JD-HITBSECCONF 2018, BEIJING

CEDRIC TESSIER
SECURITY RESEARCHER / ctessier@quarkslab.com

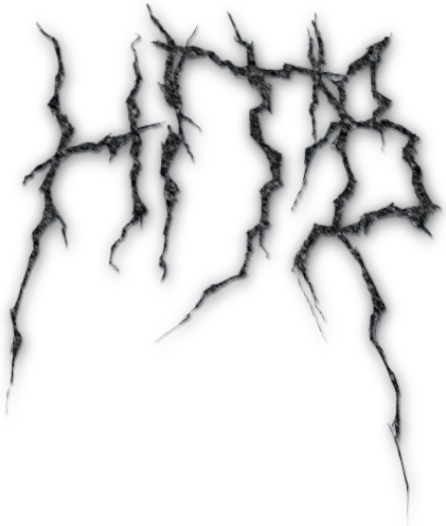
Vulnerability research: what it takes to keep going

and going and going…

Quarkslab
SECURING EVERY BIT OF YOUR DATA

Who Am I?

- Obviously **not** Fred Raynal (aka pappy)
  - No grey beard, way too young ;)
- Cédric Tessier (@nezetic)
  - One of Fred's padawans
- Dark arts enthusiast
  - Reverse engineering
  - Vulnerability research
  - Functional programming
  - Black metal



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**

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 **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)

Finding vulnerabilities

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

→ How to keep going?

What next?

Google P0 will do the job…

What do we need?

- More time, more money!
  - Our customers will sure love that one…
- More people!
  - We are recruiting ;)
- New ideas!
  - How to be smarter?
- Better tools!
  - Be more efficient

Better tools?

- Lots of **progress** during the last 10 years
- Plenty of amazing tools **available**
  - IDA
  - Frida
  - PIN
  - Clang / ASAN / libFuzzer (❤️ LLVM)
  - AFL
- More and more free and open-source

What do we dream?

Ideal tools should all be:

- Multiplatform
  - Same tools on every platforms
- Flexible
  - Adapt to exotic approaches or targets
- Efficient
  - Don’t waste resources (as we don’t have much…)
- Robust…

Reality is a…

- We need tons of things
  - And we want them now!
- Big challenges ahead
  - Development is **hard**
  - Maintaining tools even worse
- Long and tough road…
  - …and time is money

Who are we?

- French cyber-security company
  - ~50 employees
- Creating products
  - Software protection
  - Content analysis
- Providing high-end services
  - Vulnerability research
  - Reverse engineering
  - Software and hardware security

Quarkslab
SECURING EVERY BIT OF YOUR DATA

R&D

- Small private R&D lab
  - Self-financed
- Many research fields
  - Reverse engineering
  - Vulnerability research
  - Cryptography
  - Obfuscation
- Limited resources
  - Who said « long and tough road »?

Do… or do not

- Service activity
  - First hand feedbacks
  - What is really needed?
- Product activity
  - Experience in development
  - Infrastructure (Continuous Integration)
- R&D at core
  - Technical challenges are in company's DNA

Unrealistic?

- Not a multi-billion dollar company…
  - …but a small one with **specific needs**

Analysing a 20MB binary

vs

1 million of 1MB ones

Let's try to improve things…

...at least the one that **matter** to us

Binary analysis

- Many (like many many) existing tools
  - And dozen of frameworks
- All of them with limitations
  - « only support ELF file format »
- Different customers, various needs
  - « can you send us an ELF instead? »

Multiplatform? Flexibility? Efficiency?

Executable Formats

- Parsers are **fundamental** components
- Often **overlooked**
  - Seen as mandatory but boring
  - « Let's hack around libelf »
- « Easy » to create something
  - Hard to make it **last**…
- Do one thing…
  - …but do it as well as you can

