Inside the Binary Analysis Tool

헤멜 아마인
Armijn Hemel, MSc
Tjaldur Software Governance Solutions
armijn@tjaldur.nl

June 5, 2015

About Armijn

- owner Tjaldur Software Governance Solutions
- creator of Binary Analysis Tool (BAT)
- studied computer science at Utrecht University (The Netherlands)
- using open source software since 1994
- part of coreteam of `gpl-violations.org` from 2005-2012, gaining very detailed knowledge about enforcement practices and supply chain issues

About Tjaldur Software Governance Solutions

- tooling (BAT, source code scanning)
- research (provenance, build tracing, audit techniques, supply chain issues)
- GPL compliance engineering (Patrick McHardy cases, BusyBox cases)
- audits (for OSADL and others)

About this presentation

This presentation is to show backgrounds of and techniques behind the Binary Analysis Tool:

- why to use tools like BAT
- better understanding of what to expect from BAT
- know where strong points and weak points are in BAT
- know where BAT will go in the (near) future

Binary Analysis Tool background

BAT was started with the following goals in mind:

- demystify compliance engineering by codifying knowledge
- make it easier to have reproducable results
- create common language for binary analysis
- only analyse binaries, but draws no legal conclusions

Development started in late 2009. The first version was released in early 2010. The latest version is BAT 21.

Why analyse binaries?

- software is often supplied in binary form by vendors (on a device/CD/DVD/flash chip/download/app). Sometimes source code is supplied and sometimes it might match the binary.
- between companies (for example in a supply chain) software is often shipped as binaries
- even *inside* companies software is frequently shipped as binaries

Shipping software as source code is the *exception*.

If you pass on binary software that you get from upstream (for example 3rd party software in a product) you *have* to know what you ship, for compliance, security, and so on. This means you have to analyse the binaries.

Typical use case for Tjaldur Software Governance Solutions

Often I am approached by companies in the consumer electronics/industrial automation space that want to know what their supplier has given them:

- presence of any GPL programs (recently especially focusing on programs written by Patrick McHardy)
- presence of any GPL code in proprietary applications
- presence of security bugs

And they need to know yesterday...

What is a binary?

A “binary file” could mean several things:

- single executable or library file
- firmware of a (consumer electronics) device (containing many executables and library files)
- a random blob of data
- etcetera

When analysing binaries (any of the above) it is important to find out what is inside.

Finding out what is inside a binary: firmware

A firmware for a device (router, IP camera, tablet, IVI system, industrial controller, TV, HVAC, etc) has a structure:

- compressed files
- file systems
- plain files
- "padding" (data that is not used during execution to fill up space so different partitions are aligned correctly)

These can be found by looking for certain patterns. After that the contents can be extracted, possibly recursively (file system inside compressed file inside an installer, on a CD image), until all contents have been extracted.

Extraction steps (firmware)

1. search for known identifiers like start of a file system or start of a compressed file
2. carve possible candidate files (file systems, compressed files) out of the firmware
3. extract contents of candidate files and verify if this was done correctly
4. for every file extracted successfully from the candidate files go to 1.

After completing these steps all contents are extracted. This can be done manually, but BAT already can do this automatically for you for 35+ file systems, compression methods, media formats, and so on.

Finding out what is inside a binary: individual files (1)

After extraction individual files can be examined. For compliance and security research the most interesting files are:

- executables
- libraries

For compliance “dynamic linking interaction” is also interesting.

For security other files are also interesting:

- password files

The focus today is on executables and libraries.

Finding out what is inside a binary: individual files (2)

There are several methods that can be used to determine what is inside an executable or library:

1. decompilation of binary code into source code, followed by comparison to known source code
2. extraction of identifiers from binary code, followed by comparison of identifiers extracted from source code

The first method reveals the most information like (partial) control flow, but is more difficult to get right: information about the compiler and compiler flags needs to be known.

The second method is much more simple: information that is used (string literals, function names, etc.) can very easily be extracted using standard tools and is (largely) platform independent.

BAT uses the second method. This is also the method that many compliance engineers and license enforcers use (but manually)

Extraction of identifiers from a binary

For Java (including DEX/ODEX formats) a lot of information (including type information) is kept inside the binary file and there are tools to extract this:

- `jcf-dump` for regular Java class files
- Dedexer for Android DEX/ODEX files

For ELF files information can be extracted easily as well:

- `strings` to extract string literals (from certain ELF sections, without typing information, so comes with some false positives)
- `readelf` to extract information about functions and variables from dynamically linked files

Often hundreds, or even thousands of identifiers can be extracted.

