OSADL Networking Week 2020

Armijn Hemel, MSc

Tjaldur Software Governance Solutions

About Armijn

- studied computer science at Utrecht University
- coreteam gpl-violations.org 2005 – 2012
- owner Tjaldur Software Governance Solutions (since 2011)
- board member at NixOS Foundation
- technical part at OSADL audits
- not a lawyer, but I have opinions

Today's topic: scanning binary files

- why scan binaries?
  - code deliveries from upstream vendors are frequently mixed source/binary
  - some vendors only deliver binaries
  - some vendors deliver binaries and source code separately and might or might not match
- binary scanning should therefore be part of any compliance process

Deliveries from suppliers

- examples:
  - SDKs from chipset vendors
  - drivers for peripherals
  - software from ISV
- depending on how thorough the supplier is there could be only source code, only binaries, or mixed source/binary deliveries

Goals of scanning

- finding binary artifacts for which there is no complete and corresponding source code when it should be provided
- finding errors in build systems

Why find binary artifacts?

- some binaries might need source code disclosure:
  - binaries containing GPL licensed software
  - statically linked binaries that are linked with LGPL licensed C library
- some might need copyright statement copying:
  - binaries containing BSD/MIT code
- etc.

Common binary mistakes in source code releases

- binaries from a previous build (object files, complete binaries): sometimes these are from other versions than the source code.
- entire firmware releases for different devices
- binaries without source code: Android has “prebuilt” binaries to speed up builds, Yocto has a prebuilt toolchain (“uninative”)

Finding errors in build systems

- binary artifacts might hide build system errors:
  - some build systems don't rebuild source code if a binary is present: binaries are merely copied. Removing these binaries can reveal that the source code doesn't build
  - also: binary artifacts might be a different version
- “make clean” (or equivalent) before releasing a source code release should be standard!

Tools useful for scanning binaries

- common sense/experience
- standard Linux tools:
  - find + xargs + file
  - strings
  - readelf
  - hexdump
  - etc.
- binwalk
- Binary Analysis Next Generation (BANG)

Use common sense and experience

- for (embedded) Linux there is a finite number of programs in frequent use
- you don't have to start from scratch: a binary called “busybox” will almost 100% of the time contain BusyBox
- in my experience most binary file leftovers are actually object files from a previous build
- this will be your most important tool

Standard Linux tools: file

- surprisingly effective:
  - $ find -type f | xargs file
- things to watch out for in archives:
  - compressed files (tar.gz, tar.bz2, zip, RAR, etc.)
  - executables: ELF, bFLT, Java class
  - file systems: ext2/3/4, romfs, jffs2, etc.
  - “data”
- “file” uses heuristics which might be wrong at times

Standard Linux tools: strings

- “strings” is very good to extract human readable strings from binaries. If you are uncertain what a certain binary does use “strings” and a search engine
- compilation does not remove things like help texts or error messages: great for fingerprinting!
- simple trick to find C file names that might have been recorded in a binary:
  $ strings /path/to/binary | egrep -e '\.[ch]$' | sort | uniq

Standard Linux tools: readelf

- use readelf to analyze (dynamically linked) ELF binaries you don't recognize:
  - dynamically linked ELF files record variable names and function names that are needed and defined
  - defined variable names and function names are a really good indicator
  - $ readelf -W --dyn-syms /path/to/binary | grep -v UND

Standard Linux tools: hexdump

- hexdump can be very effective to see patterns in files
- $ hexdump -C /path/to/file | less
- file system boundaries and compressed/encrypted data
- in many cases it is not needed

binwalk

- binwalk (https://github.com/ReFirmLabs/binwalk) is a fairly basic firmware unpacking tool
  - support for various compression formats
  - support for some file systems
  - some support recursion
  - not seeing a lot of development anymore after a company was built around it

BANG

- BANG (https://github.com/armijnhemel/binaryanalysis-ng) is a very advanced firmware unpacking tool
  - support for unpacking and recognizing around 100 file types
  - very good support for recursion
  - (very basic) support for exporting data to for example ElasticSearch
  - currently being refactored (so please wait for some time before using)

“Hidden binaries”

- sometimes binaries can be in unexpected places
  - as hexadecimal text
  - as Intel Hex, or Motorola SREC, or similar format
- these “binaries” first need to be converted before analysis
- most extreme example found so far: 52 MiB C source code file, with a char* containing a hex converted Linux kernel + file system

Possible workflow

- A possible workflow when getting code from a supplier:
  1. find all files that are not source code, including recursing into archives
  2. analyze what binary files exist and what they are
  3. possibly remove all files that can be removed
  4. rebuild to see if the build still succeeds

Rebuild: verifying binaries match

- the best way to verify if a source code is complete and corresponding and matches the binary is to do a rebuild:
  1. copy the original binary
  2. rebuild the software
  3. compare the original binary and the binary from the rebuild (using checksums like SHA256, and tools like “strings”)
- the binaries do not have to be identical, but “equivalent”

Recommendations

- try to get source code only releases: only having to worry about source code reduces headaches
- (when doing development in house) ditch any supplier specific SDKs and use standard build environments such as Yocto, ptx-dist, buildroot, ELBE, and so on (see last year's Networking Day)
- when finding errors in a supplier SDK document these and write clean up scripts to prevent errors from being reintroduced
- do these scans at the beginning of your project and remove anything that could be a problem
- rebuild rebuild rebuild

Questions?

- There is the opportunity to ask questions at 15:45
- Please join in using the following URL: https://jitsi.osadl.org/OSADLNWWDiscussion