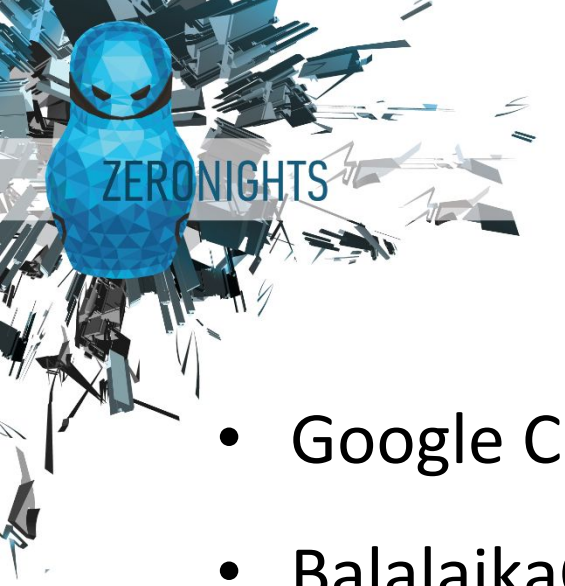
Modern Fuzzing of C/C++ projects

Max Moroz
Google



Bio

- Google Chrome Security team, Bugs--
- BalalaikaCr3w, LC↯BC
- CTF, BugBounty, etc

Agenda

1. TODO: write agenda
2. ???
3. Slides
4. Workshop
5.

My first year in university

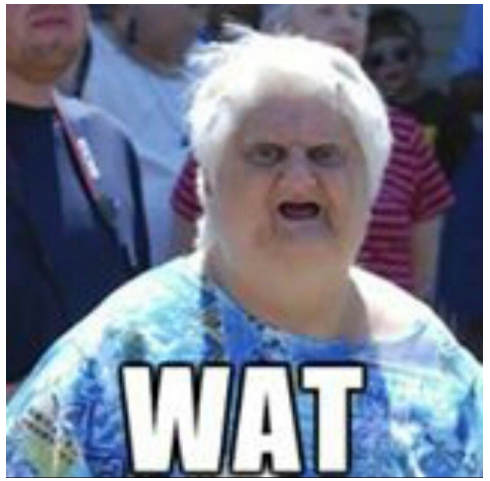
```
$ ./fact
Enter n to compute n! : 5
5! = 120
```

My first year in university

```
$ ./fact
Enter n to compute n! : 5
5! = 120

$ ./fact
Enter n to compute n! :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
```

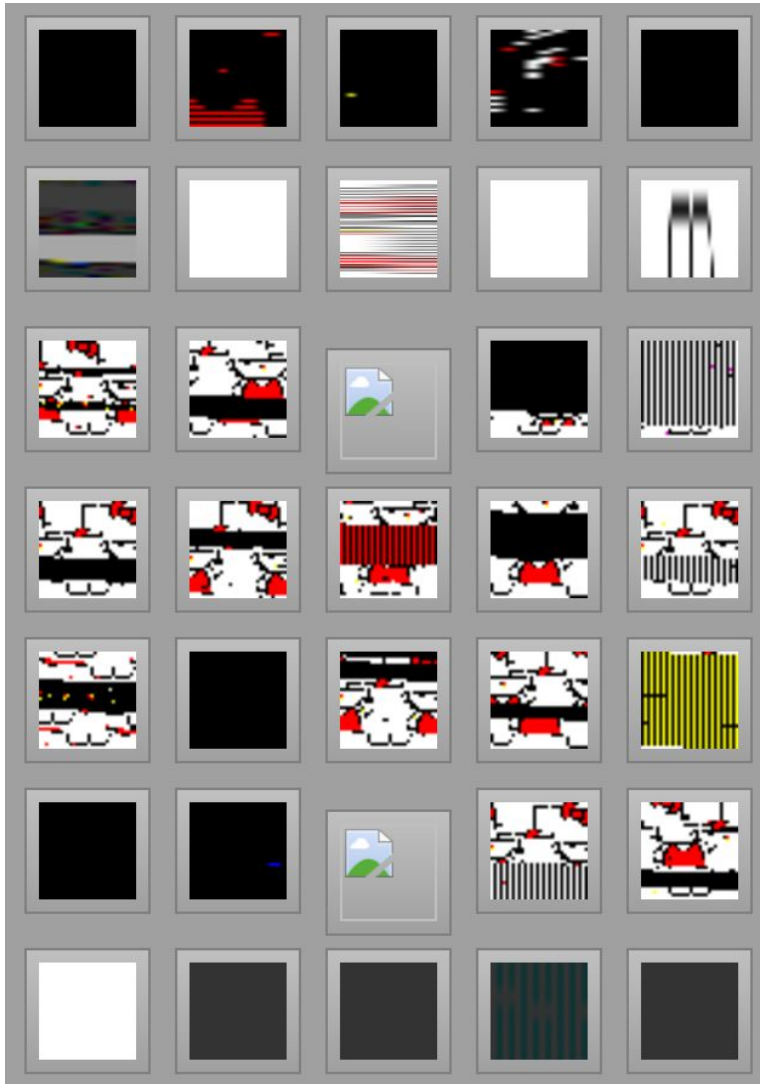
My first year in university

```
$ ./fact
Enter n to compute n! : 5
5! = 120

$ ./fact
Enter n to compute n! :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)

$ ./fact
Enter n to compute n! : 12345678990
-539222898! = 1
```



