Fuzzing + Libfuzzer

23/06/2021 – Estevam Arantes

Um mundo de ???sZd?Ə?f???)?z¸?9

Revisando Fuzzing

Bill Sempf
@sempf

Follow

QA Engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 999999999 beers. Orders a lizard. Orders -1 beers. Orders a sfdeljknesv.

7:56 PM - 23 Sep 2014

29,331 Retweets **21,080** Likes

642 29K 21K

O que é fuzzing

Teste extensivo de aplicações

Gerar inputs baseados em um ou mais arquivos base

Tentar crashar aplicações para achar vulnerabilidades

Por quê fuzzing?

- IJG jpeg libjpeg-turbo libpng libtiff mozjpeg PHP Mozilla Firefox Internet Explorer Apple Safari Adobe Flash / PCRE sqlite OpenSSL LibreOffice poppler freetype GnuTLS GnuPG OpenSSH PuTTY ntpd nginx bash (post-Shellshock) tcpdump JavaScriptCore pdfium ffmpeg libmatroska libarchive wireshark ImageMagick BIND QEMU lcms Oracle BerkeleyDB Android / libstagefright iOS / ImageIO FLAC audio library libsndfile less / lesspipe strings (+ related tools) file dpkg rcs systemd-resolved libyaml Info-Zip unzip libtasn OpenBSD pfctl NetBSD bpf man & mandoc IDA Pro [reported by authors] clamav libxml glibc clang / llvm nasm ctags mutt procmail fontconfig pdksh Qt wavpack redis / lua-cmsgpack taglib privoxy perl libxmp radare SleuthKit fwknop [reported by author] X.Org exifprobe jhead [?] capnproto Xerces-C metacam djvulibre exiv Linux btrfs Knot DNS curl wpa_supplicant libde [reported by author] dnsmasq libbpg () lame libwmf uudecode MuPDF imlib libraw libbson libsass yara WC tidy-html VLC FreeBSD syscons John the Ripper screen tmux mosh UPX indent openjpeg MMIX OpenMPT rxvt dhcpcd Mozilla NSS Nettle mbed TLS Linux netlink Linux ext Linux xfs botan expat Adobe Reader libav libical OpenBSD kernel collectd libidn MatrixSSL jasper MaraDNS wm Xen OpenH irssi cmark OpenCV Malheur gstreamer Tor gdk-pixbuf audiofile zstd lz stb cJSON libpcre MySQL gnulib openexr libmad ettercap lrzip freetds Asterisk ytnef raptor mpg Apache httpd exempi libgmime pev Linux mem mgmt sleuthkit Mongoose OS iOS kernel

