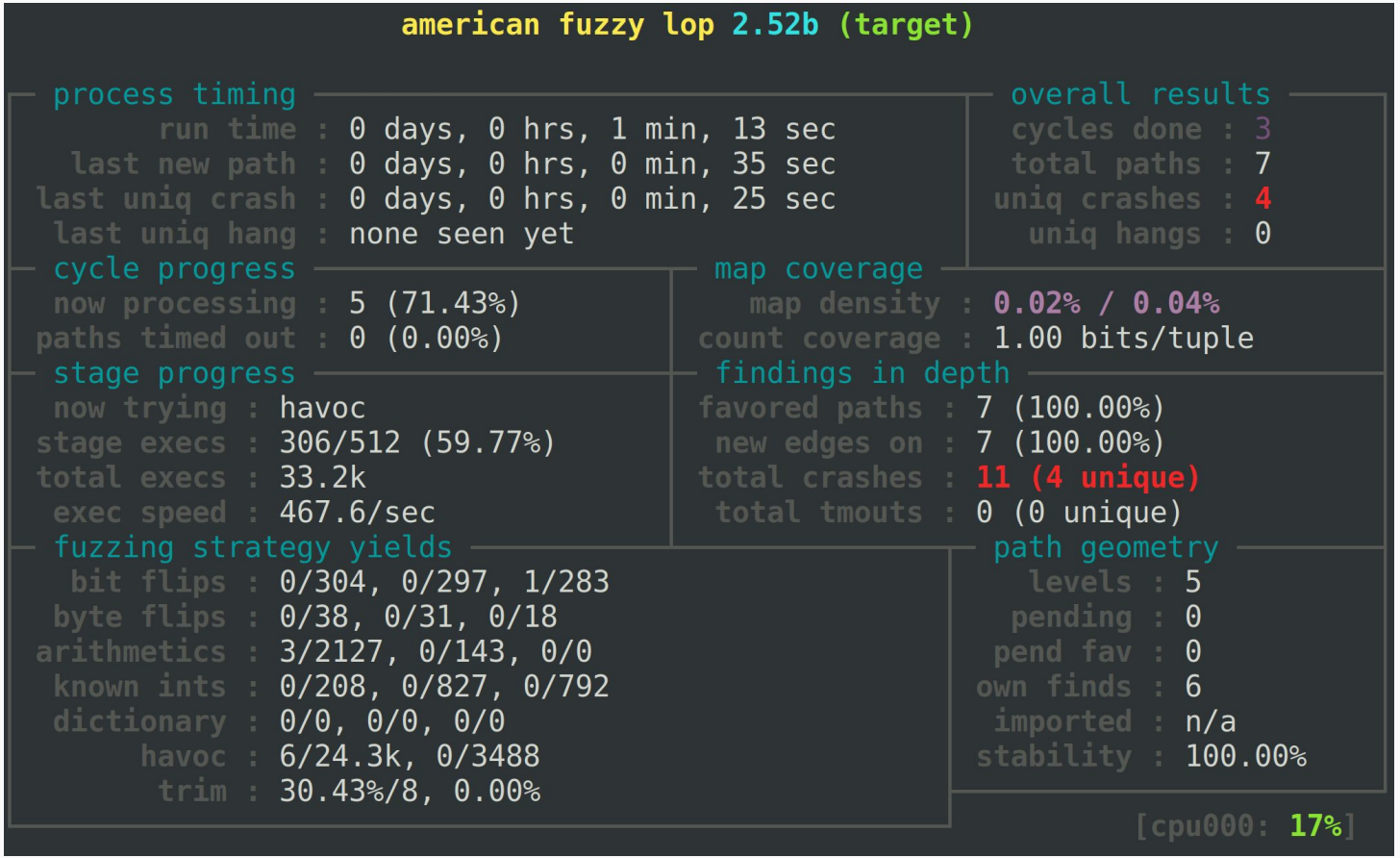
Going Beyond Coverage-Guided Fuzzing with Structured Fuzzing