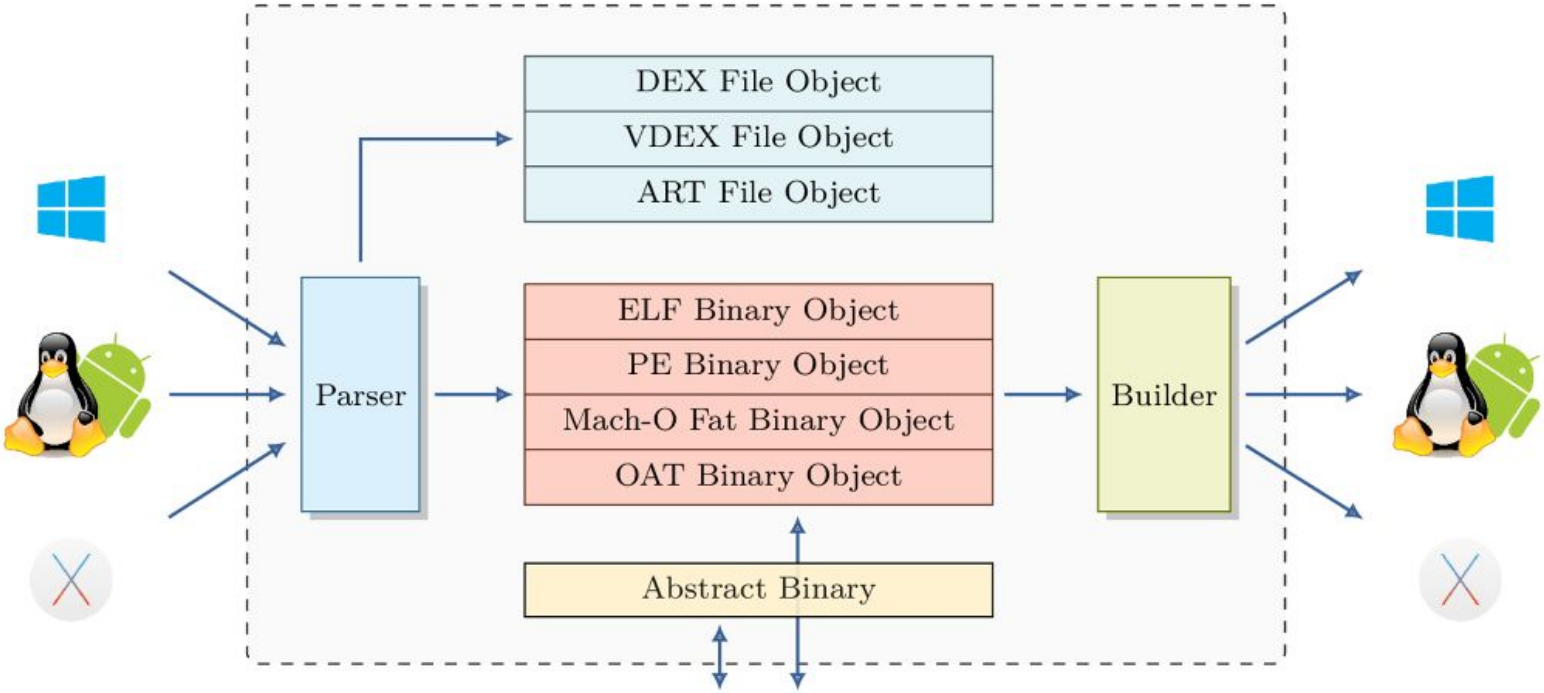
LIEF

Library to **I**nstrument **E**xecutable **F**ormats

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

- **Cross platform** library
- Parse (and **abstract**)
  - ELF, PE, MachO, DEX, OAT, ART
- Modify
  - **some** parts of these formats
- User-friendly
  - Powerful C/C++/Python APIs

One ring



Information extraction
Information adding
...

Sales Pitch

- Flexible
  - Just a (nice) library
  - Abstractions (common APIs for all formats)
- Robust (we do our best…)
  - Clean build system (cmake)
  - Continuous Integration
  - Fuzzing (integrated in CI)
- Efficient
  - Core implemented in C++
  - *pybind11* Python bindings

DBI

"Transformation of a program into its **own measurement** tool"

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

Use Cases

- Finding memory bugs
  - Allocations / deallocations
  - Accesses
- Fuzzing
  - Code coverage
  - Symbolic representation of code
- Recording execution traces
  - “Timeless” debugging
  - Software side-channel attacks against crypto

