RubyKaigi 2019

Fuzzing native Ruby code with Kisaten

Ariel Zelivansky / Twistlock

About

- Security research lead of Twistlock Labs
- From Tel-Aviv, Israel
- Auditing security of cloud native and open-source projects
- Publishing write-ups
- Personally interested in Ruby

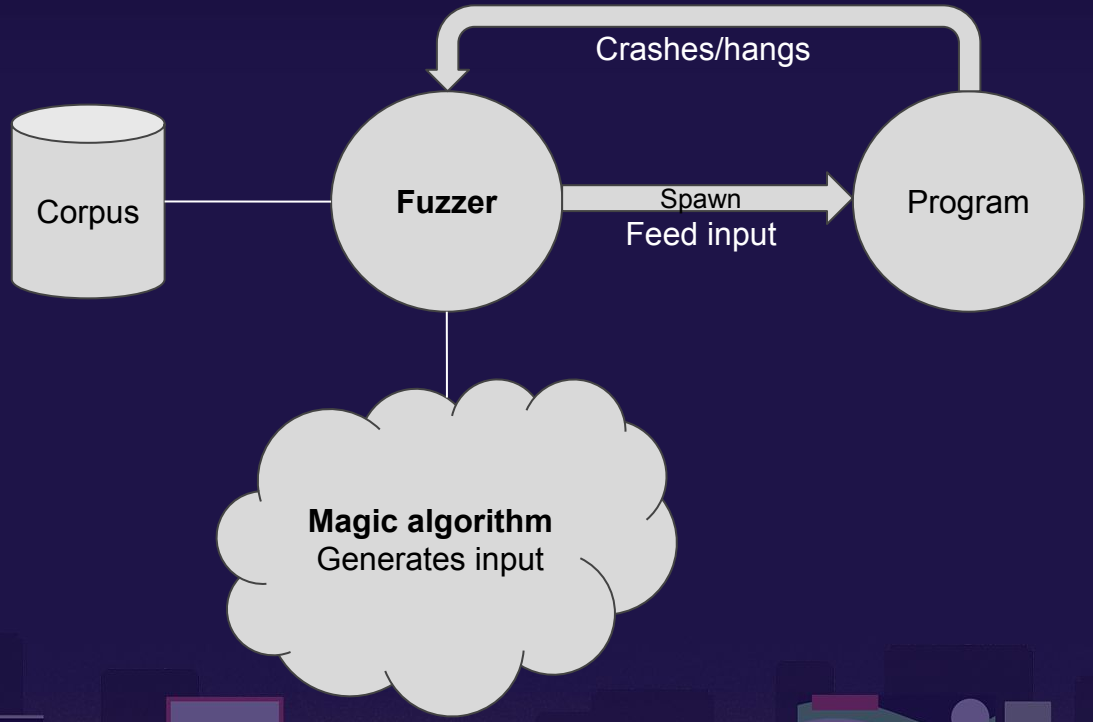
Agenda

- Introduction to fuzzing
- Ruby fuzzing
- Developing a native fuzzer
- Kisaten usage
- Kisaten future

What is fuzzing?

- Technique for testing software by providing it with random, unexpected or invalid input
- Fuzzing finds many bugs
  - Including complex bugs and security issues that can be missed by other techniques
- Fuzzers vary by
  - Source code awareness (Whitebox/greybox/blackbox)
  - Input awareness

Simple fuzzing cycle



American fuzzy lop (afl)

- Security-oriented greybox fuzzer
- Mutation-based fuzzing algorithm
  - Uses **binary instrumentation**
  - Able to to synthesize file formats
- Fast
- Easy setup
- **It works!**

