

# How automatisisation can improve firmware analysis ?

---

*Introduction to Pyrrha: a firmware mapper*



Eloïse Brocas <[ebrocas@quarkslab.com](mailto:ebrocas@quarkslab.com)>  
Jean-Marc Bourgeois <[jbourgeois@quarkslab.com](mailto:jbourgeois@quarkslab.com)>  
Robin David <[rdavid@quarkslab.com](mailto:rdavid@quarkslab.com)>

Quarkslab

# Quarkslab

Securing every bit of your data



Société de cybersécurité française  
fondée en 2011



€ 7,5m de levée de fond 2020



100+ employés  
20% Docteurs



Bureaux en France  
et en Argentine



Sécurité offensive et défensive  
Audit de sécurité et R&D appliquée  
SW, FW, HW  
Blockchain, bibliothèques cryptographiques  
Formation  
Certification de sécurité (CESTI - CSPN)

## R&D

Ouvrir la voie aux futures innovations  
utilisées dans le développement de  
**produits** et **services**

## Produits



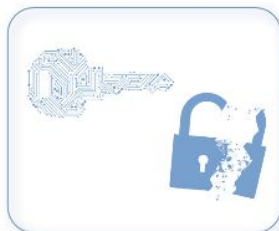
Protection logicielle du  
code, des données et des  
clés



Plateforme d'analyse  
automatique de fichiers



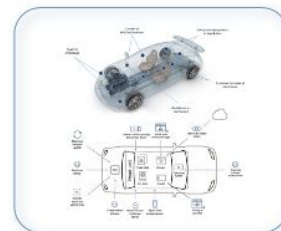
**OBJET CONNECTÉ**  
CONSEIL ET AUDIT  
DE SÉCURITÉ



**CRYPTOGRAPHIE**  
CONSEIL ET AUDIT  
DE SÉCURITÉ



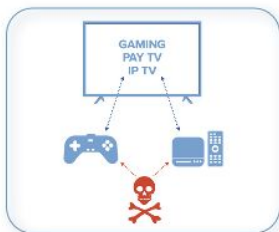
**CSPN**



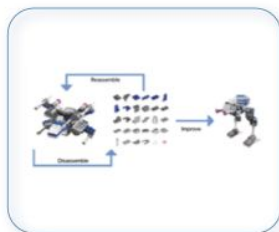
**AUTOMOTIVE**  
CONSEIL ET AUDIT  
SÉCURITÉ



**TECHNICAL  
DUE DILIGENCE**



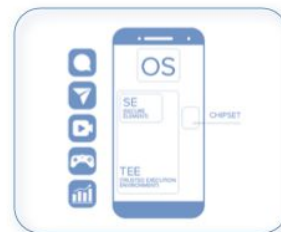
**PROTECTION  
PROPRIÉTÉ  
INTELLECTUELLE**



**RECHERCHE  
EXTERNALISÉE**



**CLOUD  
AUDIT  
D'ENVIRONNEMENT**



**ÉCOSYSTÈME  
MOBILE  
CONSEIL ET AUDIT  
DE SÉCURITÉ**



**BLOCKCHAIN  
CONSEIL ET AUDIT  
DE SÉCURITÉ**

[illegible]

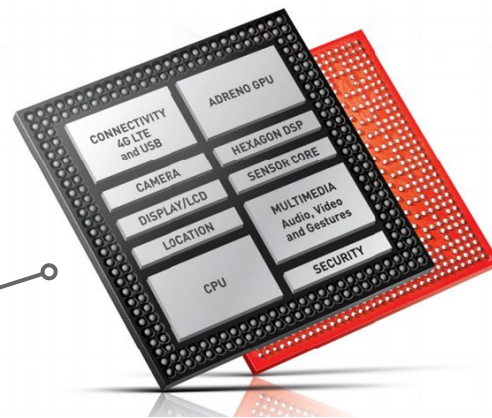
## Definition: Firmware Package

Set of files to update the various components of a whole system.

⇒ Fully-featured OS:

- base operating system
- various libraries, SDKs etc
- vendor application specific code

⇒ Can be composed of **multiple sub-firmwares** (OSes) for the different components of the SoC (*for instance*).



**Problematic:** Aggregate of components, binaries, files coming from various **third-parties**.



# Why Analyzing Firmwares ?

- Finding vulnerabilities
- Artifacts left (*private keys, build files, symbols*)
- Compliance:
  - checking patches are applied
  - checking firmware signatures (*trust management*)
- Bill-of-Materials

# Exemple: Netgear RAX30



## Components:

- SoC ARM
- Linux (*base OS*)
- network configuration softwares
- web administration panel
- firmware upgrade features

## Metrics: (2275 files)

- 713 Executable files
  - 255 579 functions
  - 12,612,710 asm instructions
- 1562 Files
  - 152 php files
  - 125 shell scripts
  - ...