Fuzzing

Fuzzing



A software testing technique, often automated or semi-automated, that involves passing invalid, unexpected or random input to a program and monitor result for crashes, failed assertions, races, leaks, etc.

Unit testing vs. Fuzz testing

	Unit Testing	Old Fuzzing
Test small parts of code	✔	X
Can be automated	✔	✔
Regression testing	✔	✔ / X
Easy to write	✔	X
Looking for new bugs	✔ / X	✔✔✔
Looking for vulnerabilities	X	✔

Unit testing vs. Fuzz testing

	Unit Testing	Old Fuzzing	Modern Fuzzing
Test small parts of code	✔	X	✔
Can be automated	✔	✔	✔
Regression testing	✔	✔ / X	✔
Easy to write	✔	X	✔
Looking for new bugs	✔ / X	✔✔✔	✔✔✔✔✔✔
Looking for vulnerabilities	X	✔	✔

Vocabulary

- **Target**
  - Consumes an array of bytes
  - Calls the code we want to test
- **Fuzzer**
  - A tool that feed the target with different random inputs
- **Corpus**
  - A set of valid & invalid inputs for the target
  - Collected manually, by fuzzing, or by crawling

Fuzzer types

Overview

Fuzzer types

Generation Based

Generate from scratch with no prior state

Example

https://bugs.webkit.org/show_bug.cgi?id=60831

```
<script>
document.body = document.createElement('iframe');
</script>
```

Fuzzer types

Mutation Based

Mutate existing state based on some rules

Example

crbug.com/552046

```
--- orig.pdf
+++ crash.pdf
@@ -57,7 +57,7 @@
 /DecodeParms [null 8 0 R]
 /Type /XObject
 /Width 1760
-/Filter [/FlateDecode /DCTDecode]
+/Filter [/JBIG2Decode /DCTDecode]
 /Height 1248
 /Length 2277
```

Fuzzer types

Evolutionary

Generation or mutation based or both, in-process with code coverage feedback

Example

crbug.com/575205

```
SELECT'\xef(\xfb;DS\x1aLEETABL\xfeES'REGEXP
';0\t\tC LE|A*(\xc8*.+!*)*h*00\x0b$T''&'
```

Fuzzing in the past

Old school fuzzing

Routine

1. Generate an HTML page

Routine

1. Generate an HTML page
2. Write it to the disk

Routine

1. Generate an HTML page
2. Write it to the disk
3. Launch browser

Routine

1. Generate an HTML page
2. Write it to the disk
3. Launch browser
4. Open the page or serve it over HTTP

Routine

1. Generate an HTML page
2. Write it to the disk
3. Launch browser
4. Open the page or serve it over HTTP
5. Check if the browser crashed

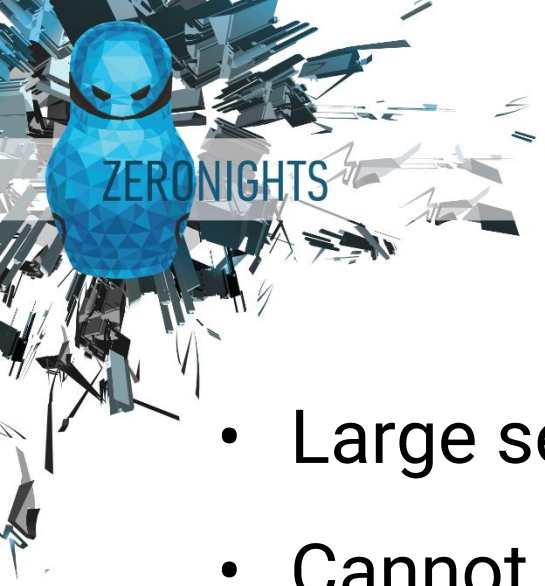
Routine

1. Generate an HTML page
2. Write it to the disk
3. Launch browser
4. Open the page or serve it over HTTP
5. Check if the browser crashed
6. Close the browser