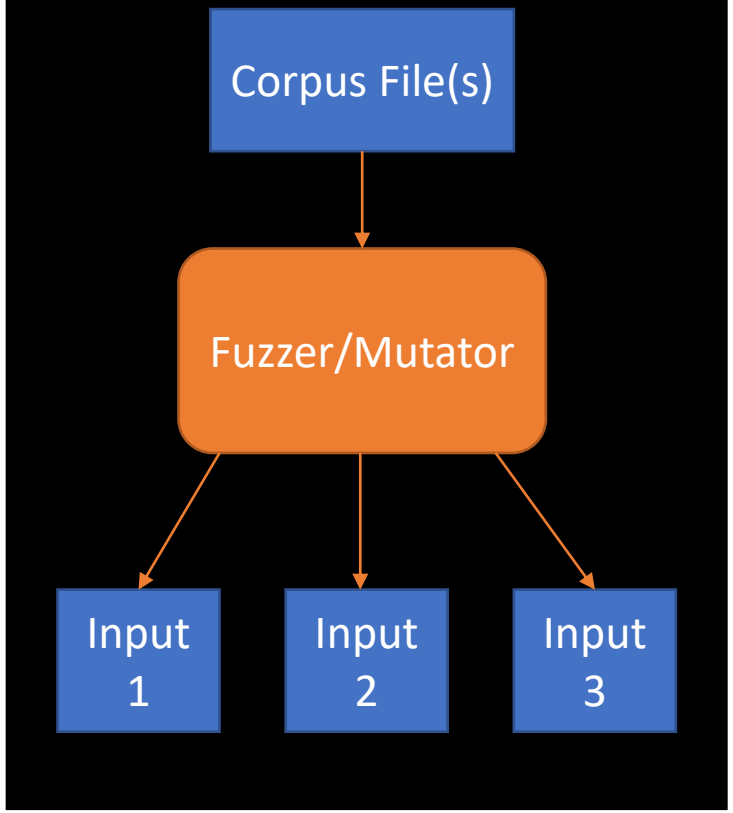
- AFL Hall of Fame

Unit testing vs Fuzz testing

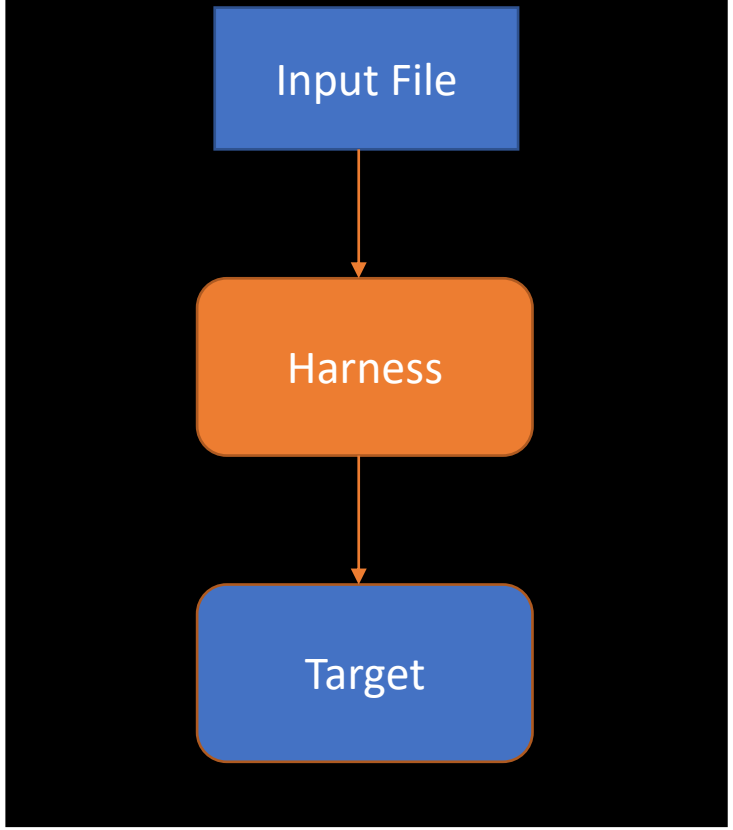
	Unit Testing	Old Fuzzing	Modern Fuzzing
Test small parts of code	✔	X	✔
Can be automated	✔	✔	✔
Regression testing	✔	✔ / X	✔
Easy to write	✔	X	✔
Looking for new bugs	✔ / X	✔ ✔ ✔	✔ ✔ ✔ ✔ ✔ ✔
Looking for vulnerabilities	X	✔	✔

Fuzzer vs Harness

- Fuzzer



- Harness



Como sei se um fuzzer é bom?

Code Coverage

code.google.com/p/go.blog/content/cover/size.go not tracked

```go
package size

func Size(a int) string {
	switch {
	case a < 0:
		return "negative"
	case a == 0:
		return "zero"
	case a < 10:
		return "small"
	case a < 100:
		return "big"
	case a < 1000:
		return "huge"
	}
	return "enormous"
}
```