Netgear RAX30

# Analysis Workflow





# Analysis Workflow



## #1 Acquisition



firmware

# Analysis Workflow

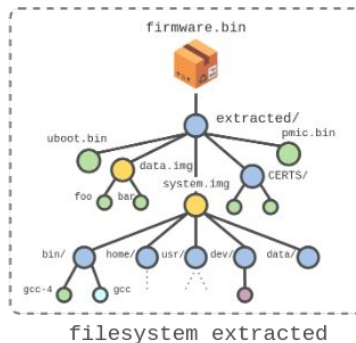


## #1 Acquisition



firmware

## #2 Extraction



# Analysis Workflow

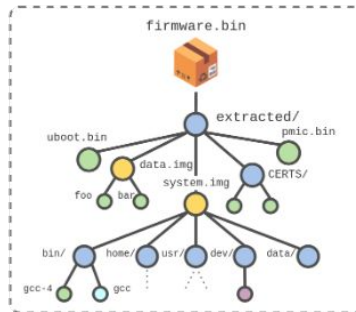


## #1 Acquisition



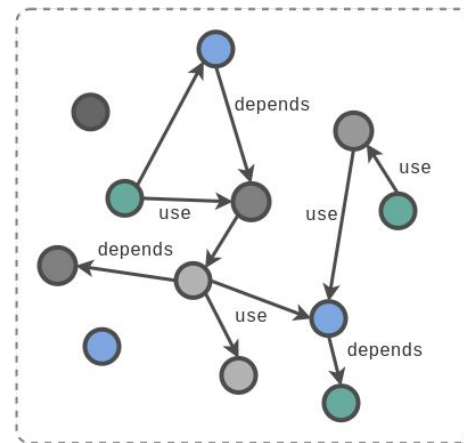
firmware

## #2 Extraction



filesystem extracted

## #3 Mapping / Correlations



Cartography

# Analysis Workflow



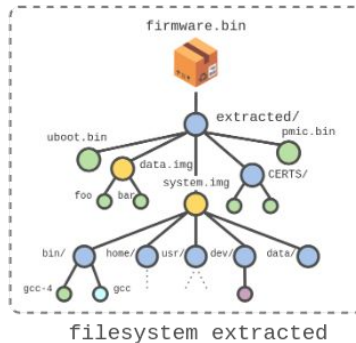
## #1 Acquisition



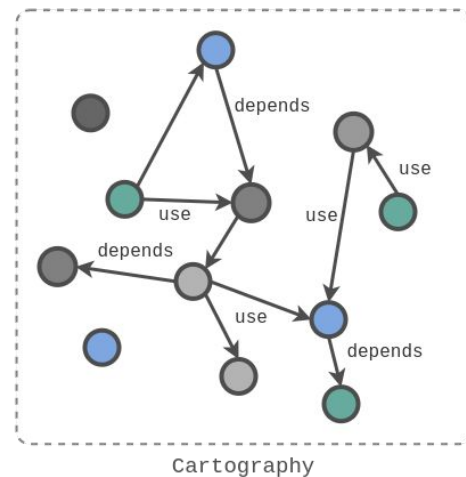
## #2 Extraction



firmware



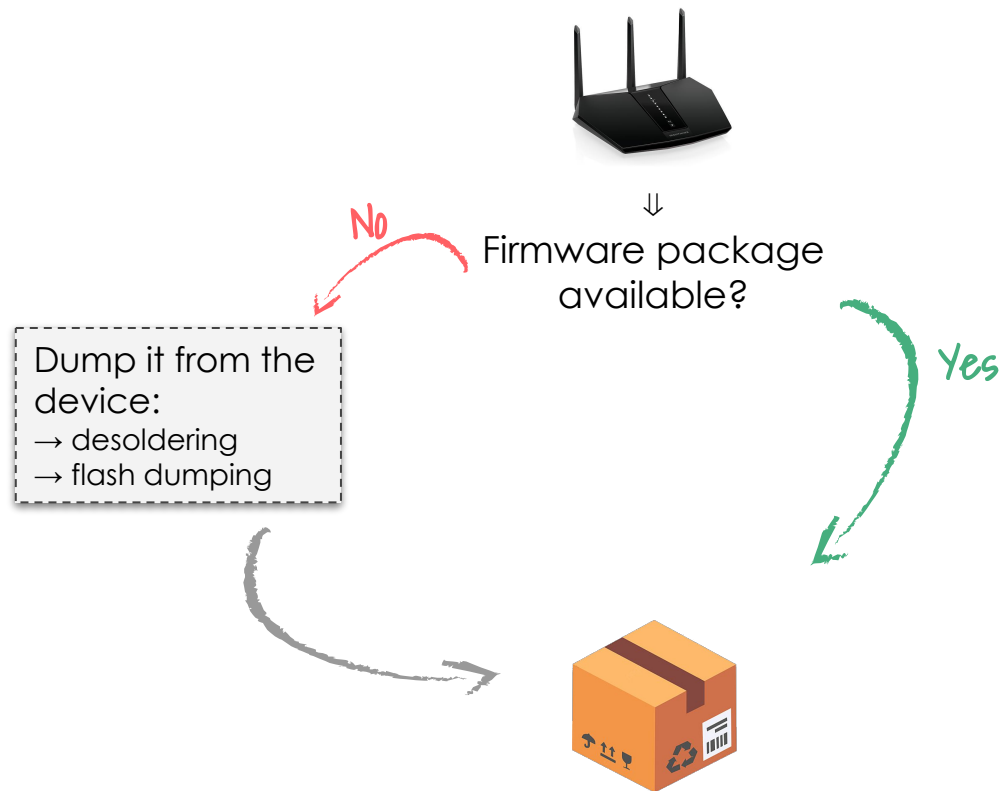
## #3 Mapping / Correlations



## #4 Security Compliance Checks



# Acquisition (#1)



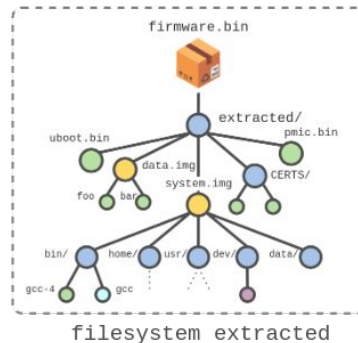
# Firmware Extraction



⇒ **Problem:** Usually custom **proprietary format** !



firmware



**Unblob** (by OneKey)

<https://unblob.org/>



**binwalk** (by ReFirmLabs)

<https://github.com/ReFirmLabs/binwalk>

# Firmware Cartography

*(with Pyrrha)*



## What ?

**Mapping** dependencies between executables, and between files as a **graph**.

## Why ?

**Saving time** & be more **efficient**.



- > Analyse executables and shared libraries dependencies (.so / .dll)
- > Resolve symbolic links
- > Resolve mount points
- > etc..

⇒ Put all this data into a **graph database** and **visualize** it.

# Introducing Numbat & Pyrrha



***Both open source !***



# Demo #1: Sourcetrail



The screenshot displays the Sourcetrail IDE interface. The top menu bar includes Project, Edit, View, History, Bookmarks, and Help. The main window is titled 'falco:outputs:abstract\_output' and shows a graph view of the symbol 