Let’s write some code

Lesson 02

No coverage



Problems of old school fuzzing

- Large search space
- Cannot fuzz specific function
- Hard to fuzz network protocols
- Speed of regular fuzzers (html, css, dom, etc mutators)

Coverage

Goals

- More focused fuzzing
- Faster fuzzing
- Smarter fuzzing
- Easier fuzzer writing

LibFuzzer

New school fuzzing

LibFuzzer

- In-process, in-memory

LibFuzzer

- In-process, in-memory
- Guided fuzz testing

LibFuzzer

- In-process, in-memory
- Guided fuzz testing
- Very effective at a function / protocol level

LibFuzzer

- In-process, in-memory
- Guided fuzz testing
- Very effective at a function / protocol level
- 1000x faster

LibFuzzer

- In-process, in-memory
- Guided fuzz testing
- Very effective at a function / protocol level
- 1000x faster
- It’s easy to write a *libFuzzer*-based fuzzer

LibFuzzer

- In-process, in-memory
- Guided fuzz testing
- Very effective at a function / protocol level
- 1000x faster
- It’s easy to write a *libFuzzer*-based fuzzer
- Can be checked along with unit-tests

Coverage-guided fuzz testing

Memory Tools

How to see the invisible

Memory Tools

- **AddressSanitizer** (aka **ASan**)
  - Detects use-after-free, buffer overflows (heap, stack, globals), stack-use-after-return, container-overflow
  - Cpu: 2x, memory 1.5x-3x

- **MemorySanitizer** (aka **MSan**)
  - Detects uninitialized memory reads
  - Cpu: 3x, memory: 2x
  - Special mode: origins

- **UndefinedBehaviorSanitizer** (aka **UBSan**)
  - Detects several classes of bugs (19?), esp on type confusion, signed-integer-overflow, undefined shift, etc.
  - Cpu: 10-50%
  - Memory: ~1x (no allocator, no shadow)

Memory tools: example

- Container-overflow (**ASan**):

```
#include <vector>
#include <assert.h>
typedef long T;
int main() {
  std::vector<T> v;
  v.push_back(0);
  v.push_back(1);
  v.push_back(2);
  assert(v.capacity() >= 4);
  assert(v.size() == 3);
  T *p = &v[0];

  // Here the memory is accessed inside a heap-allocated buffer
  // but outside of the region `[v.begin(), v.end())`.
  return p[3]; // OOPS.
}
```