Existing Frameworks

	Valgrind	DynamoRIO	Intel Pin
Release Date	2000	2002	2004
Open Source	✅ GPLv2	✅ BSD	❌ Proprietary
Cross-platform	❌ Limited to POSIX	❌ No Darwin support	✅
Cross-architecture	✅ Relying on VEX IR	✅	❌ Only x86 and x86-64
Instrumentation abstraction	✅ Insertion of VEX IR instructions	❌ Raw assembly instrumentation	✅ Callback on specific execution events
Modular	❌	❌	❌

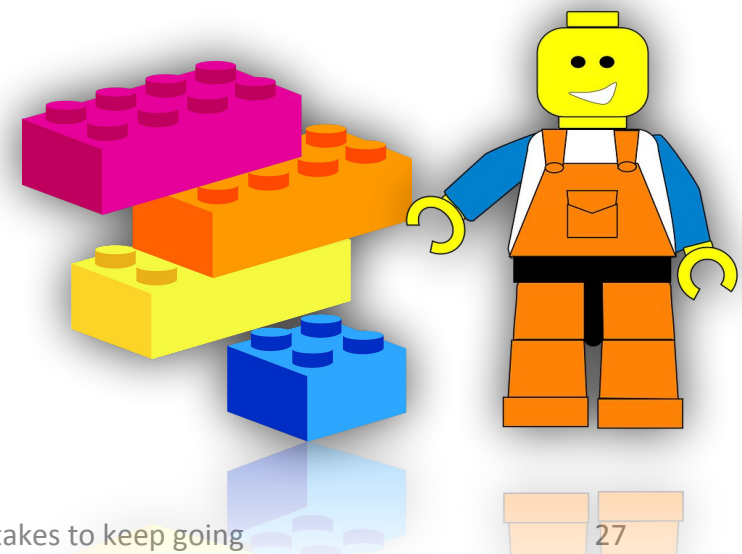
QuarkslaB **D**ynamic binary **I**nstrumentation

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

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

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

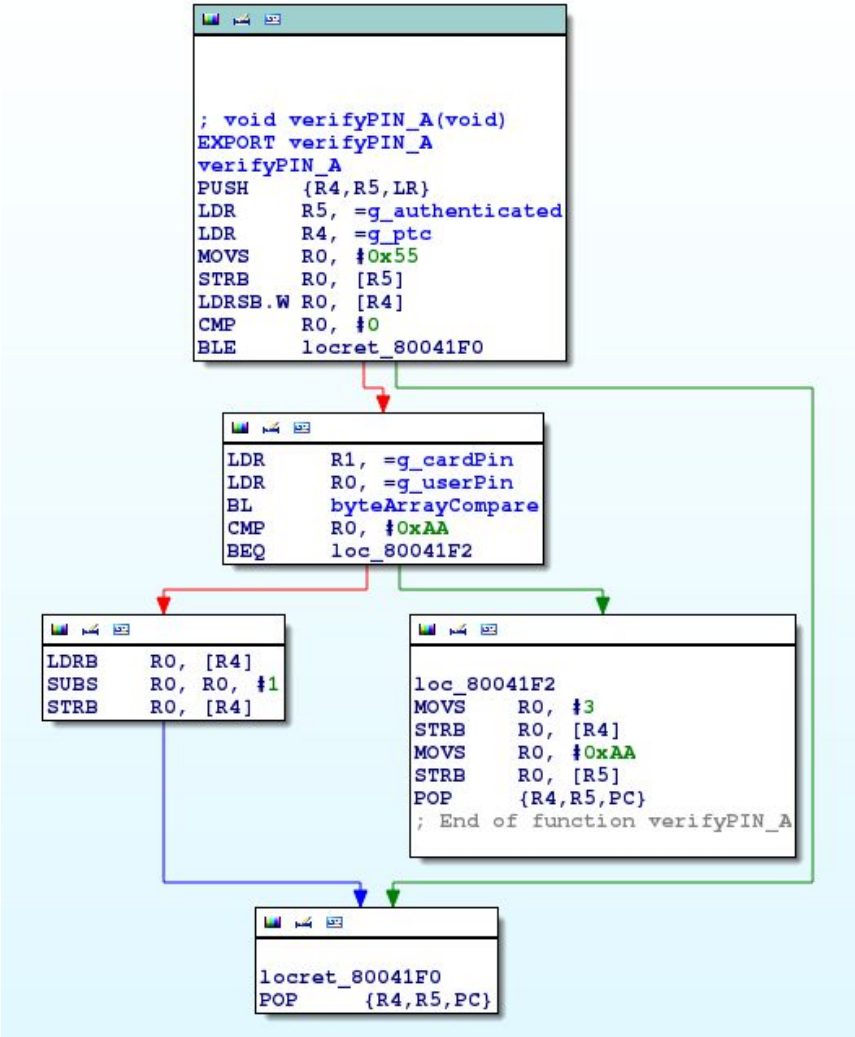
- Fuzz testing software
  - Injects randomized or **mutated** inputs
  - Provides a way to **find bugs**
- Completely **automated**
  - Input **generation**
  - Software **execution**
  - Crash (pre)analysis (or **triage**)
- « Fire and forget »
  - Nice, we lack ressources…