Jonathan Metzman (metzman@chromium.org)

Unstructured Fuzzing = Magic



```
void Visit(const ImageFilterChild&);
void Visit(const ImageFilterParent&, const int num_inputs_required);
void Visit(const MatrixImageFilter&);
void Visit(const Matrix&, bool is_local = false);
void Visit(const SpecularLightingImageFilter&);
void Visit(const PaintImageFilter&);
void Visit(const Paint&);
void Visit(const PaintEffects&);
void Visit(const PathEffectChild&);
void Visit(const LooperChild&);
void Visit(const LayerDrawLooper&);
void Visit(const LayerInfo&);
void Visit(const ColorFilterChild&);
void Visit(const ComposeColorFilter&);
void Visit(const OverdrawColorFilter&);
void Visit(const ToSRGBColorFilter&);
void Visit(const ColorFilterMatrix&);
void Visit(const ColorMatrixFilterRowMajor255&);
void Visit(const MergeImageFilter&);
void Visit(const XfermodeImageFilter&);
void Visit(const DiffuseLightingImageFilter&);
void Visit(const XfermodeImageFilter_Base&);
void Visit(const TileImageFilter&);
void Visit(const OffsetImageFilter&);
void Visit(const ErodeImageFilter&);
void Visit(const DilateImageFilter&);
void Visit(const DiscretePathEffect&);
void Visit(const MatrixConvolutionImageFilter&);
void Visit(const MagnifierImageFilter&);
```

Bio

- Jonathan Metzman
  - Representing myself, not Google.
- Chrome Security
  - ClusterFuzz

What is Structured Fuzzing?

	Structure Unaware	**Structure Aware**
Unguided	/dev/urandom	Script that generates HTML files
Coverage-Guided	AFL/libFuzzer (Unstructured Fuzzing)	Structured Fuzzing

Why Structured Fuzzing?

More Bugs!

