Innovation & Research Symposium

Cisco and Ecole Polytechnique

8-9 April 2018

CEDRIC TESSIER
INSTRUMENTATION TEAM LEADER / ctessier@quarkslab.com

Automatizing vulnerability research

to better face new software security challenges

quarkslab
SECURING EVERY BIT OF YOUR DATA

Software Security

- **Data security** depends on **secure software**
- Software contains **bugs**
- Some bugs are **vulnerabilities**
  - software intended **behaviour** can be **abused**

Vulnerabilities

- **Unknown** vulnerabilities **will be** discovered
  - so-called **0 days**
- **A lot** of them **independently** by several peoples
  - contrary to popular opinion
- 0 days **will be exploited** in the wild
  - NSA or CIA leaks
  - Ransomware (WannaCry  ETERNALBLUE)

Vulnerability research

Vulnerability research cannot be **reserved** to the **bad** guys…
… as it will give them the **advantage**

- **motive** (why)
- **attack surface** (where)
- **knowledge** (how)
- **first move** (when)

Offensive Security

From a **defensive only** security paradigm…

…to **both** defensive AND **offensive**

- Deep **complementarity**
- **Counterbalance** bad guys advantages
- **Increase** the **cost** of attacks
- Knowledge is **power**

“Ignorance has taken over
Yo, we gotta take the power back!”

Rage Against the Machine

Auditing Software

Auditing software and **finding vulnerabilities** is **crucial**

“Who looks outside, knows nothing; who looks inside, glimpses the incredible waiting to be known.”

Carl Snow

Platforms Diversity

- Huge **diversity** of **platforms**
  - toward the **end** of **Wintel** (Windows + ~~Intel~~ x86) **era**
  - **ARM**'s dominance on **mobile** markets
  - MIPS, PowerPC, [*your 90s architecture*] still kicking

Software Complexity

- **Increasing complexity** of the applications
  - **multi-megabyte** software libraries are **common**
  - **web browsers** are more like small **operating systems**
- Closed source binaries
  - very **common** in the **industry**
  - require **reverse engineering**
  - but **fewer eyes** often means **more bugs**…

Increased Difficulty

- Overall **improvements** over the past years
  - more and more **mitigations** and **compiler** enhancements
  - better development cycles (continuous bugs hunt)
- Finding **exploitable bugs** is more **difficult**
  - **low-hanging** fruits less and less common
  - yes, it’s bad news (think as a James Bond villain)

Costs

- **Bad guys** have **resources** (sometimes much more than you think)
  - criminal organizations
  - state-sponsored groups
  - military and secret services

- **Good guys** have **limited** resources (sometimes even less than you think)
  - time (money)
  - workforce

Finding vulnerabilities

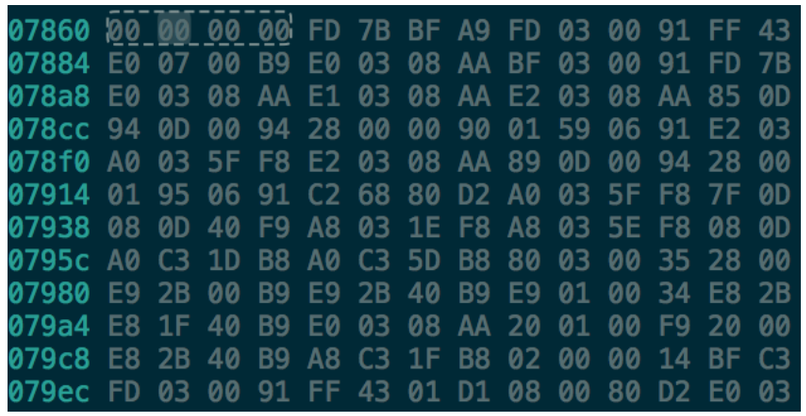
- Never-ending quest (growing code base)
- Renewed challenge (increasing difficulty)
- Competitive field (inflating investment)

New tools and **strategies** are **needed**

Innovation is **mandatory**

Binary Analysis

- Dedicated **tools**
  - disassembler
  - debugger
- Specific **techniques**
  - static analysis
  - **dynamic instrumentation**



“Transformation of a program into its own measurement tool”

- Observe **any state** of a program **anytime** during runtime
- **Automate** the data collection and processing

