CSC 574
Computer and Network Security

Fuzzing

Alexandros Kapravelos
kapravelos@ncsu.edu

Let's find some bugs (again)

- We have a potentially vulnerable program
- The program has some inputs which can be controlled by the attacker

Can we generate automatic tests?

Fuzzing

- A form of vulnerability analysis
- Steps
  - Generate random inputs and feed them to the program
  - Monitor the application for any kinds of errors

- Simple technique
- Inefficient
  - Input usually has a specific format, randomly generated inputs will be rejected
  - Probability of causing a crash is very low

Example

Standard HTML document

- <html></html>

Randomized HTML

- <html>AAAAAAA</html>
- <html><></html>
- <html></html></html>
- <html>html</html>
- <html>/</<>></html>

Types of Fuzzers

- Mutation Based
  - mutate existing data samples to create test data
- Generation Based
  - define new tests based on models of the input
- Evolutionary
  - Generate inputs based on response from program

Mutation Based Fuzzing

- Little or no knowledge of the structure of the inputs is assumed
- Anomalies are added to existing valid inputs
- Anomalies may be completely random or follow some heuristics
- Requires little to no setup time
- Dependent on the inputs being modified
- May fail for protocols with checksums, those which depend on challenge response, etc.

- Example Tools:
  – Taof, GPF, ProxyFuzz,
  – Peach Fuzzer, etc.

Fuzzing a pdf viewer

- Google for .pdf files (about 1,640,000,000 results)
- Crawl pages and build a pdf dataset
- Create a fuzzing tool that:
  - Picks a PDF file
  - Mutates the file
  - Renders the PDF in the viewer
  - Check if it crashes

Mutation Based Fuzzing

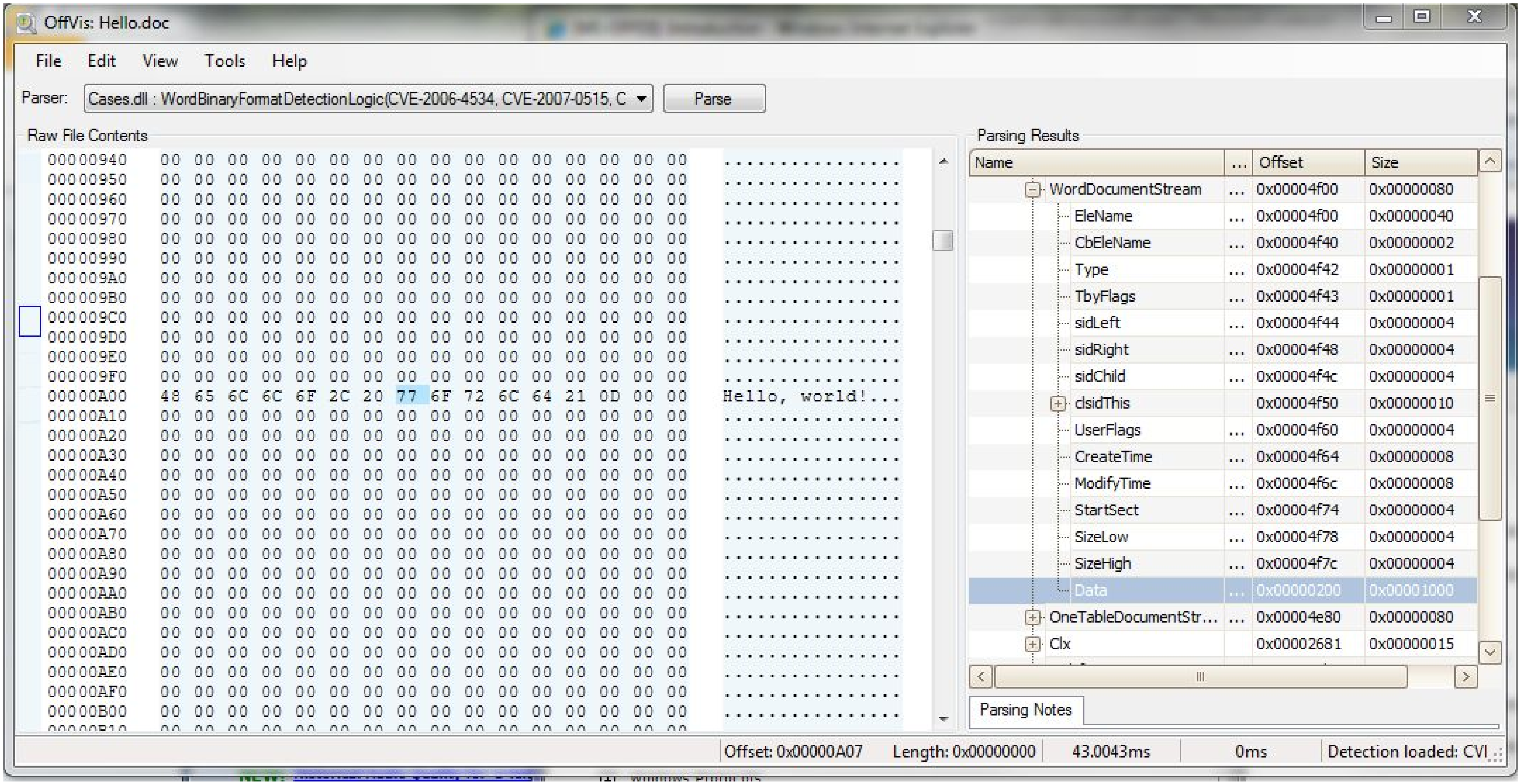
- East to setup and automate
- Little to no protocol knowledge required