AFL

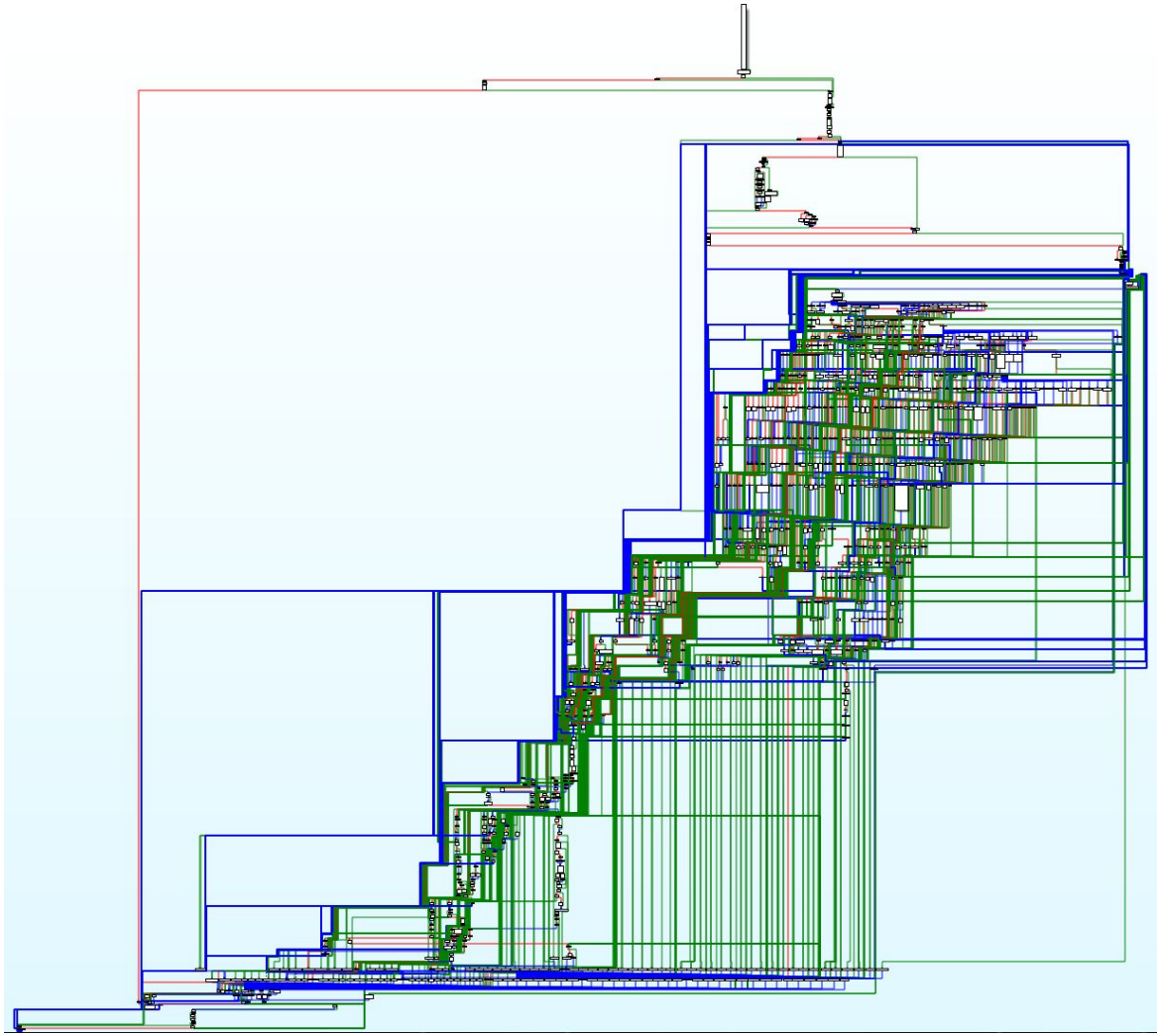
- State-of-the-art fuzzer
  - A **reference** in industry
  - Impressive trophies (openssl, openssh, …)
- Open-source