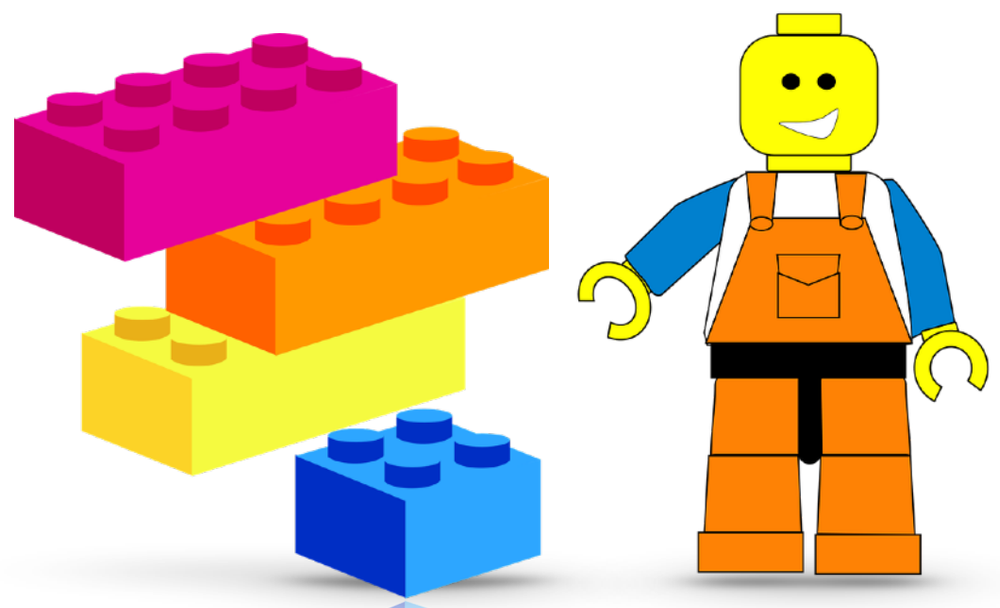
Quarksla**B** **D**ynamic binary **I**nstrumentation

- Open-source
- Cross-platform
  - macOS, Windows, Linux, Android and iOS
- Cross-architecture
  - x86_64, ARM (more to come)
- Modular design (Unix philosophy)

Give it a try! https://qbdi.quarkslab.com/

Modularity

- Only provides what is **essential**
- **Don’t force** users to do thing in **your way**
- Easy integration everywhere



Integration

```
# frida --enable-jit -l /usr/local/share/qbdi/frida-qbdi.js ./demo.bin
     ____
    / _  |   Frida 10.6.26 - A world-class dynamic instrumentation framework
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Spawned `./demo.bin`. Use %resume to let the main thread start executing!
[Local::demo.bin]-> var vm = new QBDI()
undefined
[Local::demo.bin]-> var state = vm.getGPRState()
undefined
[Local::demo.bin]-> vm.addInstrumentedModule("demo.bin")
true
[Local::demo.bin]->
```