- Limited to the initial dataset
- May fail on protocols with checksums, or other challenges

Generation-Based Fuzzing

- Generate random inputs with the input specification in mind (RFC, documentation, etc.)
- Add anomalies to each possible spot
- Knowledge of the protocol prunes inputs that would have been rejected by the application

Word (.doc) Binary File Format



https://msdn.microsoft.com/en-us/library/office/cc313105(v=office.14).aspx (576 pages)

Generation-Based Fuzzing

- Completeness
- Can deal with complex input, like checksums

- Input generator is labor intensive for complex protocols
- There has to be a specification

Evolutionary Fuzzing

- Attempts to generate inputs based on the response of the program

- Autodafe
  - Fuzzing by weighting attacks with markers
  - Open source

- EFS
  - Generates test cases based on code coverage metrics

Challenges

- Mutation based
  - Enormous amount of generated inputs
  - Can run forever
- Generation based
  - Less inputs (we have more knowledge)
  - Is it enough?

Code Coverage

- A metric of how well your code was tested
- Percent of code that was executed during analysis
- Profiling tools
  - Gcov

- Code coverage types:
  - Line coverage
    - which lines of source code have been executed
  - Branch coverage
    - which branches have been taken
  - Path coverage
    - which paths were taken

Fuzzing Chrome

- AddressSanitizer
- ClusterFuzz
- SyzyASAN
- ThreadSanitizer
- libFuzzer
- more...

Chrome's fuzzing infrastructure

- Automatically grab the most current Chrome LKGR (Last Known Good Revision)
- Hammer away at it to the tune of multi-million test cases a day
- Thousands of Chrome instances
- Hundreds of virtual machines

AddressSanitizer

- Compiler which performs instrumentation
- Run-time library that replaces malloc(), free()and friends
- custom malloc() allocates more bytes than requested and “poisons” the redzones around the region returned to the caller

- Heap buffer overrun/underrun (out-of-bounds access)
- Use after free
- Stack buffer overrun/underrun

- Chromium’s “browser_tests” are about 20% slower

SyzyASAN

- AddressSanitizer works only on Linux and Mac
- Different instrumenter that injects instrumentation into binaries produced by the Microsoft Visual Studio toolchain
- Run-time library that replaces malloc, free, et al.

ThreadSanitizer

- Runtime data race detector based on binary translation
- Supports also compile-time instrumentation
  - Greater speed and accuracy
- Data races in C++ and Go code
- Synchronization issues
  - deadlocks
  - unjoined threads
  - destroying locked mutexes
  - use of async-signal
  - unsafe code in signal handlers
  - others...

libFuzzer

- Engine for in-process, coverage-guided, white-box fuzzing
- In-process
  - don't launch a new process for every test case
  - mutate inputs directly in memory
- Coverage-guided
  - measure code coverage for every input
  - accumulate test cases that increase overall coverage
- White-box
  - compile-time instrumentation of the source code
- Fuzz individual components of Chrome
  - don't need to generate an HTML page or network payload and launch the whole browser