© Michał Zalewski

Code Path


; void verifyPIN_A(void)
EXPORT verifyPIN_A
verifyPIN_A
PUSH {R4,R5,LR}
LDR R5, =g_authenticated
LDR R4, =g_ptc
MOVS R0, #0x55
STRB R0, [R5]
LDRSB.W R0, [R4]
CMP R0, #0
BLE locret_80041F0
LDR R1, =g_cardPin
LDR R0, =g_userPin
BL byteArrayCompare
CMP R0, #0xAA
BEQ loc_80041F2
LDRB R0, [R4]
SUBS R0, R0, #1
STRB R0, [R4]
loc_80041F2
MOVS R0, #3
STRB R0, [R4]
MOVS R0, #0xAA
STRB R0, [R5]
POP {R4,R5,PC}
; End of function verifyPIN_A
locret_80041F0
POP {R4,R5,PC}

(not so Huge) Code Path



Smart Fuzzer

- Hybrid approach
  - Various **brute force** strategies (input mutation)
  - **Genetic** algorithm (input selection)
- Focus on **inputs** that produced **new paths**
  - 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:
  - Portability issues
  - Targets **sources** are **required**

Bad news: we rarely have sources (weird isn't it?)…

AFL/QBDI

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

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

Best Friends

- Improved **along** with QBDI
  - Better performances (raw speed)
  - On-the-fly optimizations (code coverage)
  - Memory error detection (accuracy)
  - ...
- and LIEF
  - Transform a binary in a library
  - Statically inject your fuzzer
  - Add symbols for **internal** functions
  - ...

Sales Pitch

- Easy to use C / C++ APIs
  - With proper documentation
  - Yes, it matters...
  - ...even if used internally by a few peoples
- Modular architecture
  - Various libraries (core, forkserver, loader)
  - Not drowned in a fork of AFL
- Robust build system
- Regression tests
  - A multiplatform custom memory allocator…
  - Seriously it’s painful, boring, but mandatory

“Demo”

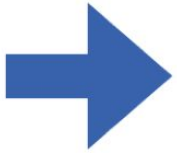
```
                    american fuzzy lop 2.52b (afl-fuzz)

  process timing                                          overall results
        run time :  0 days, 0 hrs, 0 min, 0 sec          cycles done :  0
   last new path :  none seen yet                         total paths :  1
 last uniq crash :  none seen yet                        uniq crashes :  0
  last uniq hang :  none seen yet                          uniq hangs :  0
  cycle progress                          map coverage
  now processing :  0 (0.00%)               map density :  1.61% / 1.61%
 paths timed out :  0 (0.00%)            count coverage :  1.00 bits/tuple
  stage progress                          findings in depth
  now trying :  init                     favored paths :  1 (100.00%)
 stage execs :  0/-                       new edges on :  1 (100.00%)
 total execs :  8                        total crashes :  0 (0 unique)
  exec speed :  83.33/sec (slow!)         total tmouts :  0 (0 unique)
  fuzzing strategy yields                                  path geometry
   bit flips :  0/0, 0/0, 0/0                               levels :  1
  byte flips :  0/0, 0/0, 0/0                              pending :  1
 arithmetics :  0/0, 0/0, 0/0                             pend fav :  1
  known ints :  0/0, 0/0, 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, n/a
                                                                   [cpu: 35%]
```