Fuzzing

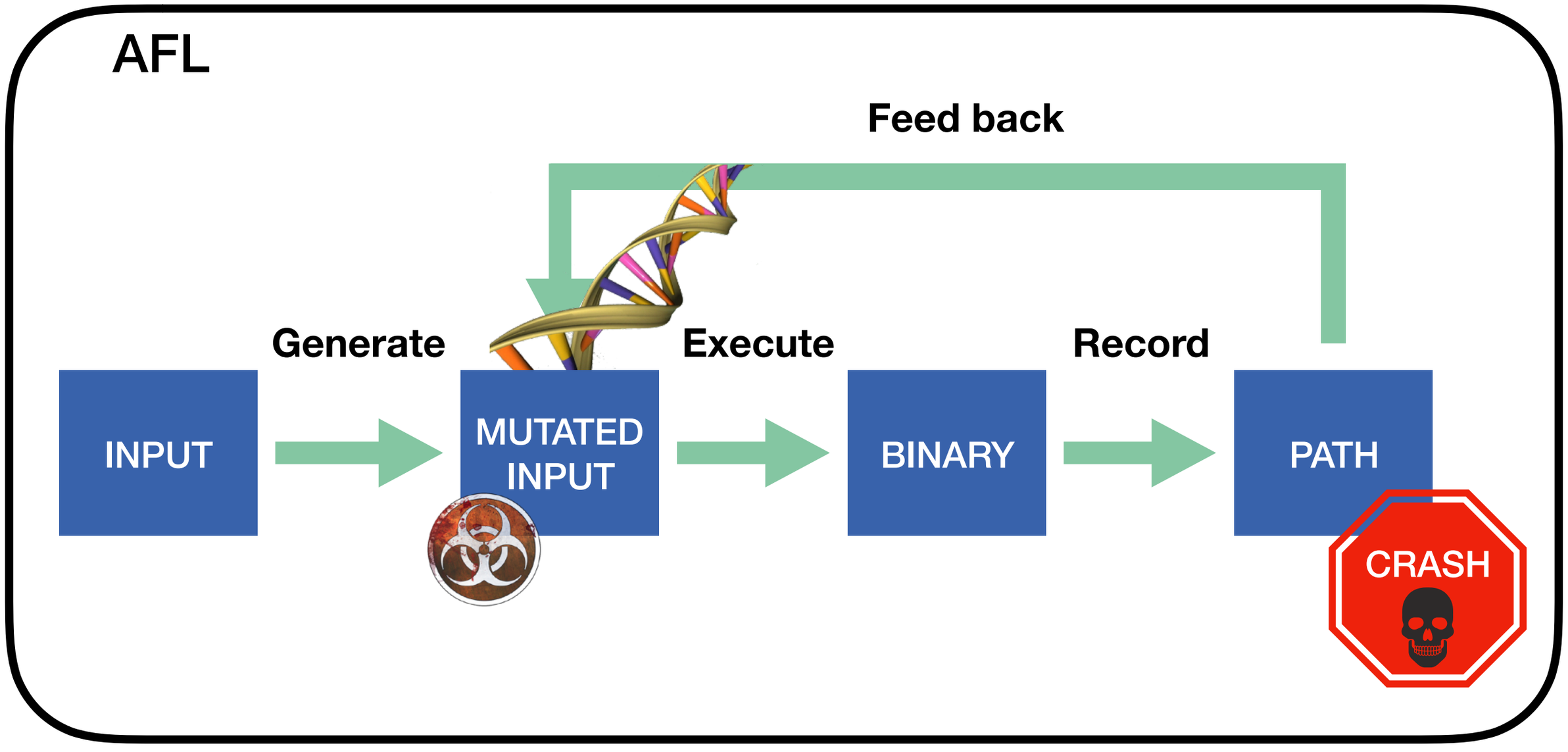
- Fuzz testing software (aka **fuzzing**)
  - injects randomized or **mutated** inputs
  - provides a way to find **bugs**
- Completely **automated**
  - input **generation**
  - software **execution**
  - crash (pre)analysis (or **triage**)

AFL

- State-of-the-art fuzzer
  - a **reference** in industry
  - impressive trophies (*openssl*, *openssh*, …)
- Open-source

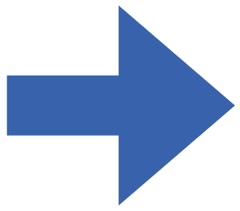
© Michał Zalewski

Guided Fuzzing



Smart Fuzzer

- Hybrid approach
  - various **brute force** strategies (input mutation)
  - **genetic** algorithm (input selection)
- Focus on **inputs** that produced **new path**
  - Maximise **code coverage** (better results)
  - Minimise **search space** (less time)



aims at better efficiency

AFL Limitations

- Pros:
  - **Fast** (scale for thousand executions per second)
  - **Efficient** (find bugs in real-world applications)
- Cons:
  - Targets **sources** are **required**
  - Portability

AFL/QBDI

AFL with **QBDI** as the **instrumentation engine**

- Targets **closed** source **binaries**
- Allows **runtime optimizations** (search space reduction)
- Reverse engineering needed (no sources)
  - often minimal but mandatory when targeting internals