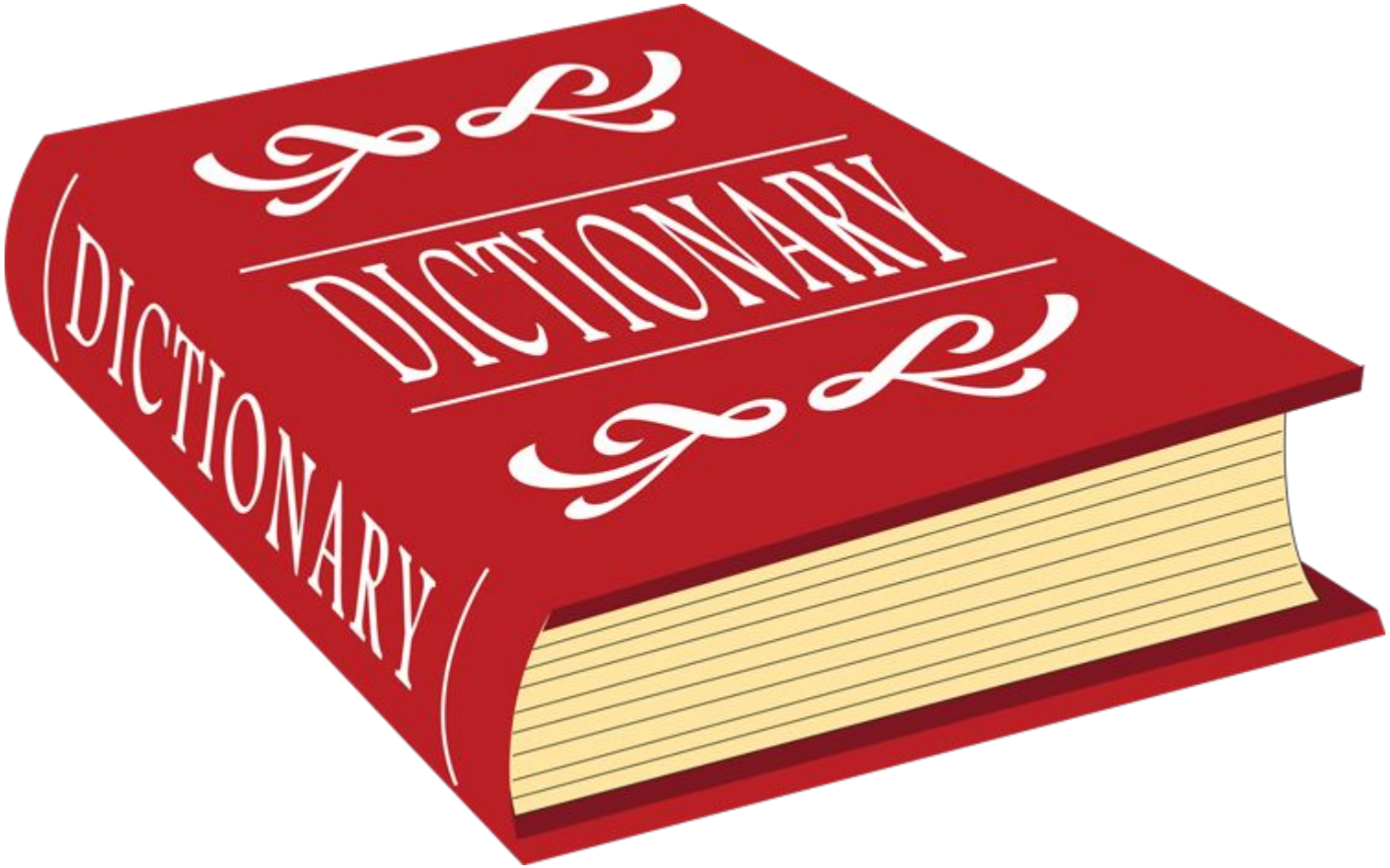
Let’s write some code

Lessons 03 - 06

How to improve your fuzzer

- Dictionaries
- Seed corpus
- Custom options
- Fuzzing of non-raw data arguments
- Optimization

Dictionaries



Dictionaries

XML:

```
" encoding=\"1\""
" a=\"1\""
" href=\"1\""
" standalone=\"no\""
" version=\"1\""
" xml:base=\"1\""
" xml:id=\"1\""
" xml:lang=\"1\""
" xml:space=\"1\""
" xmlns=\"1\""
"&lt;"
"&#1;"
"&a;"
"&#x1;"
"ANY"
"[]"
"CDATA"
":fallback"
":include"
"--"
"EMPTY"
"\"\""
"''"
"ENTITIES"
"ENTITY"
```

PNG:

```
header_png="\x89PNG\x0d\x0a\x1a\x0a"
section_IDAT="IDAT"
section_IEND="IEND"
section_IHDR="IHDR"
section_PLTE="PLTE"
section_bKGD="bKGD"
section_cHRM="cHRM"
section_fRAc="fRAc"
section_gAMA="gAMA"
section_gIFg="gIFg"
section_gIFt="gIFt"
section_gIFx="gIFx"
section_hIST="hIST"
section_iCCP="iCCP"
section_iTXt="iTXt"
section_oFFs="oFFs"
section_pCAL="pCAL"
section_pHYs="pHYs"
section_sBIT="sBIT"
section_sCAL="sCAL"
section_sPLT="sPLT"
section_sRGB="sRGB"
section_sTER="sTER"
section_tEXt="tEXt"
section_tIME="tIME"
```

Dictionary generation

Fuzzer	Dict. size	Corpus, %	Coverage, %	Speed, %
expat_xml_parse_fuzzer	428	-3.68	-2.72	1.59
libxml_xml_read_memory_fuzzer	780	30.81	41.14	24.13
net_ftp_directory_listing_fuzzer	842	1.84	1.03	-16.02
net_http_proxy_client_socket_fuzzer	2,006	**303.90**	26.36	-0.74
net_http_stream_parser_fuzzer	2,006	**259.34**	31.16	3.96
net_url_request_fuzzer	1,239	**127.44**	6.78	4.86
net_websocket_frame_parser_fuzzer	925	3.85	0.35	0.77
pdf_css_fuzzer	2,037	-7.56	-0.22	-1.79
pdf_xml_fuzzer	927	8.04	0.74	4.52
sqlite3_prepare_v2_fuzzer	657	**370.04**	**201.68**	-11.21
url_parse_fuzzer	650	**473.33**	**325.07**	-11.70
v8_script_parser_fuzzer	1,535	25.63	2.31	9.27

```
$ chromium/src/testing/libfuzzer/dictionary_generator.py \
      --fuzzer PATH_TO_FUZZER_BINARY \
      --spec PATH_TO_FORMAT_SPECIFICATION \
      --out GENERATED_DICTIONARY.dict
```