Tipos de Fuzzing

Dumb Fuzzing

- Input aleatório na esperança de achar algo
- Exemplo: zzuf

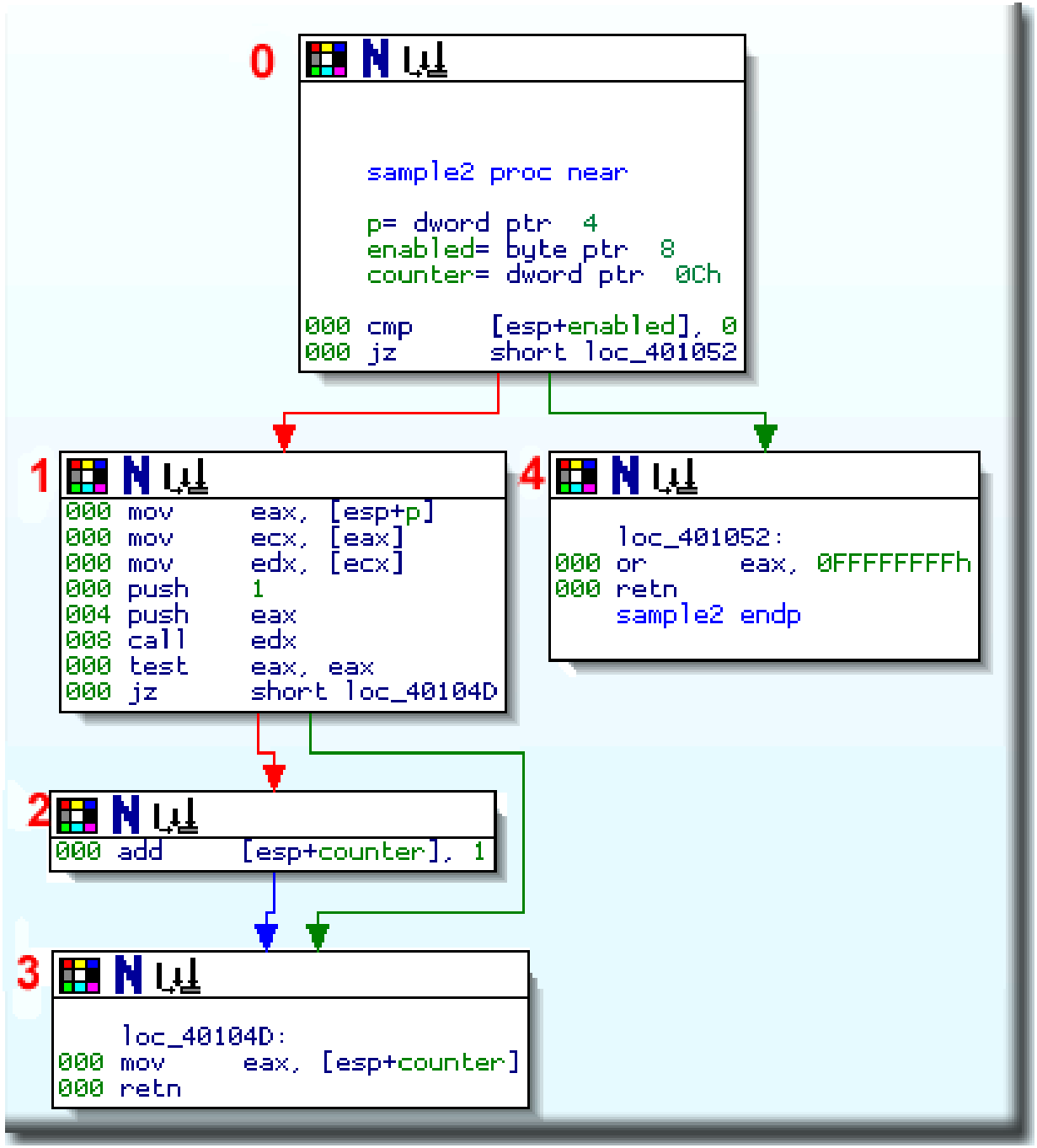
Mutational / Schema Based Fuzzing

- Baseando em um template inicial, muta ele
- Exemplo: Peach Fuzz

Instrumentation Guided Fuzzing

- Geração de entrada "evolutiva"
- Usa controle de fluxo do programa p/ mutar
- Baseado principalmente em code coverage.
- Exemplo: American Fuzzy Lop, **LibFuzzer**

Basic Blocks



AFL - American Fuzzy Lop

Desenvolvido pela Google (Michael Zalewski - @lcamtuf)

Open Source

- WinAFL, AFL++, AFL-dyninst, …

Focado em alvos com código fonte

- Também é possível fazer sem

Awesome AFL Github

```
                    american fuzzy lop 1.57b (selfs)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 1 days, 11 hrs, 14 min, 16 sec     │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 6 min, 19 sec       │  total paths : 631    │
│ last uniq crash : 0 days, 20 hrs, 53 min, 47 sec     │ uniq crashes : 4      │
│  last uniq hang : 0 days, 1 hrs, 35 min, 13 sec      │   uniq hangs : 38     │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 616 (97.62%)      │    map density : 8353 (12.75%)         │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.85 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : interest 16/8         │ favored paths : 156 (24.72%)           │
│ stage execs : 3738/8176 (45.72%)    │  new edges on : 228 (36.13%)           │
│ total execs : 8.43M                 │ total crashes : 487 (4 unique)         │
│  exec speed : 53.44/sec (slow!)     │   total hangs : 159 (38 unique)        │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 158/373k, 28/372k, 28/372k            │    levels : 7          │
│  byte flips : 0/46.6k, 1/32.6k, 2/36.7k             │   pending : 427        │
│ arithmetics : 120/1.64M, 29/1.34M, 6/385k           │  pend fav : 7          │
│  known ints : 12/148k, 44/1.00M, 52/1.79M           │ own finds : 630        │
│  dictionary : 0/0, 0/0, 13/399k                     │  imported : n/a        │
│       havoc : 108/446k, 0/0                         │  variable : 371        │
│        trim : 1.99%/20.2k, 35.69%                   ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu: 52%]
```