libFuzzer

```
==9896==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62e000022836 at
pc 0x000000499c51 bp 0x7fffa0dc1450 sp 0x7fffa0dc0c00
WRITE of size 41994 at 0x62e000022836 thread T0
SCARINESS: 45 (multi-byte-write-heap-buffer-overflow)
   #0 0x499c50 in __asan_memcpy
   #1 0x4e6b50 in Read third_party/woff2/src/buffer.h:86:7
   #2 0x4e6b50 in ReconstructGlyf third_party/woff2/src/woff2_dec.cc:500
   #3 0x4e6b50 in ReconstructFont third_party/woff2/src/woff2_dec.cc:917
   #4 0x4e6b50 in woff2::ConvertWOFF2ToTTF(unsigned char const*, unsigned long,
woff2::WOFF2Out*) third_party/woff2/src/woff2_dec.cc:1282
    #5 0x4dbfd6 in LLVMFuzzerTestOneInput
   testing/libfuzzer/fuzzers/convert_woff2ttf_fuzzer.cc:15:3
```

Cluster Fuzzing

ClusterFuzz uses the following memory debugging tools with libFuzzer-based fuzzers:

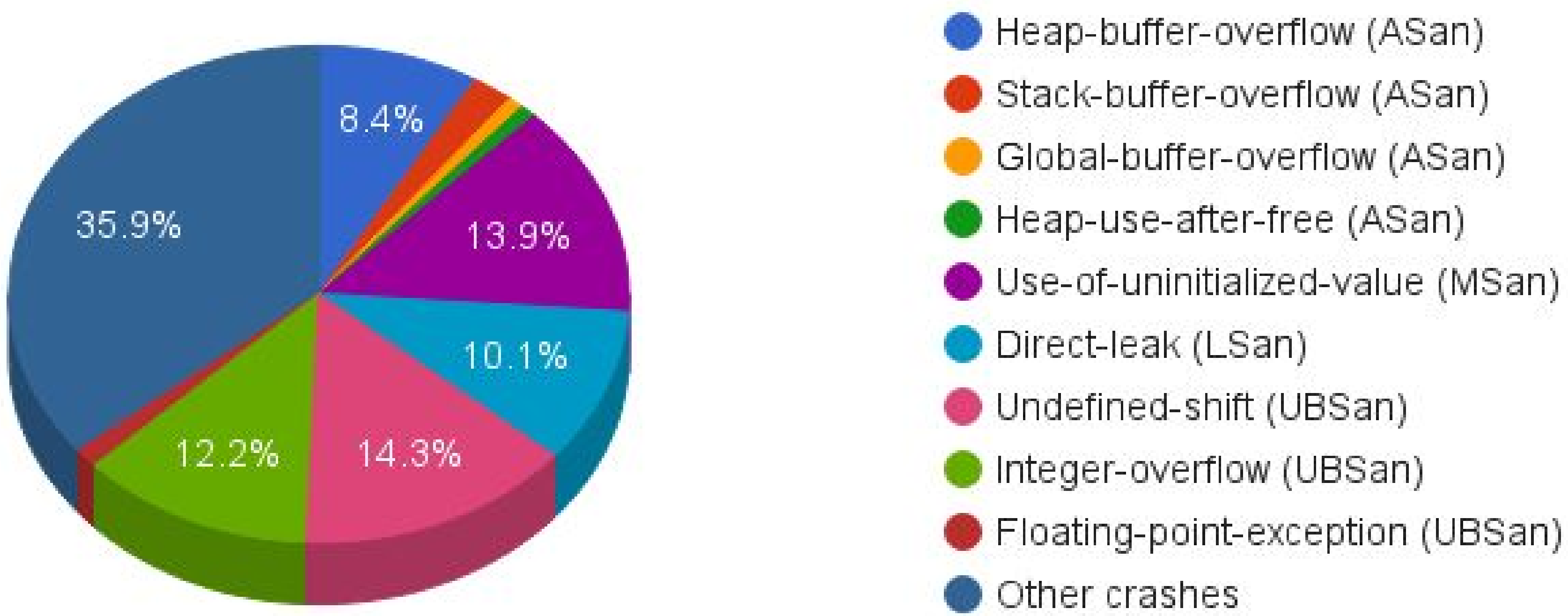
- AddressSanitizer (ASan): 500 GCE VMs
- MemorySanitizer (MSan): 100 GCE VMs
- UndefinedBehaviorSanitizer (UBSan): 100 GCE VMs

July 2016

14,366,371,459,772 unique test inputs

112 bugs filed

Analysis of the bugs found so far



Source: https://security.googleblog.com/2016/08/guided-in-process-fuzzing-of-chrome.html

Chrome's Vulnerability Reward Program

- Submit your fuzzer
- Google will run it with ClusterFuzz
- Automatically nominate bugs they find for reward payments