Recommended dictionary

```
Command: ['<...>/boringssl_read_pem_fuzzer', <...>]
Bot: clusterfuzz-linux-pre-0234
Time ran: 3251.046254

INFO: Seed: 1441206595
#0  READ   units: 1166 exec/s: 0
#1167   INITED cov: 204 bits: 366 indir: 16 units: 119 exec/s: 0
#1168   NEW    cov: 277 bits: 366 indir: 22 units: 120 exec/s: 0 L: 129 MS:
1 InsertByte-

<...>

#147891067 DONE   cov: 278 bits: 370 indir: 22 units: 123 exec/s: 45490
###### Recommended dictionary. ######
"(:(\xe7\x9d/!;!;O" # Uses: 8451959
"-----BEGIN " # Uses: 8454876
###### End of recommended dictionary. ######
Done 147891067 runs in 3251 second(s)
stat::number_of_executed_units: 147891067
stat::average_exec_per_sec:     45490
<...>
```

Recommended dictionary

```
Command: ['<...>/net_url_request_fuzzer, <...>]
Bot: clusterfuzz-linux-high-end-pre-0034
Time ran: 3251.478762

Dictionary: 128 entries
INFO: Seed: 964796221
#0  READ   units: 10055 exec/s: 0
#128    pulse  cov: 5322 bits: 7463 indir: 415 units: 10055 exec/s: 64


<...>

#366810 DONE   cov: 12167 bits: 55024 indir: 756 units: 10093 exec/s: 112
###### Recommended dictionary. ######
"blob" # Uses: 3996
"http" # Uses: 2420
"file" # Uses: 1536
"foo" # Uses: 1105
"ntlm" # Uses: 922
###### End of recommended dictionary. ######
<...>
```

Recommended dictionary

```
Command: ['<...>/net_parse_cookie_line_fuzzer', <...>]
Bot: clusterfuzz-linux-high-end-pre-0080
Time ran: 3251.114852

INFO: Seed: 1831825350
#0  READ   units: 1142 exec/s: 0
#1142   INITED cov: 318 bits: 1041 indir: 35 units: 332 exec/s: 0
#1143   NEW    cov: 427 bits: 1041 indir: 41 units: 333 exec/s: 0 L: 29 MS:
1 ChangeByte-
<...>
#6808457   NEW    cov: 428 bits: 1045 indir: 41 units: 337 exec/s: 11272
L: 698 MS: 5
ChangeByte-AddFromTempAutoDict-ShuffleBytes-AddFromTempAutoDict-CrossOver-
DE: "httponly"-"path"-
<...>
#36643517   DONE   cov: 428 bits: 1045 indir: 41 units: 337 exec/s: 11271
###### Recommended dictionary. ######
"httponly" # Uses: 1917525
"path" # Uses: 1942187
###### End of recommended dictionary. ######
<...>
```

Seed corpus

```
$ ls testing/libfuzzer/fuzzers/woff2_corpus/
AhemSpaceLigature.woff2    DejaVuSerif.woff2    MEgalopolisExtra.woff2
Ahem.woff2                 EzraSIL.woff2        mplus-1p-regular.woff2
DejaVuSerif-webfont.woff2  LinLibertineO.woff2  OpenSans.woff2
tcu-font.woff2
```

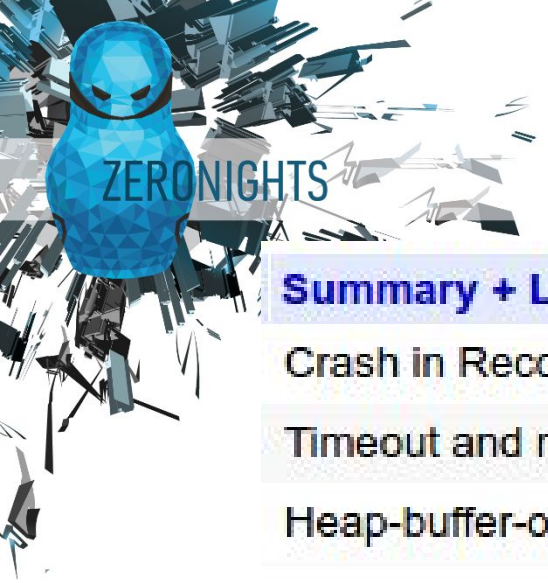
Seed corpus

```
$ ls testing/libfuzzer/fuzzers/woff2_corpus/
AhemSpaceLigature.woff2     DejaVuSerif.woff2     MEgalopolisExtra.woff2
Ahem.woff2                  EzraSIL.woff2         mplus-1p-regular.woff2
DejaVuSerif-webfont.woff2   LinLibertineO.woff2   OpenSans.woff2
tcu-font.woff2
```

```
fuzzer_test("convert_woff2ttf_fuzzer") {
 sources = [
  "convert_woff2ttf_fuzzer.cc",
 ]
 deps = [
  "//third_party/woff2:woff2_dec",
 ]
 seed_corpus = "//testing/libfuzzer/fuzzers/woff2_corpus"
 libfuzzer_options = [ "max_len=803500" ]
}
```