Libfuzzer

- Versão do projeto LLVM de Coverage Guided Fuzzing
- Mantém os casos em memória, faz operações e vê se resultou em algo
- Facilita chamadas diretas de Código
- Não faz tantas operações de disco

Usando o libfuzzer

- Implementar uma função que recebe uma string e o tamanho
  - int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size);
- Compilar com clang/clang++ usando flags de fuzzing (+ASAN)
  - clang -fsanitize=fuzzer,address target.c libs/*.cc -g
  - clang++ -std=c++11 -fsanitize=fuzzer,address target.cc libs/*.cc -g
- Executar passando pasta com corpus como parâmetro.
  - ./a.out my_corpus_dir

Clang Coverage

- Adicionar flags de compilação
  - clang++ -fprofile-instr-generate -fcoverage-mapping …
- Hit Count
  - Flags: -mllvm -runtime-counter-relocation
  - https://clang.llvm.org/docs/SourceBasedCodeCoverage.html

Coverage Reports

- Criar/Indexar as profiles geradas
  - llvm-profdata merge -sparse foo.profraw –o foo.profdata
- Visualizar o code coverage – Diversas maneiras
  - llvm-cov show ./foo -instr-profile=foo.profdata
- Flags interessantes
  - -show-line-counts-or-regions, --show-branches=count, --show-expansions
- Gerar relatório
  - llvm-cov report ./a.out –instr-profile=foo.profdata

llvm-cov documentation/command guide

Otimizando o fuzzer e seus resultados

Dicionários

- Indicação de sequências de bytes comuns no arquivo
- Facilita que o fuzzer chegue em strings importantes e passe por checagens.

Dicionário 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\""
- "<"
- "&#1;"
- "&a;"
- "&#x1;"
- "ANY"
- "[]"
- "CDATA“
- …

Memory Tools

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

- **MemorySanitizer** (aka **MSan**)
  - Detecta leitura em memória não inicializada.
  - Cpu: 3x, memória: 2x
  - Special mode: origins

- **UndefinedBehaviorSanitizer** (aka **UBSan**)
  - Detecta bugs do tipo “Undefined behavior” - Esp on type confusion, signed-integer-overflow, undefined shift, etc.
  - Cpu: 10-50%
  - Memória: ~1x (no allocator, no shadow)

Clusterfuzz

Infraestrutura para fazer fuzzing nos projetos da Google *de graça*

https://google.github.io/clusterfuzz/


WHEN YOU MADE A GOOD FUZZER

Links interessantes

- OSS-Fuzz
- Libfuzzer Tutorial
- Libfuzzer workshop
- Google/Fuzzing
- Fuzzing and Cross Checking – Openssl1.0.2d
- 50 adobe CVEs in 50 days

Talks e vídeos sobre o tema

- OffensiveCon19 - Ned Williamson - Modern Source Fuzzing (47 min)
- Fuzzing with AFL - Erlend Oftedal (45 min)
- GynvaelEN - Hacking Livestream #17: Basics of Fuzzing (1:40 hr + 1:30 hr)
- BSidesSF 113 Fuzz Smarter Not Harder An afl fuzz Primer Craig Young (50 min)
- Live Overflow – How fuzzing with AFL works! (15 + 10 min, incompleto)
- MurmusCTF - Life of an Exploit: Fuzzing PDFCrack with AFL for 0days (10 min + 8hr live)
- Brandon Falk – Aventures in Fuzzing (NYU 2018) (~1 hora)
- Gamozo Labs (Brandon Falk) – Fuzz Week (~50 horas de stream)

Como continuar?

- Tópicos Interessantes:
  - Snapshot based fuzzing
  - Generative fuzzing

- Possível roadmap:
  - Fuzzing em projetos simples (da graduação)
  - Rebuildar e tentar rodar projetos OSS-Fuzz
  - Modificar fuzzers já existentes (e.g. do OSS-Fuzz)
  - Rebuildar projetos grandes p/ rodar fuzzing
  - Fuzzing em DLLs closed source/só com símbolos