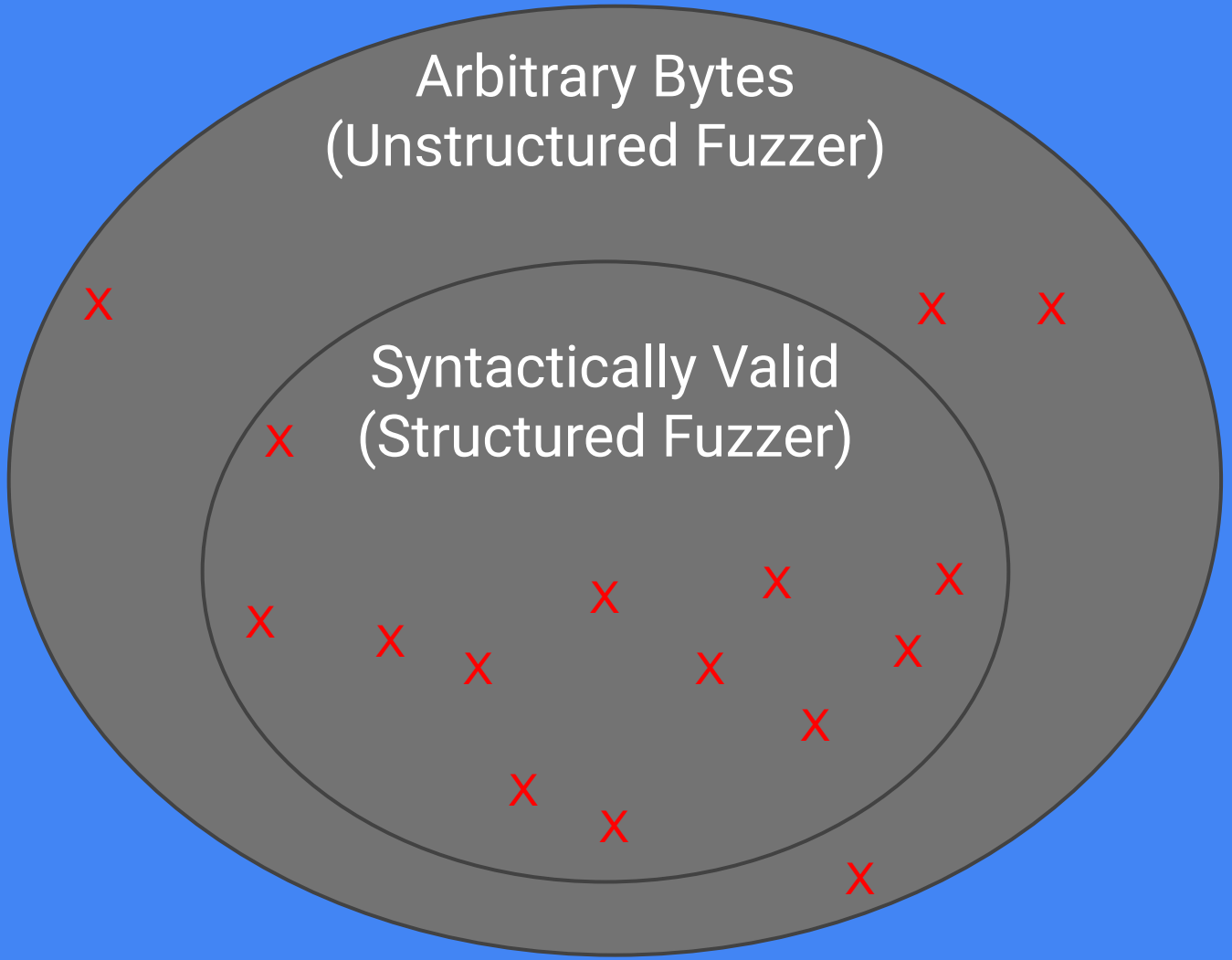
```
==158261==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60d0000023c0 at pc 0x55a4fea33789 bp 0x7ffef814b590 sp 0x7ffef814ad40
READ of size 72 at 0x60d0000023c0 thread T0
    #0 0x55a4fea33788 in __asan_memcpy /b/swarming/w/ir/kitchen-workdir/src/third_party/llvm/compiler-rt/lib/asan/asan_interceptors_memintrinsics.cc:23:3
    #1 0x7f4f564790fd in exprCodeBetween third_party/sqlite/amalgamation/sqlite3.c:100476:11
    #2 0x7f4f56472410 in sqlite3ExprCodeTarget third_party/sqlite/amalgamation/sqlite3.c:100031:7
    #3 0x7f4f56470012 in sqlite3ExprCode third_party/sqlite/amalgamation/sqlite3.c:100305:13
    #4 0x7f4f56460f9c in sqlite3Insert third_party/sqlite/amalgamation/sqlite3.c:116578:9
    #5 0x7f4f56443301 in yy_reduce third_party/sqlite/amalgamation/sqlite3.c
    #6 0x7f4f5643e209 in sqlite3Parser third_party/sqlite/amalgamation/sqlite3.c:150805:15
    #7 0x7f4f5638b257 in sqlite3RunParser third_party/sqlite/amalgamation/sqlite3.c:151965:5
    #8 0x7f4f56439d0d in sqlite3Prepare third_party/sqlite/amalgamation/sqlite3.c:123386:5
    #9 0x7f4f56389c26 in sqlite3LockAndPrepare third_party/sqlite/amalgamation/sqlite3.c:123479:10
    #10 0x7f4f56375483 in chrome_sqlite3_prepare_v2 third_party/sqlite/amalgamation/sqlite3.c:123850:8
    #11 0x55a4fea9a689 in sql_fuzzer::RunSqlQueriesOnConnection(sqlite3*, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >, std::__1::allocator<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > > >) third_party/sqlite/fuzz/sql_run_queries.cc:73:10
    #12 0x55a4fea9b1fb in sql_fuzzer::RunSqlQueries(std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >, std::__1::allocator<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > > >) third_party/sqlite/fuzz/sql_run_queries.cc:130:3
    #13 0x55a4fea646c9 in TestOneProtoInput(sql_query_grammar::SQLQueries const&) third_party/sqlite/fuzz/sql_fuzzer.cc:41:3
    #14 0x55a4fea64043 in LLVMFuzzerTestOneInput third_party/sqlite/fuzz/sql_fuzzer.cc:28:1
    #15 0x55a4feaaefb5 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) third_party/libFuzzer/src/FuzzerLoop.cpp:571:15
    #16 0x55a4feaa626c in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long) third_party/libFuzzer/src/FuzzerDriver.cpp:280:6
    #17 0x55a4feaa892b in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) third_party/libFuzzer/src/FuzzerDriver.cpp:713:9
    #18 0x55a4feab4ca2 in main third_party/libFuzzer/src/FuzzerMain.cpp:20:10
```