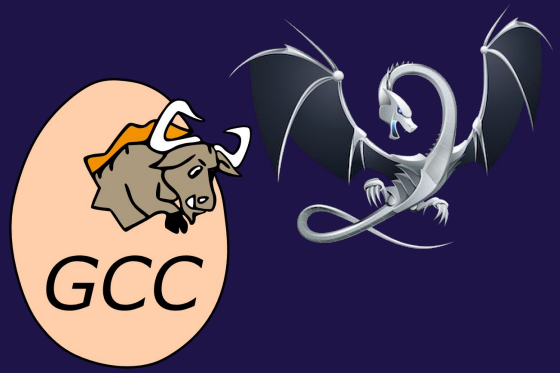
```
                american fuzzy lop 2.52b (ruby)

 process timing                                      overall results
        run time : 0 days, 0 hrs, 0 min, 7 sec        cycles done : 0
   last new path : none seen yet                      total paths : 1
 last uniq crash : 0 days, 0 hrs, 0 min, 5 sec       uniq crashes : 1
  last uniq hang : none seen yet                       uniq hangs : 0
 cycle progress                        map coverage
  now processing : 0 (0.00%)              map density : 0.00% / 0.00%
 paths timed out : 0 (0.00%)           count coverage : 1.00 bits/tuple
 stage progress                        findings in depth
  now trying : havoc                  favored paths : 1 (100.00%)
 stage execs : 946/1024 (92.38%)       new edges on : 1 (100.00%)
 total execs : 1175                   total crashes : 5 (1 unique)
  exec speed : 148.0/sec               total tmouts : 0 (0 unique)
 fuzzing strategy yields                             path geometry
   bit flips : 0/16, 0/15, 0/13                        levels : 1
  byte flips : 0/2, 0/1, 0/0                          pending : 1
 arithmetics : 0/112, 0/25, 0/0                      pend fav : 1
  known ints : 0/8, 0/28, 0/0                       own finds : 0
  dictionary : 0/0, 0/0, 0/0                         imported : n/a
       havoc : 0/0, 0/0                             stability : 100.00%
        trim : n/a, 0.00%
                                                    [cpu000: 50%]
```