'abstract\_output' in the 'falco::outputs' namespace. The graph view includes a symbol browser on the left with a search bar and a list of symbols. The central workspace shows a graph with nodes for 'abstract\_output', 'config', 'message', and 'Non-indexed Symbols'. The right pane shows the source code for 'outputs.h' with 8 references. The bottom status bar shows a log of messages and errors.

**Symbol Browser (Left):**

- Public:
  - abstract\_output (output)
  - init (reopen)
  - get\_name (cleanup)
- Protected:
  - m\_buffered (m\_json\_output)
  - m\_hostname (m\_oc)

- Derived Symbols: 4
- Non-indexed Symbols: 2

**Source Code (Right):**

```
... falco::outputs
56 //
57 // This class acts as the primary interface for implementing
58 // a Falco output class.
59 //
60
61 class abstract_output
62 {
63 public:
64     virtual ~abstract_output() {}
65
66     void init(const config& oc, bool buffered, const std::string& hostname, bool json_output)
67     {
68         m_oc = oc;
69         m_buffered = buffered;
70         m_hostname = hostname;
71         m_json_output = json_output;
72     }
73
74     // Return the output's name as per its configuration.
75     const std::string get_name() const
76     {
77         return m_oc.name;
78     }
79
80     // Output a message.
81     virtual void output(const message *msg) = 0;
82
83     // Possibly close the output and open it again.
84     virtual void reopen() {}
85
86     // Possibly flush the output.
87     virtual void cleanup() {}
88
89 protected:
90     config m_oc;
91     bool m_buffered;
92     std::string m_hostname;
93     bool m_json_output;
94 };
95
96 } // namespace outputs
97 } // namespace falco
```

