# Vulnerabilities research - QuarksLab

Adel Benamra Thomas Gougeon Ludovic Robin

October 27, 2016

1 Introduction

2 Find library version

3 Conclusion

# Introduction

## Context

## Fred's problems

- Libraries version
- Find other execution paths for a root cause

# Find library version automatically

## Objective

- Given a binary lib $L$ used in a program $P$
- What is the version of $L$ ?
- A database $D$ of sources of all the version is available

## Motivation

- Programs often use old libraries
- These libraries include known vulnerabilities

## Issues

- Manual investigation is not trivial (strings, calls, . . . )
- Only 2 persons in the team (who is Adel ?)

# First approaches

## Compare binaries

- Compare the binary code of $L$ with binaries generated by the sources of $D$
- Compare a signature of $L$ with a signature of $D$ (CFG ?)
- Compare the symbol list of $L$ with those of $D$

## Issues

- Binaries, CFG, and symbols depend on the architecture compilation target and the compiler version
- Binaries, CFG, and symbols depend on the source code
- How to detect if the difference comes from the patch or the compilation ?

# Second approaches

## Compare execution

- Compare traces of execution from $L$ to those of $D$
- Compare signatures of functions
- Call functions with NULL arguments

## Issues

- How to call functions of $L$ ? Sometimes there are tests
- Differentiate only major versions

# Differentiate minor versions

## Heuristics

- Compare CFG of functions
-

# Find other root paths

## Coccinelle

- Find known bugs in source code
- Analyse patterns in source
- Issue: We deal with binaries
- Useful to find other execution paths for root cause

# Conclusion

## Results

- Differentiate major versions =