IJG jpeg [1]	libjpeg-turbo [1] [2]	libpng [1]	clang / llvm [1] [2] [3] [4] [5] [6] [7] [8] ...	nasm [1] [2]	ctags [1]	dhcpcd [1]	Mozilla NSS [1]	Nettle [1]
libtiff [1] [2] [3] [4] [5]	mozjpeg [1]	PHP [1] [2] [3] [4] [5] [6] [7] [8]	mutt [1]	procmail [1]	fontconfig [1]	mbed TLS [1]	Linux netlink [1]	Linux ext4 [1]
Mozilla Firefox [1] [2] [3] [4]	Internet Explorer [1] [2] [3] [4]	Apple Safari [1]	pdksh [1] [2]	Qt [1] [2] ...	wavpack [1] [2] [3] [4]	Linux xfs [1]	botan [1]	expat [1] [2]
Adobe Flash / PCRE [1] [2] [3] [4] [5] [6] [7]	sqlite [1] [2] [3] [4] ...	OpenSSL [1] [2] [3] [4] [5] [6] [7]	redis / lua-cmsgpack [1]	taglib [1] [2] [3]	privoxy [1] [2] [3]	Adobe Reader [1]	libav [1]	libical [1]
LibreOffice [1] [2] [3] [4]	poppler [1] [2] ...	freetype [1] [2]	perl [1] [2] [3] [4] [5] [6] [7] ...	libxmp	radare2 [1] [2]	OpenBSD kernel [1]	collectd [1]	libidn [1] [2]
GnuTLS [1]	GnuPG [1] [2] [3] [4]	OpenSSH [1] [2] [3] [4] [5]	SleuthKit [1]	fwknop [reported by author]	X.Org [1] [2]	MatrixSSL [1]	jasper [1] [2] [3] [4] [5] [6] [7] ...	MaraDNS [1]
PuTTY [1] [2]	ntpd [1] [2]	nginx [1] [2] [3]	exifprobe [1]	jhead [?]	capnproto [1]	w3m [1] [2] [3] [4]	Xen [1]	OpenH232 [1] ...
bash (post-Shellshock) [1] [2]	tcpdump [1] [2] [3] [4] [5] [6] [7] [8] [9]	JavaScriptCore [1] [2] [3] [4]	Xerces-C [1] [2] [3]	metacam [1]	djvulibre [1]	irssi [1] [2] [3]	cmark [1]	OpenCV [1]
pdfium [1] [2]	ffmpeg [1] [2] [3] [4] [5]	libmatroska [1]	exiv [1] [2]	Linux btrfs [1] [2] [3] [4] [5] [6] [7] [8]	Knot DNS [1]	Malheur [1]	gstreamer [1] ...	Tor [1]
libarchive [1] [2] [3] [4] [5] [6] ...	wireshark [1] [2] [3]	ImageMagick [1] [2] [3] [4] [5] [6] [7] [8] [9] ...	curl [1] [2] [3]	wpa_supplicant [1]	libde265 [reported by author]	gdk-pixbuf [1]	audiofile [1] [2] [3] [4] [5] [6] ...	zstd [1]
BIND [1] [2] [3] ...	QEMU [1] [2]	lcms [1]	dnsmasq [1]	libbpg (1)	lame [1] [2] [3] [4] [5] [6]	lz4 [1]	stb [1]	cJSON [1]
Oracle BerkeleyDB [1] [2]	Android / libstagefright [1] [2]	iOS / ImageIO [1]	libwmf [1]	uudecode [1]	MuPDF [1] [2] [3] [4]	libpcre [1] [2] [3]	MySQL [1]	gnulib [1]
FLAC audio library [1] [2]	libsndfile [1] [2] [3] [4]	less / lesspipe [1] [2] [3]	imlib2 [1] [2] [3] [4]	libraw [1]	libbson [1]	openexr [1]	libmad [1] [2]	ettercap [1]
strings (+ related tools) [1] [2] [3] [4] [5] [6] [7]	file [1] [2] [3] [4]	dpkg [1] [2]	libsass [1]	yara [1] [2] [3] [4]	W3C tidy-html5 [1]	lrzip [1] [2] [3]	freetds [1] ...	Asterisk [1]
res [1]	systemd-resolved [1] [2]	libyaml [1]	VLC [1] [2]	FreeBSD syscons [1] [2] [3]	John the Ripper [1] [2]	ytnef [1] [2] [3] [4] ...	raptor [1]	mpg123 [1]
Info-Zip unzip [1] [2]	libtasn1 [1] [2] ...	OpenBSD pfctl [1]	screen [1] [2] [3]	tmux [1] [2]	mosh [1]	Apache httpd [1]	exempi [1] [2]	libgmime [1] [2] [3]
NetBSD bpf [1]	man & mandoc [1] [2] [3] [4] [5] ...	IDA Pro [reported by authors]	UPX [1]	indent [1]	openjpeg [1] [2]	pev [1] [2] [3] [4]	Linux mem mgmt [1]	sleuthkit [1]
clamav [1] [2] [3] [4] [5] [6]	libxml2 [1] [2] [4] [5] [6] [7] [8] [9] ...	glibc [1]	MMIX [1]	OpenMPT [1] [2]	rxvt [1] [2]	Mongoose OS [1]	iOS kernel [1]	

