The Metabrik Platform - Rapid Development of Reusable Security Tools

Patrice Auffret

metabrik.org

August 2016

Whoami?

GomoR

- Information security engineer for 15 years
- *Perl* developper for same amount of time
- CPAN author
  - Net::Packet (obsolete)
  - Net::Frame suite (successor)
  - Net::Write
  - Net::SinFP/Net::SinFP3
- Speaker at security conferences
- First time speaker at YAPC
  - Wanted to share my *Perl* work

What is Metabrik?

A plateform

- A *UNIX*-like shell
- A *Perl* interpreter with *Read-Eval-Print-Loop*
- Many *Brik*s
  - A development/prototyping plateform
  - To build quickly the right tool

Why?

CLI rules

- Everything should be possibla via command line
  - Automate all the things
- *Do it once* principle
  - Tired of repeated throw-away scripts
  - Code reusability rules
- *UNIX* shells too simple
  - *Pipe* too limited
  - Needed a powerful language
- Rapid development from a *CLI*
  - Writing scripts is also possible
- Normalized Syntax in *human readable* form

Comparison with REbus

Interactive usage versus fully automatic one

- Same goal: normalize tool usage
- REbus: replace the human
  - Not a shell
  - Usage is not so easy
  - Written in Python
  - Input/output automation
- Metabrik: help the human
  - Manual input/output
  - Easy *Brik* usage
- Both are using *wrapper*s around existing tools
  - Metabrik does not only use *wrapper*s
  - It tries to use best *Perl* CPAN modules

Demo 1 - The Metabrik Shell (~3 minutes)

- 3 kinds of command lines
  - external
  - *Brik*'s commands
  - *Perl* code (*REPL*)
- 5 *Brik*'s commands
  - *use, set, get, run, help*

Features

Most notable ones

- Shell: builtins
  - cd, alias, ...
- Customizable shell with history handling
  - File *.metabrik_rc*
  - File *.metabrik_history*
- *Metasploit*-like syntax
- Completion for commands, files, variables
- Control keys like *Ctrl+R*

Briks

You have the knowledge, detail is in the *Brik*

- Many *wrapper*s around external programs
  - but not only
- Object-oriented
  - a *Brik* may inherit from one or more others
  - Example: file::csv, file::psv, ...
- Add features to existing tools
  - There is always one lacking
  - Example: *scalpel*
- A reusable *Perl* module
  - a command line interface
  - a classic interface

brik::tool *Brik*

Dependency handling

- Two kind of dependencies
  - System packages
  - *Perl* modules
- As easy to install as:
  - run brik::tool install network::nmap
  - use network::nmap
  - help network::nmap
- As easy to update as:
  - run brik::tool update_core
  - run brik::tool update_repository
  - run brik::tool update

Special variables

Where we keep some *Perl* philosophy

- Example:
  - run shell::command capture ls /
  - my $count = scalar(@$RUN)
- Input/output handling via $RUN
  - This is the new pipe
  - *Perl* basic data types
  - You sculpt it to your needs
- Other special variables
  - $SET
  - $GET
  - $CON, $LOG, $GLO, $SHE
  - $USE, $ERR, $MSG, $REF

Demo 2 - forensic challenge (~3 minutes)

Or how to quickly solve a problem

- Some miscreants kidnapped your cat
- We found an old device on crime-scene
- We have to analyze this data
- File analysis
  - *file::type*
  - *file::compress*
  - *image::exif*
- Extract data
  - *forensic::scalpel*

brik::tool and brik::search

Your best friends

- More than 200 *Brik*s...
- *brik::search* to the rescue
  - search by tag, command, category or string...
  - run brik::search tag video
- *brik::tool* for management
  - create a skeleton of a new *Brik*
    - run brik::tool create_brik my::first
  - create a skeleton of a new program
    - run brik::tool create_tool my_tool.pl

A Metatool

From prototype to industrialisation

- Finalized prototype
  - run brik::tool create_tool iplocation.pl
- Shell commands conversion to code

```
1  # Shell Metabrik
2  use lookup::iplocation
3  run lookup::iplocation from_ip 93.184.216.34
4
5  # Perl program
6  use Metabrik::Core::Context;
7  my $con = Metabrik::Core::Context->new;
8
9  use Metabrik::Lookup::Iplocation;
10 my $li = Metabrik::Lookup::Iplocation->new_from_brik_init($con);
11 my $h = $li->from_ip($ip);
```

Demo 3 - automate malware analysis (~3 minutes)

Or how to extract *Indicators of Compromise*

- Use a *VM* as a scapegoat (sacrifice it)
- Take a fingerprint of its memory/process/registry before
- Run a malware
- Take a fingerprint of its memory/process/registry after
- Instrumentalise a *VM* and take a snapshot
  - *system::virtualbox*
- Execute program remotely
  - *remote::winexe*
  - *remote::wmi*
- Perform a diff on a *Windows* machine-state
  - *forensic::volatility*

How to contribute?

No github, but mercurial/trac

- hg clone https://www.metabrik.org/hg/core
- hg clone https://www.metabrik.org/hg/repository
- Trac: http://trac.metabrik.org/

Conclusion

- *Production-ready*
- More than 200 *Brik*s ...
- Blog: https://www.metabrik.org/
- Code: http://trac.metabrik.org/browser
- Twitter: @Metabrik

Question(s)?

Metabrik

There is a Brik for that.

- https://www.metabrik.org/install/
- @Metabrik
- @PatriceAuffret