Fuzzing Binaries

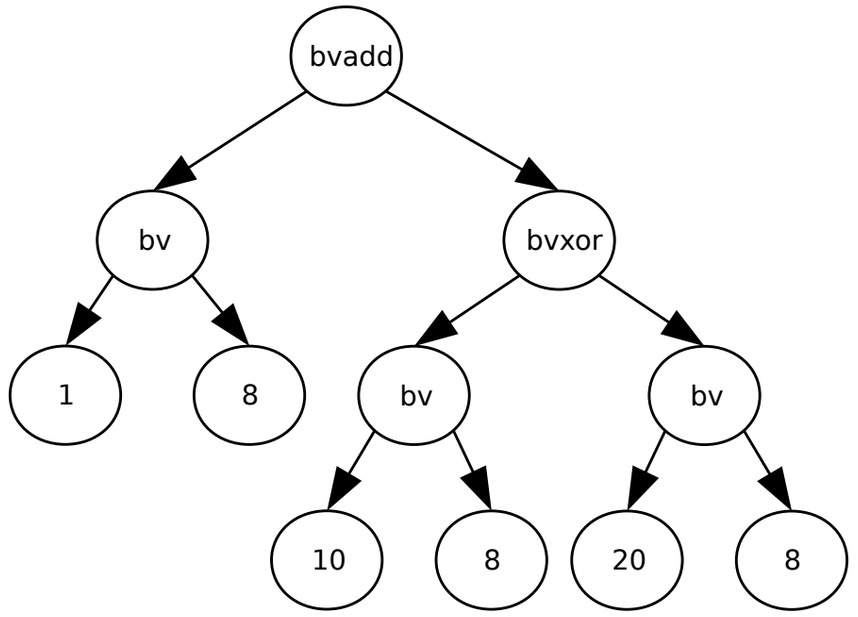
```
                american fuzzy lop 2.52b (afl-fuzz)

  process timing                                overall results
       run time :  0 days, 0 hrs, 0 min, 17 sec      cycles done :  0
  last new path :  0 days, 0 hrs, 0 min, 1 sec       total paths :  46
 last uniq crash :  0 days, 0 hrs, 0 min, 1 sec     uniq crashes :  3
  last uniq hang :  none seen yet                    uniq hangs :  0
  cycle progress                          map coverage
 now processing :  0 (0.00%)               map density :  1.61% / 2.11%
paths timed out :  0 (0.00%)            count coverage :  1.41 bits/tuple
  stage progress                          findings in depth
 now trying :  bitflip 1/1               favored paths :  1 (2.17%)
stage execs :  2092/73.5k (2.85%)         new edges on :  28 (60.87%)
total execs :  3598                      total crashes :  51 (3 unique)
 exec speed :  215.5/sec                  total tmouts :  19 (2 unique)
  fuzzing strategy yields                              path geometry
   bit flips :  0/0, 0/0, 0/0                           levels :  2
  byte flips :  0/0, 0/0, 0/0                          pending :  46
arithmetics :  0/0, 0/0, 0/0                        pend fav :  1
 known ints :  0/0, 0/0, 0/0                       own finds :  45
 dictionary :  0/0, 0/0, 0/0                        imported :  n/a
      havoc :  0/0, 0/0                            stability :  100.00%
       trim :  0.00%/1135, n/a
                                                             [cpu: 35%]
```

Symbolic Execution

- Analyzes software without running it
- Uses **symbolic values** instead of inputs (abstract interpretation)
- Represents computations as **expressions**

```
mov al, 1
mov cl, 10
mov dl, 20
xor cl, dl
add al, cl
```



Triton

Open-source **dynamic** binary **analysis** framework

- Cross-platform (macOS, Windows, Linux)
- Dynamic **Symbolic Execution** (DSE) engine
- Integrated constraints **solver** interface

Constraints Solving

- Taking a path or not depends on **conditions**
- Conditions create **path constraints**
- Symbolic expressions can represent **constraints**
- Constraints can be **solved** symbolically (SAT solvers)

```
y = input[0];
z = y - 42;
if (z == 0) {
   crash();
}
```

Improving AFL

- New kind of hybrid approach
  - **discovering** paths with AFL/QBDI
    - solve **unsatisfied** path constraints with **Triton**
- Inspired by Shellphish's **Driller**
  - used in 2016 DARPA's Cyber Grand Challenge

Scalability

- Scalability is a major challenge
  - **path explosion** (both in AFL and symbolic execution)
  - amount of **generated data**
- **Machine learning** is essential to vulnerability research
  - it is making it more **efficient** today
  - it will make it more **scalable** tomorrow

Quarkslab

SECURING EVERY BIT OF YOUR DATA