Custom options

```
fuzzer_test("convert_woff2ttf_fuzzer") {
 sources = [
  "convert_woff2ttf_fuzzer.cc",
 ]
 deps = [
  "//third_party/woff2:woff2_dec",
 ]
 seed_corpus = "//testing/libfuzzer/fuzzers/woff2_corpus"
 libfuzzer_options = [ "max_len=803500" ]
}
```



Custom options

Summary + Labels ▼
Crash in ReconstructTransformedHmtx Reproducible ClusterFuzz
Timeout and memory leaks in woff2::ConvertWOFF2ToTTF()
Heap-buffer-overflow in Read Reproducible Clusterfuzz

```
fuzzer_test("convert_woff2ttf_fuzzer") {
 sources = [
  "convert_woff2ttf_fuzzer.cc",
 ]
 deps = [
  "//third_party/woff2:woff2_dec",
 ]
 seed_corpus = "//testing/libfuzzer/fuzzers/woff2_corpus"
 libfuzzer_options = [ "max_len=803500" ]
}
```

Optimization [initial version]

```
#include "third_party/zlib/zlib.h"

// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const unsigned char *data, size_t size) {
    uint8_t buffer[1024 * 1024] = { 0 };
    size_t buffer_length = sizeof(buffer);

    if (Z_OK != uncompress(buffer, &buffer_length, data, size)) {
        return 0;
    }

    return 0;
}
```

Optimization [initial version]

```
#include "third_party/zlib/zlib.h"

// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const unsigned char *data, size_t size) {
    uint8_t buffer[1024 * 1024] = { 0 };
    size_t buffer_length = sizeof(buffer);

    if (Z_OK != uncompress(buffer, &buffer_length, data, size)) {
        return 0;
    }

    return 0;
}
```

Optimization [why you did that?]

```
#include "third_party/zlib/zlib.h"

// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const unsigned char *data, size_t size) {
    const int NUM_ITEMS = 1024 * 1024;
    const int BUF_SIZE = NUM_ITEMS * sizeof(uint8_t);
    uint8_t *buffer = new uint8_t[NUM_ITEMS];
    uLongf buffer_length = (uLongf)BUF_SIZE;
    memset(buffer, 0, BUF_SIZE);

    if (Z_OK != uncompress(buffer, &buffer_length, data, static_cast<uLong>(size))) {
        delete[] buffer;
        return 0;
    }

    delete[] buffer;
    return 0;
}
```

Optimization [reanimation]

```
#include "third_party/zlib/zlib.h"

// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const unsigned char *data, size_t size) {
    const int NUM_ITEMS = 1024 * 1024;
    const int BUF_SIZE = NUM_ITEMS * sizeof(uint8_t);
    uint8_t *buffer = new uint8_t[NUM_ITEMS];
    uLongf buffer_length = (uLongf)BUF_SIZE;
    memset(buffer, 0, BUF_SIZE);

    if (Z_OK != uncompress(buffer, &buffer_length, data, static_cast<uLong>(size))) {
        delete[] buffer;
        return 0;
    }

    delete[] buffer;
    return 0;
}
```

Optimization [reanimation]

```
#include "third_party/zlib/zlib.h"
static Bytef buffer[256 * 1024] = { 0 };
// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const unsigned char *data, size_t size) {
    uLongf buffer_length = static_cast<uLongf>(sizeof(buffer));
    const int BUF_SIZE = NUM_ITEMS * sizeof(uint8_t);
    uint8_t *buffer = new uint8_t[NUM_ITEMS];
    uLongf buffer_length = (uLongf)BUF_SIZE;
    memset(buffer, 0, BUF_SIZE);

    if (Z_OK != uncompress(buffer, &buffer_length, data, static_cast<uLong>(size))) {
        delete[] buffer;
        return 0;
    }

    delete[] buffer;
    return 0;
}
```

Optimization [final version]

```
#include "third_party/zlib/zlib.h"

static Bytef buffer[256 * 1024] = { 0 };

// Entry point for LibFuzzer.
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
    uLongf buffer_length = static_cast<uLongf>(sizeof(buffer));

    if (Z_OK != uncompress(buffer, &buffer_length, data, static_cast<uLong>(size))) {
        return 0;
    }

    return 0;
}
```

Guess performance difference?