**Status Bar (Bottom):**

Type	Message
1 INFO	Starting Sourcetrail 64 bit, version 2021.4.19
2 INFO	Load settings: /home/cryptocorn/.config/sourcetrail/ApplicationSettings.xml
3 INFO	Enabled console and file logging.
4 INFO	Ran Java runtime path detection, no path found.
5 INFO	Ran Maven executable path detection, no path found.
6 INFO	Loading Project: /home/cryptocorn/Quarkslab/RD/23-03-sourcetrail-custom-db/falco.scrtrprj
7 INFO	Finished Loading
8 INFO	Activate "falco::outputs:abstract_output": 1 result with 9 references in 8 files

Activate "falco::outputs:abstract\_output": 1 result with 9 references in 8 files

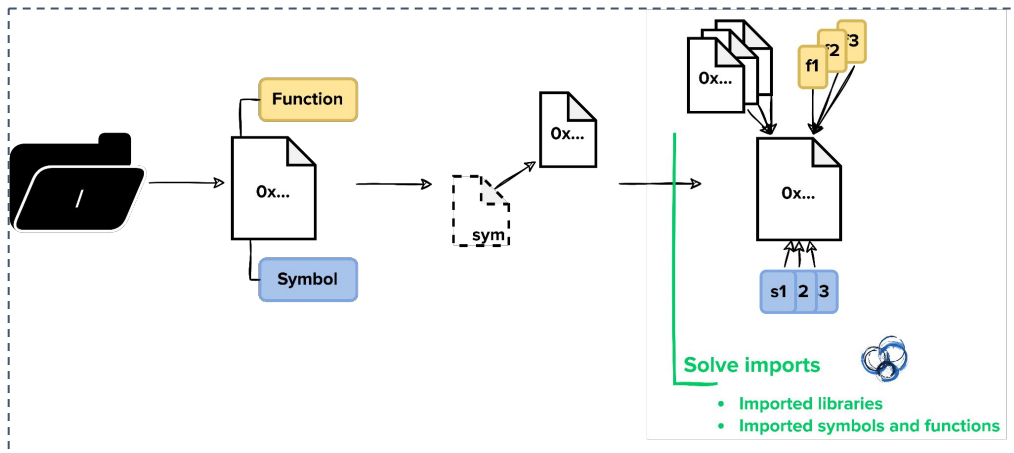
Clear Table

No IDE connected 23 errors (23 fatal)

**Pyrrha**

Quarkslab

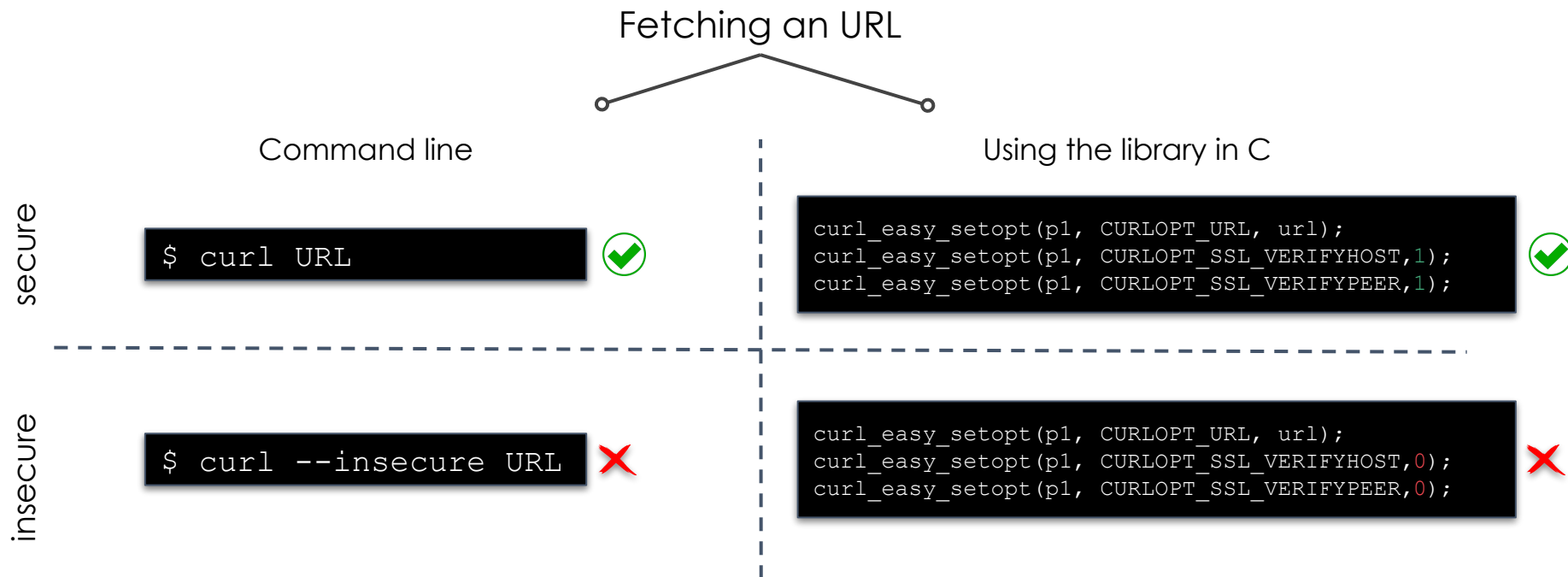
1. Scan the **root/** filesystem
2. Parse executable files
  - a. Parse **symbols**
  - b. Parse **dependencies** (*shared libraries*)
3. Resolve symbolic links
4. Create the **graph** based on imported symbols (using Numbat)



