Library version checking - QuarksLab

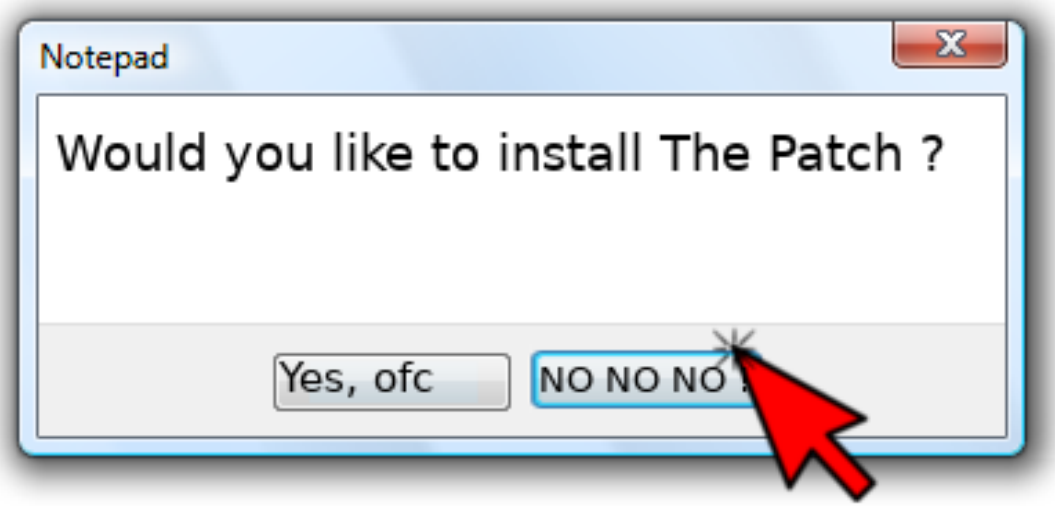
Adel Benamara Thomas Gougeon Ludovic Robin

October 27, 2016

Introduction

- Hackers look for vulnerabilities in programs ;
- Sometimes, finding one allow them to make an exploit ;
- Developers try to fix vulnerabilities, publishing a patch ;
- Users install the patch to hopefully fix the vulnerability on their version of the program.

What could go wrong ?



- The vulnerability is not necessarily patched and the old version may remain ;
- Very dangerous !

Problem

- How do we know if a library is patched ?
- How to check if we applied a patch ?

Let L be the library En entree on a une librairie binaire en .so et on va repondre a la question: Quelle est la version de cette librairie?
Du coup on nous donne un .so Et on a pleiun de sources de librairies connues
Dans la suite on va considerer que la librairie binaire a ete compilee dans la meme architecture que notre machine

Approche naive, on compile nos librairies et les compare bit par bit avec L
Probleme: L est compile avec un compilateur different et des options de ocmpilations differentes.
Les binaires sont trop differents, meme si ils font potentiellement la meme chose.

Il faut trouver des methodes plus intelligentes pour comparer les binaires.

- Linkage
- Malware signature
- Traces

Link dynamically the library

blabla

Malware signature

Analyse traces

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

Toy example

Mettre un exemple pour montrer comment ca marche cette idee

Automating

Scripts in Perl

- Generate headers from sources
- Generate so from sources
- Generate test.c

Differentiate minor versions

Heuristics

- Compare CFG of functions
-

Further ideas

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

Issues

- Adel leaves us alone
- A lot of tools do nothing

Results

- Differentiate major versions =

Organisation

Tools and communication

- GitHub
- IRC
- Audio meetings

Agility

- Separation in tasks
- Meetings every hour

It brought us..

Mind

- Inner peace
- Happiness

Technical

- Perl
- Binaries
- Compilation
- Bash