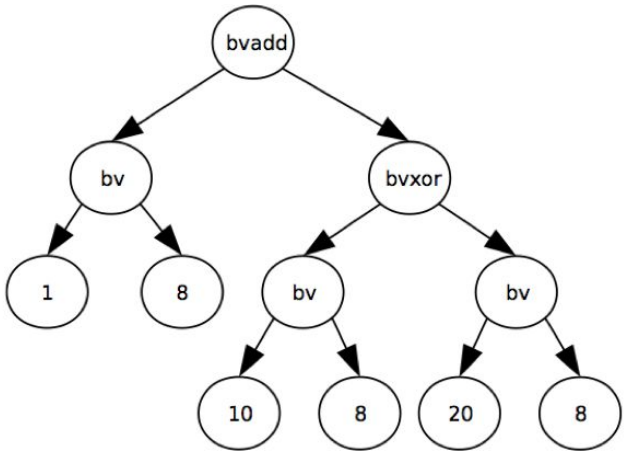
For more, enjoy Gwaby’s talk: https://www.whinysoot.com/slides/AFL_QBDI_KSE_On_a_Boat.pdf

Symbolic Execution

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

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





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/SMT solvers (like Z3)

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

→ **z == 0 ?** → **y = 42**

Triton

Dynamic Symbolic Execution Library

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

- Cross-platform
  - macOS, Windows, Linux
- x86 and x86-64
  - ARM / ARM64 in the pipeline
- Modular and easy to integrate
  - **LIEF**
  - IDA
  - **QBDI**
- Python and C++ API

Smarter Fuzzer

- New kind of hybrid approach
  - Discover paths with AFL/QBDI
  - Use symbolic execution when **stuck** (solve hard to **guess** conditions)
- Inspired by Shellphish's Driller (NDSS 2016)
  - DARPA's Cyber Grand Challenge
  - Simplified environment and constraints

To the moon

- Guided fuzzers are **fast** but not (that) smart
- Symbolic execution is **smart** but not fast

1. Find the good **ratio** between smart and fast
2. Scale on real world programs

Automation

- Fuzzing is **automating** the vulnerability research
  - Good, very good (resources?)
  - But who is automating the fuzzer?
- **Reduce** the setup and post processing times
  - Avoid repetitive and boring tasks
  - Focus only on what really matter
- Infrastructure needed

Infrastructure

- Good news:
  - Many **existing bricks** (Vagrant, Docker, …)
- Bad news:
  - Very **specific needs** (heterogeneous environments, isolation, ...)
  - Tons of bricks **missing** (orchestration, triage, ...)
  - We are not sysadmin :(