More Bugs: The Data

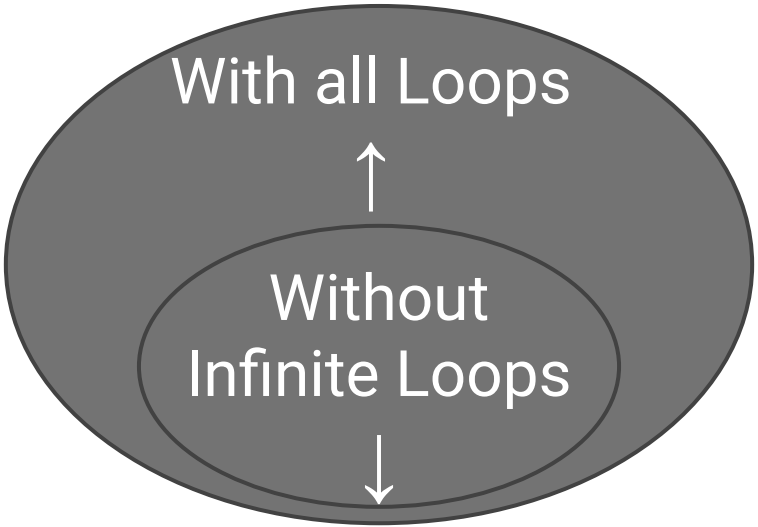
948944	High	CHECK failure: !address.is_initialized() || sizeof(*data_) == address.BlockSize() in storage_bl Reproducible allpublic Clusterfuzz
946539	High	Heap-buffer-overflow in disk_cache::EntryImpl::UserBuffer::Write Reproducible allpublic Clusterfuzz
946434	High	Heap-use-after-free in base::LinkNode<disk_cache::MemEntryImpl>::RemoveFromList Reproducible allpublic Clusterfuzz
940205	High	Heap-use-after-free in renameTokenCheckAll Reproducible allpublic Clusterfuzz
923675	High	DCHECK failure in candidate->location.IsValid() in modules.cc Reproducible allpublic Clusterfuzz
908196	High	DCHECK failure in !has_error() implies FunctionKind::kArrowFunction == next_arrow_function_kind_i Reproducible allpublic Clusterfuzz
791256	High	DCHECK failure in kNoSourcePosition != start_position() in scopes.cc Reproducible allpublic Clusterfuzz
787712	High	Use After Free (write) in SkPerlinNoiseShaderImpl allpublic Clustefuzz

Fuzz Where (you think) the Bugs are

Regehr, John. “Levels of Fuzzing.” Embedded in Academia, 22 Sept. 2018, https://blog.regehr.org/archives/1039