On top of this, the fuzzer helped make countless non-security improvements to core tools (*v8*, *sed*, *awk*, *make*, *m4*, *yacc*, *PHP*, *ImageMagick*, *freedesktop.org*, *patch*, *libtasn1*, *libvorbis*, *zsh*, *lua*, *ninja*, *ruby*, *busybox*, *gcrypt*, *vim*, *Tor*, *poppler*, *libopus*, *BSD sh*, *gcc*, *qemu*, *w3m*, *zsh*, *dropbear*, *libtorrent*, *git*, *rust*, *gravity*, *e2fsprogs*, etc); found security issues in all sorts of less-widespread software (e.g., *parrot*, *lodepng*, *json-glib*, *cabextract*, *libmspack*, *qprint*, *gpsbabel*, *dmg2img*, *antiword*, *arj*, *unrar*, *unace*, *zoo*, *rzip*, *lrzip*, *libiso**, *libtta*, *duktape*, *splint*, *zpaq*, *assimp*, *cppcheck*, *fasm*, *catdoc*, *pngcrush*, *cmark*, *p7zip*, *libjbig2*, *aaphoto*, *t1utils*, *apngopt*, *sqlparser*, *mdp*, *libtinyxml*, *freexl*, *bgpparser*, *testdisk*, *photorec*, *btcd*, *gumbo*, *chaiscript*, *teseq*, *colcrt*, *pttbbs*, *capstone*, *dex2oat*, *pillow*, *elftoolchain*, *aribas*, *universal-ctags*, *uriparser*, *jq*, *lha*, *xdelta*, *gnuplot*, *libwpd*, *teseq*, *cimg*, *libiberty*, *policycoreutils*, *libsemanage*, *renoise*, *metapixel*, *openclone*, *mp3splt*, *podofo*, *glslang*, *UEFITool*, *libcbor*, *lldpd*, *pngquant*, *muparserx*, *mochilo*, *pyhocon*, *sysdig*, *Overpass-API*, *fish-shell*, *gumbo-parser*, *mapbox-gl-native*, *rapidjson*, *libjson*, *FLIF*, *MultiMarkdown*, *astyle*, *pax-utils*, *zziplib*, *PyPDF*, *spiffing*, *apk*, *pgpdump*, *icoutils*, *msitools*, *dosfstools*, *schoco*, *MojoShader*, and so on); and is likely responsible for quite a few other things that weren't publicly attributed to the tool.

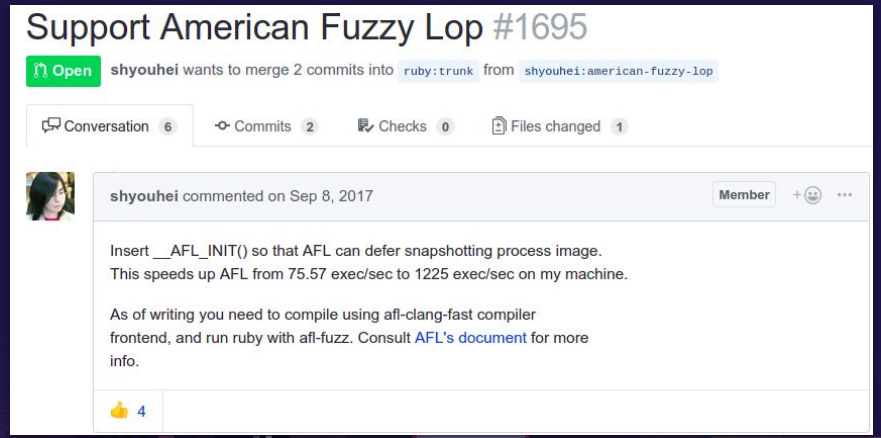
Binary Instrumentation (afl)

- Tracing program code at runtime by compilation time modification
  - Through gcc (rewriting the assembly produced)
  - Through LLVM
- No source code?
  - QEMU or Blackbox mode
- Other languages