Extraction of identifiers from source code

Identifiers in source code can be extracted using standard tools that work for source code written in many programming languages:

- `xgettext` to find extract string literals (output messages, debug messages, etc.)
- `ctags` to find function names, method names, variable names, etc.

These tools do a really good job, but miss information in some cases. In those situations using a proper source code parser would be better.

For BAT I store this information (with a lot more meta-information) in a database for easier lookups.

Advantages of using fingerprinting instead of decompilation

Fingerprinting has a few big advantages:

- straightforward: if needed it could be done manually, but it is also very well suited for automation (BAT does all the bookkeeping for you)
- platform independent: for extraction of many types of identifiers no platform specific knowledge is needed
- "good enough. . .": for most uses (detection if a particular package is used or *likely* used) it does the job very well

Drawbacks of using fingerprinting instead of decompilation

but also drawbacks:

- "... but not perfect": if there are no identifiers that can be extracted the method fails. For detection of snippets of code without identifiers it will not work well.
- fairly easy to circumvent: changing identifiers will avoid detection

In the vast majority of *current day* cases these drawbacks are not a problem: often hundreds, or thousands of identifiers can be found in binaries and matched to known source code, down to the package version level.

Defeating BAT (and why you do not want to) (1)

The techniques in BAT are quite easy to defeat:

- change string literals
- change function names/variable names

But you don't want to do this.

Defeating BAT (and why you do not want to) (2)

Circumvention is not very difficult, but it comes at a price:

- increases engineering, integration and testing costs
- circumvention could create *higher* risk for companies, because by circumventing they acknowledge they knew what is inside the software
- a company can still get caught by people who *do* use more advanced reverse engineering techniques (recent example: Allwinner Technology)
- circumvention is not for leaders, only for followers. Circumvention means playing "catch up" and not using the full potential of open source.

Circumvention could turn out to be *more* expensive than having a good software lifecycle management/compliance process. Consumer electronics is cost sensitive, so this does not happen (much).

BAT database

For the fingerprinting BAT uses a database. Tjaldur Software Governance Solutions has a database with information from over 180,000 open source packages. Information includes:

- meta information (package, version, checksums)
- string literals
- function names, method names
- variable names
- Linux kernel specific information

BAT & Protex

Very soon Black Duck will start offering BAT (plus database) as an option in Protex (certain platforms only).

For more information, please talk to Black Duck directly.

Challenges when analysing binaries

- finding structure: encryption, obfuscation, new/strange file systems
- less information available: compilers discard a lot of useful information
- internationalization: more and more code are using other languages than Western languages in identifiers. Finding out automatically in which encoding some data is is difficult.
- clones: code is copied around more and more, making it more difficult to see which software package was actually used (not unique to BAT or binary files)
- interpretation of data: because of above points results can be "fuzzy" and (manual) interpretation is time consuming

BAT future

- closing the gap between binary scanning and source code scanning
- increased focus on security
- combining (more) copyright information
- increasing quality of matches

Closing the gap between binary and source code scanning

By using information from build systems when building a binary it is possible to get a very exact view of what is used. I want to combine this with other information (licensing, copyrights, security, etcetera), and unlock this in an easy way.

Increased focus on security

Currently I am working on a project for the Dutch government to adapt BAT for security scanning. Several improvements for this are coming in BAT 22.

Ask me about more details later!

Combining (more) copyright information

At the moment BAT does not use copyright information, but this might be needed in the future:

- increased focus on copyright information/attribution in some of the German enforcement cases
- overlooked area of compliance, with a lot of (legal) unclarity

Issues:

- very difficult to correctly extract copyright information
- information about copyrights/authorship could be outside of the source code

Increasing quality of matches

Code cloning (copied code) is a problem for current BAT. For BAT 22 improvements are planned that will address this:

- automatically finding copies in the BAT database
- (slowly) moving away from package level detection to file level detection

Conclusion

In this talk you have seen:

- why to use tools like BAT
- what to expect from BAT
- where strong points and weak points are in BAT
- where BAT will go in the (near) future

Q&A and discussion

감사합니다!

- BAT: `http://www.binaryanalysis.org/`
- Tjaldur Software Governance Solutions: `http://www.tjaldur.nl/`
- Contact: `armijn@tjaldur.nl`