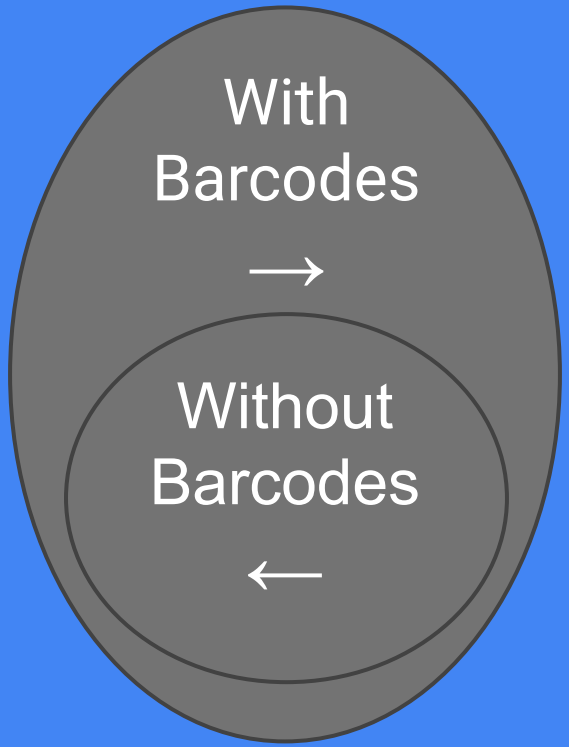

Arbitrary Bytes
(Unstructured Fuzzer)
Syntactically Valid
(Structured Fuzzer)

```
for (let i = 0; i < Infinity; i++)
  ...
doSomethingScary();  // Crash!
```



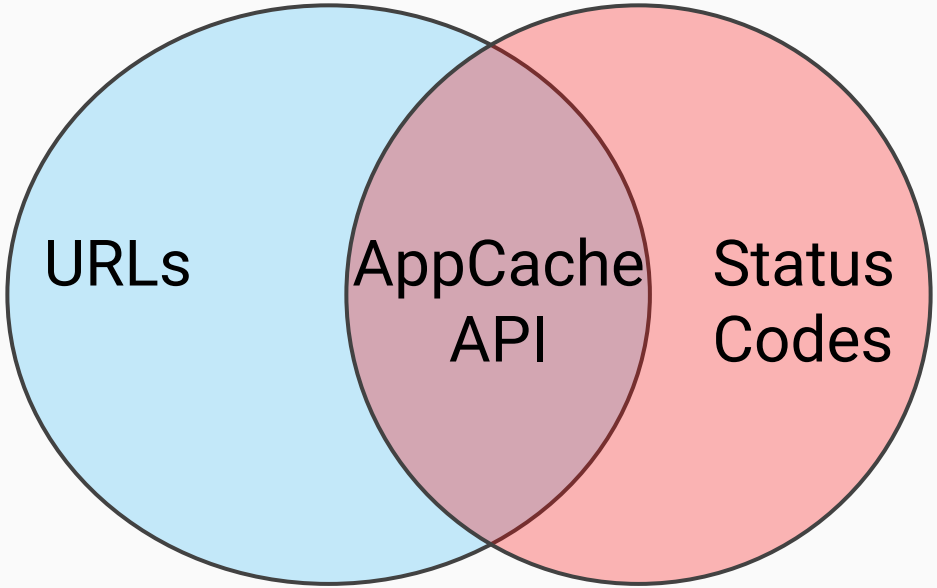

```
for (let i = 0; i < 10; i++)
  ...
doSomethingScary();  // Crash!
```

cbc_eancode.cpp	0.00% (0/44)
cbc_onecode.cpp	0.00% (0/32)
cbc_pdf417i.cpp	0.00% (0/22)
cbc_qrcode.cpp	0.00% (0/20)
cbc_upca.cpp	0.00% (0/11)



cbc_eancode.cpp	93.18% (41/44)
cbc_onecode.cpp	71.88% (23/32)
cbc_pdf417i.cpp	100.00% (22/22)
cbc_qrcode.cpp	80.00% (16/20)
cbc_upca.cpp	100.00% (11/11)

Integration Test Style Fuzzing



Williamson, Ned. Attacking Chrome IPC.
https://media.ccc.de/v/35c3-9579-attacking_chrome_ipc.

Why Structured Fuzzing?

- More bugs
- Fuzzing where the bugs are
- Integration test style fuzzing

How?

