Find differences between program versions
- Fun, Profit?

Fuzzing Limitations

Fuzzing Limitations

- Code coverage

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
  - How many files of length N bits?

Fuzzing Limitations

- Code coverage
    - Every program can be expressed as a control flow graph
    - Effective fuzzers find less commonly used code paths
    - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
    - How many files of length N bits? 2^N

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
  - How many files of length N bits? 2^N
  - 10 bit input

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
  - How many files of length N bits? 2^N
  - 10 bit input $\rightarrow 2^{10} = 1024$ files

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
  - How many files of length N bits? 2^N
  - 10 bit input → $2^{10} = 1024$ files
  - 1024 bit input

Fuzzing Limitations

- Code coverage
  - Every program can be expressed as a control flow graph
  - Effective fuzzers find less commonly used code paths
  - Hard for mutative fuzzers to find paths not already in known inputs
- Complexity
  - How many files of length N bits? 2^N
  - 10 bit input → $2^{10} = 1024$ files
  - 1024 bit input → $2^{1024} =$ 17976931348623159077293051907890247336179769789423065727 34300811577326758055009631327084773224075360211201138798 71393357658789768814416622492847430639474124377767893424 86548527630221960124609411945308295208500576883815068234 24628814739131105408272371633505106845862982399472459384 797163048353563296242241 37216 files

Demo

Dumb mutative fuzzer

Demo

Dumb mutative fuzzer

- Runs for *i* iterations

Demo

Dumb mutative fuzzer

- Runs for *i* iterations
- For each iteration, generates one input file, randomly replacing bytes with random values

Demo

Dumb mutative fuzzer

- Runs for *i* iterations
- For each iteration, generates one input file, randomly replacing bytes with random values
- Every *j* iterations, appends new random bytes

Demo

Dumb mutative fuzzer

- Runs for *i* iterations
- For each iteration, generates one input file, randomly replacing bytes with random values
- Every *j* iterations, appends new random bytes
- Target: feh a “mode-based image viewer.”