Guess performance difference?

- remove memset() for 1MB heap buffer -> **~5x**

Guess performance difference?

- remove memset() for 1MB heap buffer -> **~5x**
- move 1MB heap buffer to the stack -> **~2x**

Guess performance difference?

- remove memset() for 1MB heap buffer -> **~5x**
- move 1MB heap buffer to the stack -> **~2x**
- use 256KB instead of 1Mb on the stack -> **~3x**

Guess performance difference?

- remove memset() for 1MB heap buffer -> **~5x**
- move 1MB heap buffer to the stack -> **~2x**
- use 256KB instead of 1Mb on the stack -> **~3x**
- move 256KB buffer from stack to global -> **~2x**

Guess performance difference?

- remove memset() for 1MB heap buffer -> **~5x**
- move 1MB heap buffer to the stack -> **~2x**
- use 256KB instead of 1Mb on the stack -> **~3x**
- move 256KB buffer from stack to global -> **~2x**

53x

Chromium integration

Homework assignment

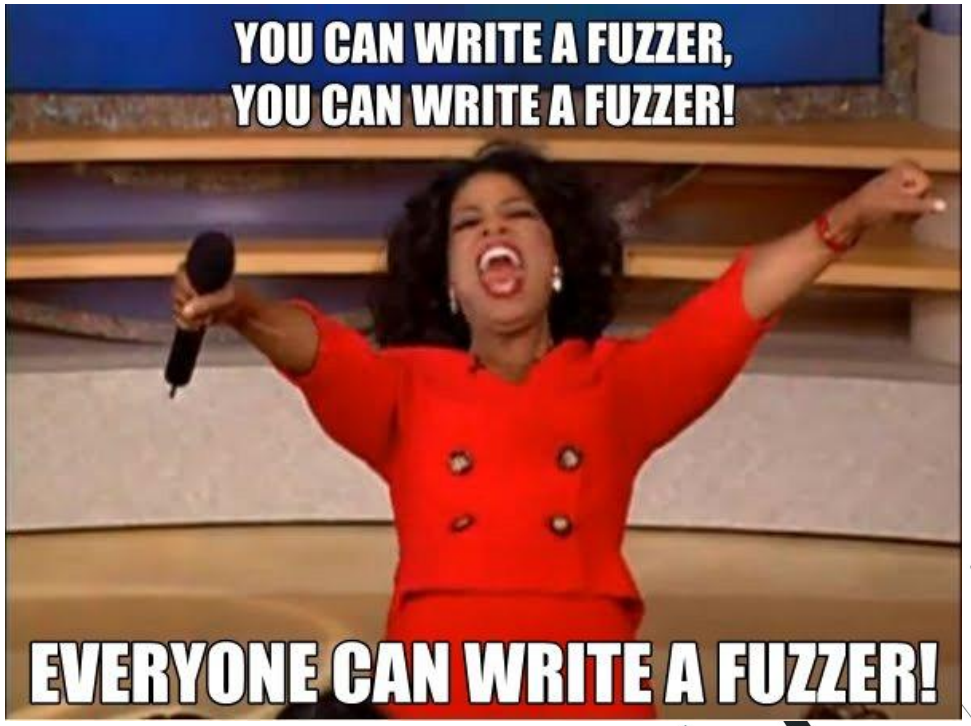
Fuzz target (== target function)

```
#include "libxml/parser.h"

extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {

  auto doc = xmlReadMemory(data, size, "noname.xml", NULL, 0);
  if (doc) {
    xmlFreeDoc(doc);
  }

  return 0;
}
```


https://chromium.googlesource.com/chromium/src/+/master/testing/libfuzzer/README.md

Build configuration

```
fuzzer_test("libxml_xml_read_memory_fuzzer") {
  sources = [
    "libxml_xml_read_memory_fuzzer.cc",
  ]
  deps = [
    "//third_party/libxml:libxml",
  ]
}
```



https://chromium.googlesource.com/chromium/src/+/master/testing/libfuzzer/README.md