Custom Mutators

```
size_t LLVMFuzzerCustomMutator(uint8_t* data,
                               size_t size,
                               size_t max_size,
                               unsigned int seed) {
  size_t mutated_size = MutateFormat(data, size, max_size, seed);
  return mutated_size;
}
```

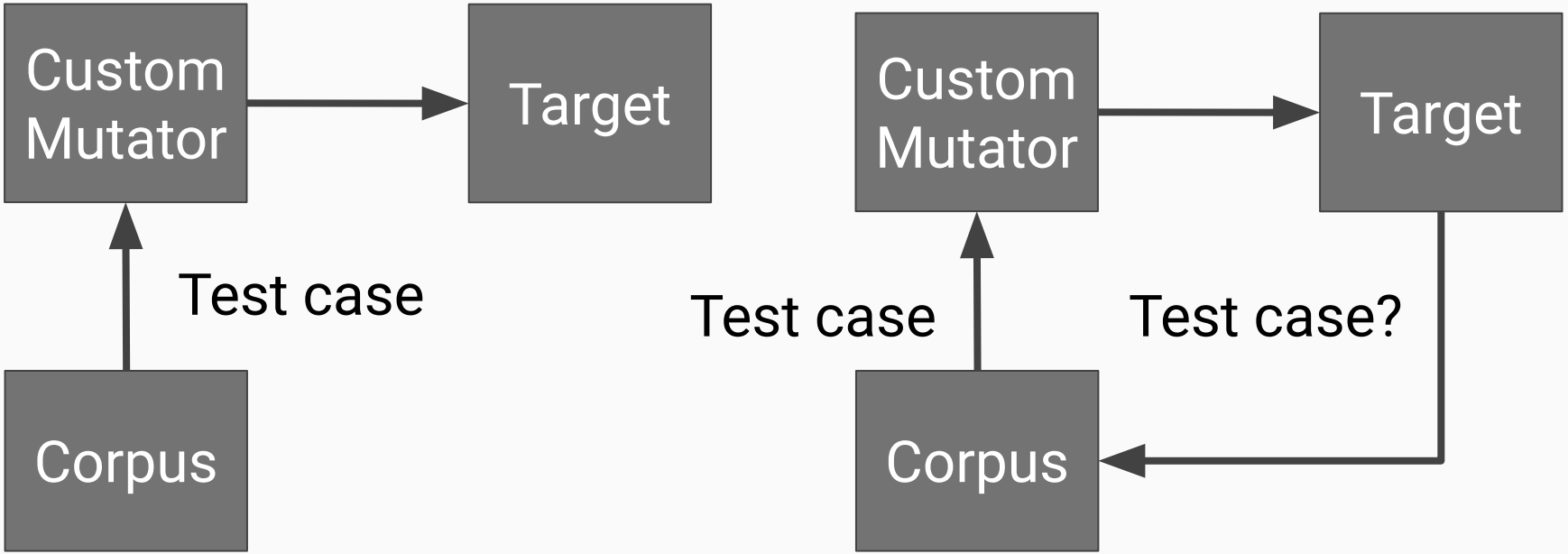
Libprotobuf-mutator

```
message AddExpression {
  int32 operand1 = 1;
  int32 oprerand2 = 2;
}
```

```
add_expression {
  operand1: 10
  operand2: 9
}
```

```
std::string ToString(const AddExpression& add_expression) {
  std::string operand1 = std::to_string(add_expression.operand1());
  std::string operand2 = std::to_string(add_expression.operand2());
  return operand1 + " + " + operand2;
}
```

Converting to a LibFuzzer Custom Mutator



How: Three Options

- Custom mutator
- Libprotobuf-mutator
- Converting to a custom mutator

Conclusion

- Find more bugs with structured fuzzing
- Use libprotobuf-mutator or custom mutators

Links

- Structure Aware Fuzzing
- Libprotobuf-mutator
- Examples:
  - Appcache fuzzer
  - Skia fuzzer
  - SQLite fuzzer