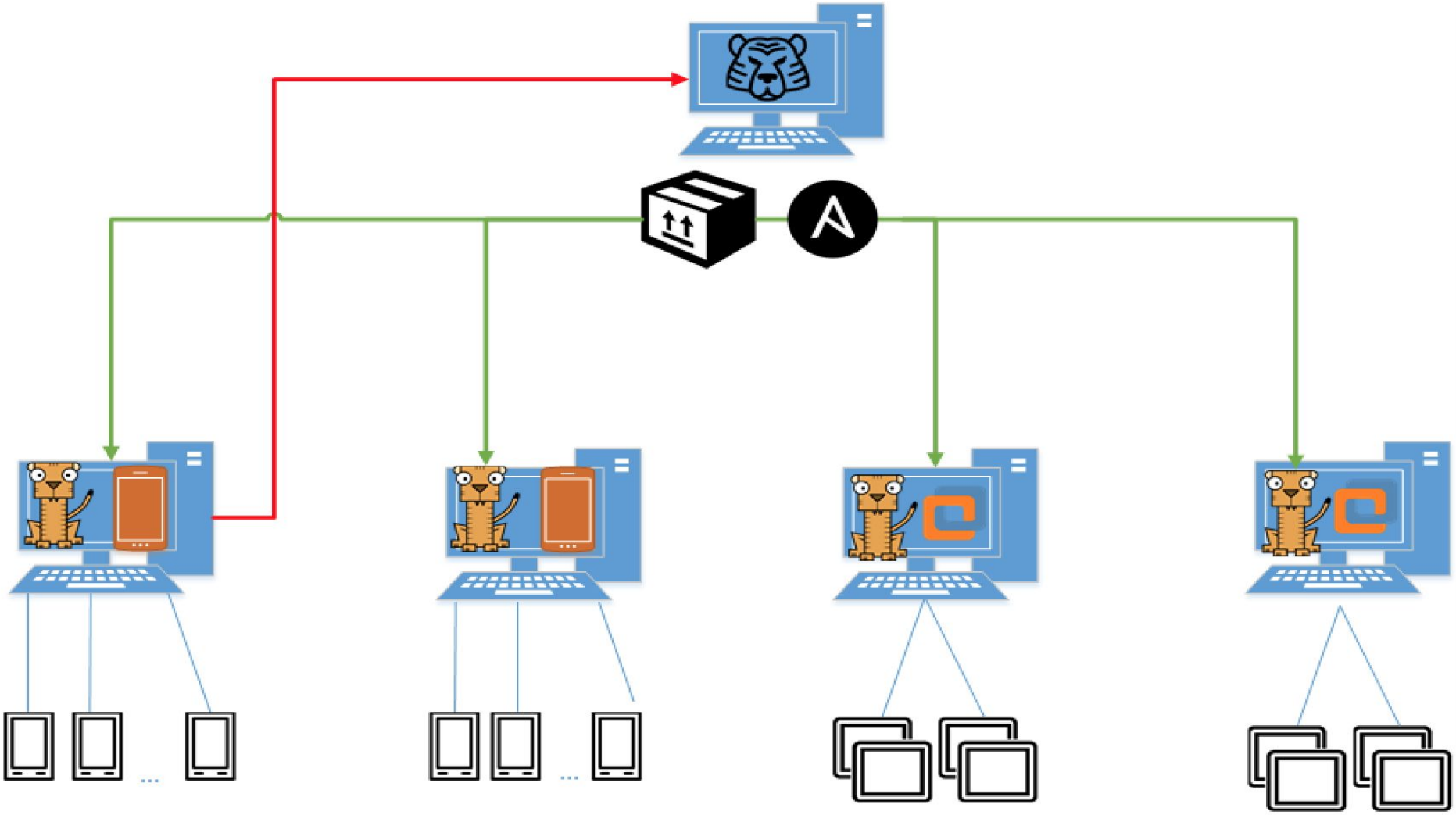
TIGRE

Terrible **I**nterface de **G**estion de **RE**ssources

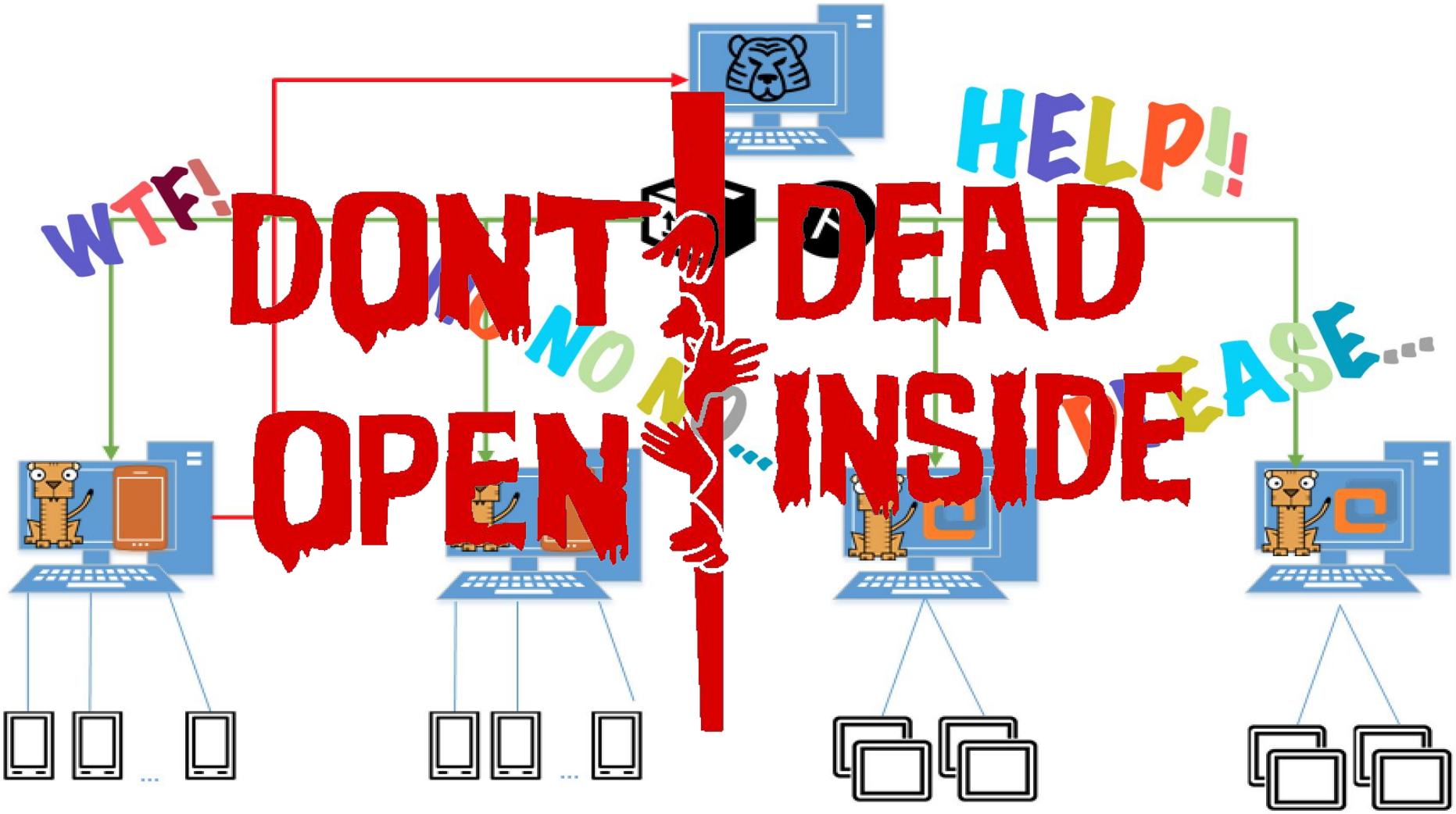
Awful Resource Management Interface ™

- Manage resources
  - Physical devices
  - **VMs**
- Configure network
  - Autodiscovery
  - **Isolation**
- Distribute jobs
  - Use resources carefully
  - Handle monitoring and **reports**

Architecture



Architecture



Dead inside

- Infrastructure automation is **hardcore**
  - Far from our core competences
  - Require very specific **skill set**
- All our goals are yet to be achieved
  - Robust
  - Scalable
  - Efficient
  - KISS 😈
  - Easy to use
  - ...

So?

- Things seem to converge
  - **Pieces** can finally be **assembled**…
  - ...and are working well together
- Amazing trip
  - Took us ~4 years…
  - ...but totally **worth it**
- Still far from the destination
  - but does it really matter?

Lessons Learned

- Vulnerability research **can't** be isolated
  - even if it always come with some secrecy
- So much to **learn** from others
  - Researchers
  - Developers
  - Sysadmins

No magic

- Security researchers are not **magicians**
  - can’t do everything by themself
- Work **smarter**, not harder
  - No pride in losing hours due to poor tooling...
  - ...yes, even if it worked
  - ...yes, even if it’s impressive
- Collaboration is key
  - Especially **interdisciplinary**

Developers!

"~~They don't care about security~~"

- Development is **hard**
  - Full time job for ~12 millions people
- To create advanced tools
  - you need specialists, experts…
  - ...who are rarely professional developers
- So much to **learn** from them
  - Code, **process**, infrastructures, …

Community

Can’t stay Alone in the Dark

- We strongly **believe** in FOSS
  - **Permissive** software licence
  - Contributors are always **welcome**
- Collaboration > Competition
- Community is **essential**
  - So much **challenges** left to overcome
  - Be nice to each others!

Quarkslab

SECURING EVERY BIT OF YOUR DATA