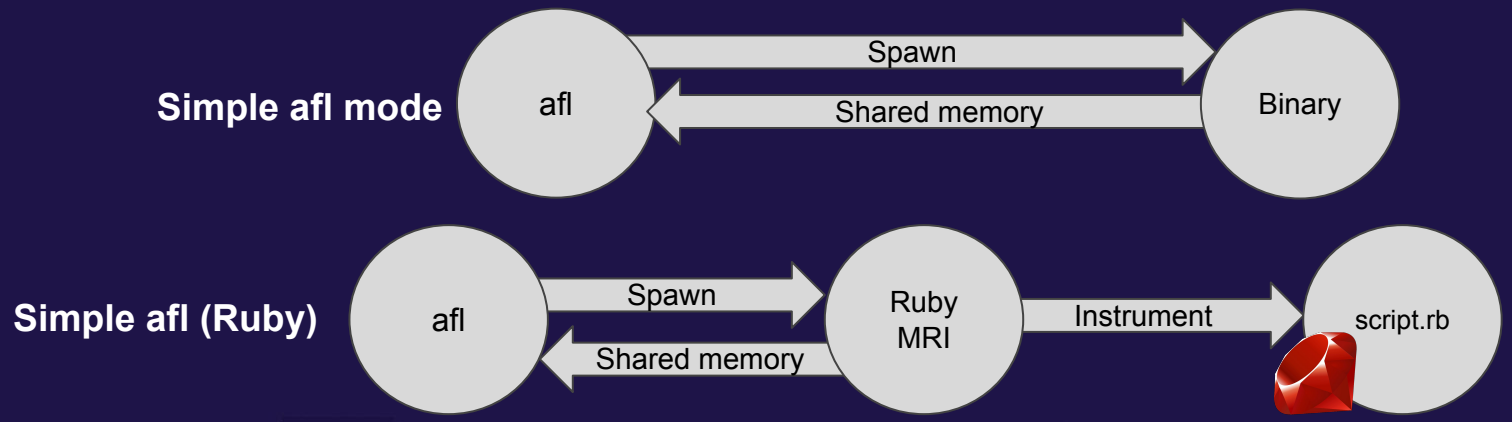
afl and Ruby

- Fuzzing the interpreter
  - Slow…!
  - Maybe good for finding bugs within the interpreter
- Fuzzing for native C Ruby extensions



afl and Ruby

- Need to instrument *native Ruby* code
- Solution - Ruby (MRI) extension to mimic afl instrumentation



Kisaten

- **Ruby gem** for fuzzing for native Ruby code
- Integrates with MRI (Matz's Ruby Interpreter) to trace Ruby lines
- Integrates with afl-fuzz for fuzzing logic
- Simple user experience
  - `require 'kisaten'`
  - `Kisaten.init`

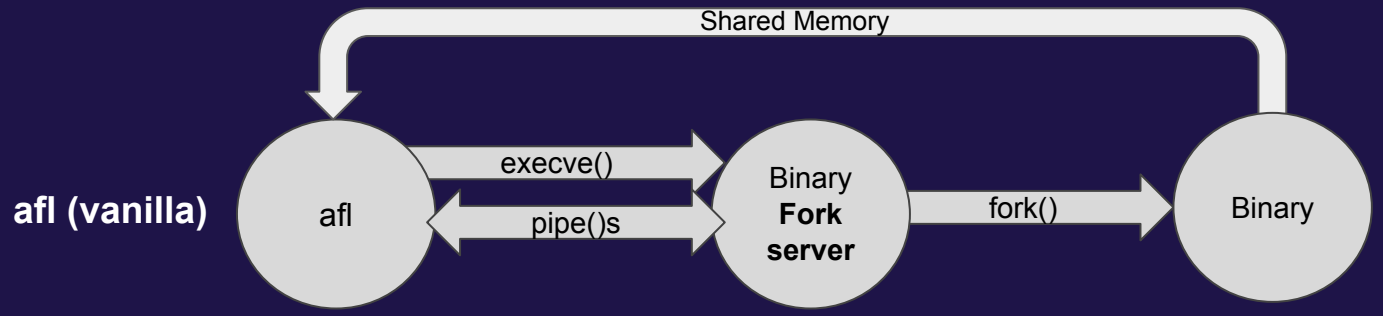
Ruby Instrumentation

- Using **TracePoint** API
  - Since Ruby 2.0
  - Similar to Kernel.#set_trace_func
- C code calls **`rb_tracepoint_new`** with `RUBY_EVENT_LINE`
- `rb_tracepoint_enable`
- Tracepoint info gives file path and line number
  - Combined into a hash that identifies each node in the execution
- Kisaten writes each tracepoint to afl's shared memory