Pyrrha result on  
**Netgear RAX30**



# Firmware Update (Netgear RAX30)



`pucfu` binary checks for firmware upgrade online using the library.

Which executables are  
using `libcurl.so` ?





## Benefits:

- **Speed-up** security assessments
- **Automate** analysis and checks in client's workflows

## Checks:

- Configuration compliance
- Firmware signatures
- CVEs
- Cryptographic keys
- Additional properties
- Bill-of-materials



## Conferences

### Conference Papers / Presentations

#### 2024

- [Spyware for Rent](#) 🖥️ [NullCon'24](#)

#### 2023

- [Google Apps Script](#) 🖥️ [GreHack'23](#)
- [Dissecting the Modern Android Data Encryption Scheme](#) 🖥️ [Hardwear.io NL](#)
- [Breaking Secure Boot on the Silicon Labs Gecko platform](#) 🖥️ [Hardwear.io NL](#)
- [Breaking Secure Boot on the Silicon Labs Gecko platform](#) 🖥️ [Ekoparty 2023](#)
- [On the All UR are to be considered harmful for fun and profit is the new cool thing](#) 🖥️ [Ekoparty 2023](#)
- [Intel SGX assessment methodology](#) 🖥️ [Azure Confidential Computing 2023](#)
- [Pyrrha: navigate easily into your system binaries](#) 🖥️ [Hack.lu'23](#)
- [Fuzzing ntop](#) 🖥️ [ntopconf'23](#)
- [Emulation de périphérique USB-ETH pour l'audit IoT/Automotive](#) 🖥️ [BarbHack'](#)
- [Introduction au CarHacking Comment construire sa "Car-in-a-box"](#) 🖥️ [BarbHack'](#)
- [Map your Firmware!](#) 🖥️ [PTS'23](#)
- [For Science! - Using an Unimpressive Bug in EDK II To Do Some Fun Exploitation](#)

<https://github.com/quarkslab/conf-presentations>

## Open source

Dynamic Analysis	🐙	<b>QBDI</b>
		<b>Qtracer</b>
Symbolic Execution	🐙	<b>Triton</b>
	🐙	<b>TritonDSE</b>
Fuzzing	🐙	<b>PASTIS</b>
		<b>HF/QBDI</b>
Firmware Analysis		<b>Pandora</b>
	🐙	<b>Pyrrha</b>
	🐙	<b>QSig</b>
Diffing	🐙	<b>python-bindiff</b>
	🐙	<b>QBinDiff</b>
Static Analysis	🐙	<b>python-binexport</b>
	🐙	<b>Quokka</b>
Deobfuscation	🐙	<b>Qsynthesis</b>

<https://github.com/quarkslab/>

## Blog

The screenshot shows the Quarkslab blog interface. The top article is titled "Exploiting GLPI during a Red Team engagement" and is dated March 21, 2024. The bottom article is titled "Audit of Allbridge Core" and is dated March 19, 2024. Both articles include a brief description of the research and a list of tags.

<https://blog.quarkslab.com>

# Thank you

## Contact information:

Email:

[contact@quarkslab.com](mailto:contact@quarkslab.com)

Phone:

+33 1 58 30 81 51

Website:

[quarkslab.com](http://quarkslab.com)



@quarkslab