Chrome Fuzzer Program

- The Chrome Fuzzer Program allows you to run fuzzers on Google hardware at Google scale across thousands of cores. You receive 100% of the reward value for any bugs found by your fuzzer plus a bonus $500, provided the same bug was not found by one of our fuzzers within 48 hours. There are two ways to participate:
  - libFuzzer
  - ClusterFuzz

https://www.google.com/about/appsecurity/chrome-rewards/index.html#fuzzerprogram

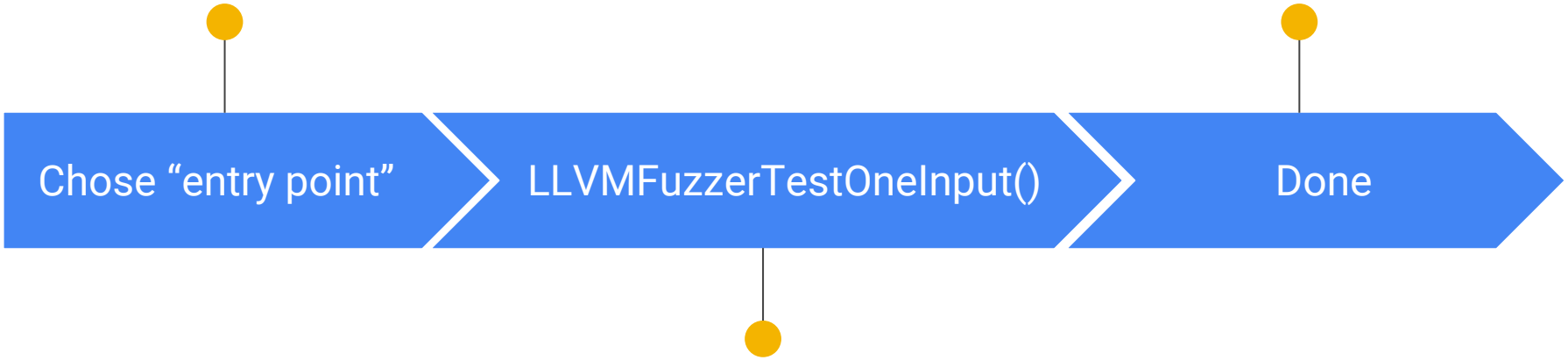
Chrome Fuzzer Program

https://www.google.com/about/appsecurity/chrome-rewards/index.html#fuzzerprogram

How to start [1/2]

Find a function with raw data input controlled by external source (user, server, etc)

First version of the fuzzer is ready, let's fuzz!

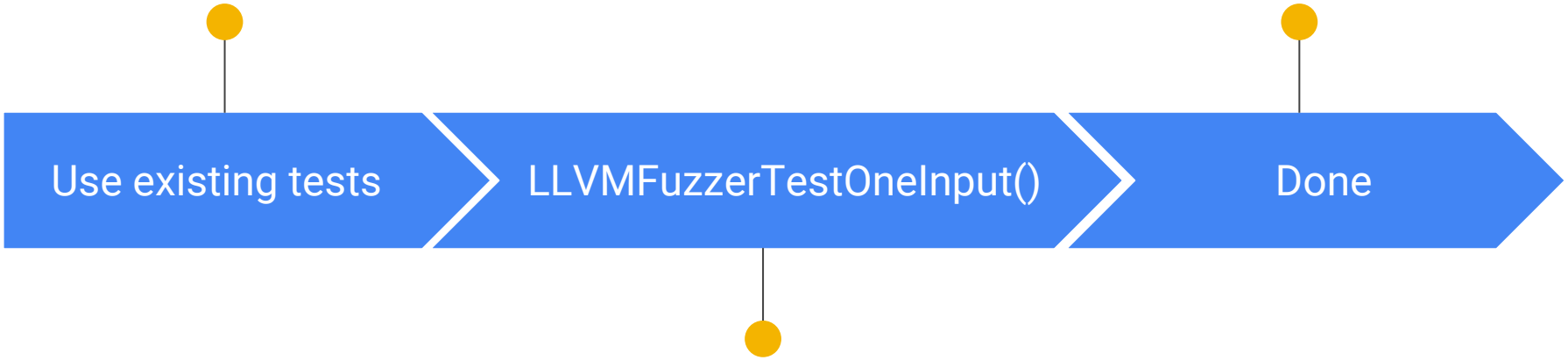


Write a target function which feeds fuzzer's data into function chosen to fuzz

How to start [2/2]

Find a function with raw data input controlled by external source (user, server, etc)

First version of the fuzzer is ready, let's fuzz!



Replace testing input with data provided by LibFuzzer and wrap it into target function

Q & A

Thank you!

mmoroz@chromium.org
Twitter: @dor3s
Telegram: @dor1s