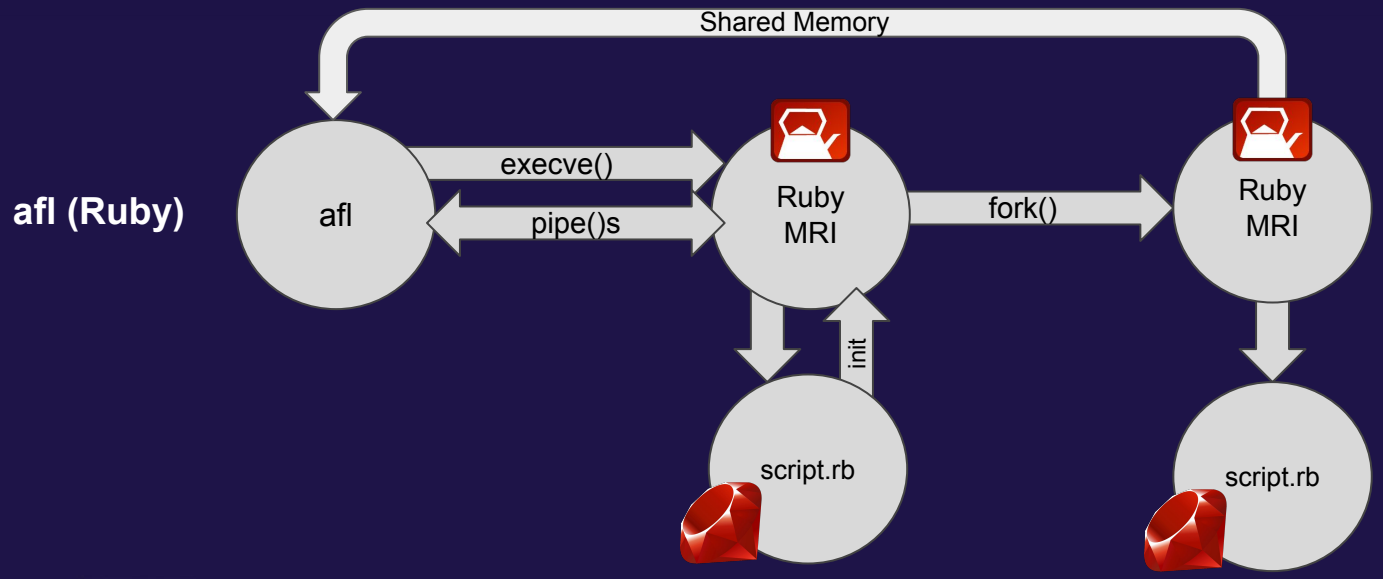
Integration with afl

- afl was ready
  - Set `AFL_SKIP_BIN_CHECK=1` environment variable
  - Prepared for python-afl development
- Shares instrumentation data through shared memory
- Implements the forkserver
  - Kisaten synchronizes with afl through pipes
    - Start, stop (timeout)
    - Return status

Forkserver Flow



Forkserver Flow



Persistent mode

```
while Kisaten.loop 10000
  gc_food = Placeholder.logic(ARGV[0])
end
```

- Purpose: run N times without forking
  - Faster fuzzing!
  - (For stateless code)
- Implemented by sending SIGSTOP signal from child Kisaten
- Forkserver sends SIGCONT to child when ready

Ruby Exceptions

- afl determines “crash” if program was terminated by **unhandled signal**
  - For example, segmentation fault signal
- For Ruby code, most bugs will result in **Exceptions**
- Current solution
  - **`RUBY_EVENT_RAISE`** tracepoints to catch all exceptions
  - Kisaten lets the user decide how to handle each exception (before init)
  - `Kisaten.crash_at [Crash array, Ignore array, Crash signal]`

```
Kisaten.crash_at [Exception], [], Signal.list['USR1']
```

Hangs

- Don’t ignore hangs!
- Hangs may indicate an infinite loop in the code
  - Or just other bugs causing suspicious slowing

Kisaten Live Demo

Bugs found

- https://github.com/twistlock/kisaten/blob/master/doc/trophy_case.md
- Found bugs in Ruby gems and Ruby Standard Library gems
- Security issues will likely be DoS

Kisaten Future

- Better crash handling
  - Catching only unhandled exceptions
- Easier fuzzing setup?
  - Find Rubyist way to integrate fuzzing with Kisaten
- **Fuzz more code**
  - Fuzz more code
  - https://github.com/twistlock/kisaten

RubyKaigi 2019

Twistlock™

THANK YOU!

Ariel Zelivansky
ariel@twistlock.com
Twistlock.com/labs
@TwistlockLabs