Binary Analysis

Armijn Hemel, MSc
Tjaldur Software Governance Solutions

October 29, 2014

About Armijn

- using Open Source software since 1994
- MSc Computer Science from Utrecht University (The Netherlands)
- created original prototype for NixOS
- core team `gpl-violations.org` from 2005 - May 2012
- owner Tjaldur Software Governance Solutions
- European coordinator Linux Defenders at Open Invention Network

Today's topic

Today I will talk about (automatically) analysing binary files, plus tell you why I do this. I will cover:

- consumer electronics supply chain issues
- some current technologies for binary analysis I (co-)developed
- new exciting projects that *you* can work on

Let's keep this interactive as much as possible. Ask questions. Make remarks.

Why analyse binaries

Contrary to what many people in the software industry want to believe most software is distributed in binary form:

- EXE, RPM, DEB
- embedded devices
- apps
- ...

Source code really is the exception!

Supply chains

Marketing and rebranding is very common in many industries:

- food industry
- soap
- consumer electronics

Know your supply chains and you can save a lot of money!

Supply chains in the electronics industry

Devices are not made in the way that many people think. Instead, a (long) supply chain of companies is responsible for a single product:

- chipset manufacturer
- original design manufacturer (ODM)
- SDK vendor
- 3rd party vendors (hardware drivers, etc.)

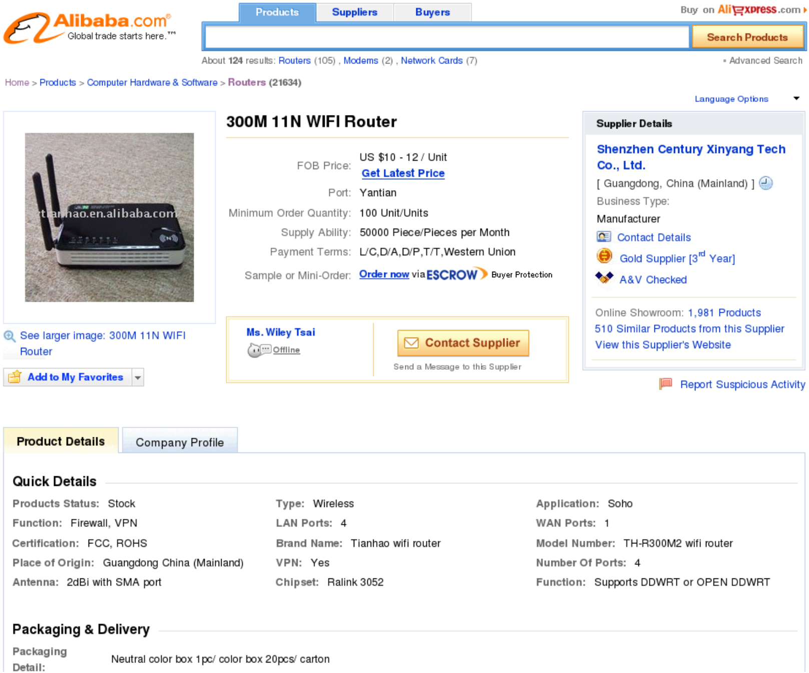
The name on the box doesn't mean anything.

Consumer electronics: the truth

Almost everything is purchased pre-fab in Asia. Making everything yourself is commercial suicide:

- extremely thin margins
- cut throat competition
- quality is less important than price
- “cowboys”

It’s like Nike: don’t do any production, just marketing and sales.



Alibaba.com® Global trade starts here.™

Products | Suppliers | Buyers

Buy on AliExpress.com

Search Products

About **124** results: Routers (105) , Modems (2) , Network Cards (7)

Advanced Search

Home > Products > Computer Hardware & Software > Routers (21634)

Language Options

tianhao.en.alibaba.com

See larger image: 300M 11N WIFI Router

Add to My Favorites

300M 11N WIFI Router

FOB Price:	US $10 - 12 / Unit Get Latest Price
Port:	Yantian
Minimum Order Quantity:	100 Unit/Units
Supply Ability:	50000 Piece/Pieces per Month
Payment Terms:	L/C,D/A,D/P,T/T,Western Union
Sample or Mini-Order:	Order now via ESCROW Buyer Protection

Ms. Wiley Tsai

Offline

Contact Supplier

Send a Message to this Supplier

Supplier Details

Shenzhen Century Xinyang Tech Co., Ltd.

[Guangdong, China (Mainland)]

Business Type:

Manufacturer

Contact Details

Gold Supplier [3rd Year]

A&V Checked

Online Showroom: 1,981 Products

510 Similar Products from this Supplier

View this Supplier's Website

Report Suspicious Activity

Product Details | Company Profile

Quick Details

Products Status:	Stock	**Type:**	Wireless	**Application:**	Soho
Function:	Firewall, VPN	**LAN Ports:**	4	**WAN Ports:**	1
Certification:	FCC, ROHS	**Brand Name:**	Tianhao wifi router	**Model Number:**	TH-R300M2 wifi router
Place of Origin:	Guangdong China (Mainland)	**VPN:**	Yes	**Number Of Ports:**	4
Antenna:	2dBi with SMA port	**Chipset:**	Ralink 3052	**Function:**	Supports DDWRT or OPEN DDWRT

Packaging & Delivery

Packaging Detail:	Neutral color box 1pc/ color box 20pcs/ carton

Problems observed in consumer electroncis

Razor thin margins and high pressure cause people to take shortcuts:

- copyrights
- security
- hardware components (out of scope)

Because more software is commoditized because of open source chipset manufacturers are providing more and more standard solutions, cutting margins of ODMs. Market pressure from customers drives prices down more.

Experiences with Asia regarding copyright

Barriers:

- copyright works differently
- cultural barriers
- NDAs were mandatory for years, now source code is demanded
- widely used open source licenses like GPLv2 are all in English
- open source code (such as GPL) is seen as “public domain”

Experiences with Asia regarding copyright (2)

- low risk getting caught, so why bother?
- “best practices” are not applied: development processes are often quite sloppy
- everything is short term
- engineers switch companies fast, especially in Taiwan
- competitors also don’t follow licenses and “save” money

Experiences with security

Security is not a feature, so there are no tests done for it.

It is simply not on their radar *at all*.

Problem: how to discover issues

We have all these devices. How are we going to find out what is inside a device and whether or not it is open source software, or is insecure?

What are *your* thoughts?

Firmware analysis

Firmwares seem like opaque blobs, but actually there is structure:

- file systems
- compressed files
- archives

First look for headers/footers, carve out file systems, unpack them, then recursively scan files that were unpacked.

Observed issues: vendor specific modifications to file systems, obfuscation/encryption

Analysing individual binaries

Given a single binary how would you try to find out:

- contents (what open source software was used)
- possible versions of open source software

Your thoughts!

Drawbacks of decompilation

- various architectures
- different compiler settings
- possibly illegal in some jurisdictions
- binaries are often stripped

Fingerprinting!

It is easier to fingerprint binaries using:

- string constants
- function/method names
- variable names

String constants

```
...
 } else {
    printf("%s %s: status is %x, should never happen\n",
           inst->prog, inst->device, status);
    status = EXIT_ERROR;
 }
 ...
```

These are not stripped by the compiler and often occur in the exact same order as in the source code. This makes it ideal for fingerprinting.

Function names/variable names

In dynamically linked ELF binaries it is possible to easily get lists of locally defined functions and variable names. Great for fingerprinting!

In Java and Dalvik binaries it is trivial to get this information.

Match with information from source code

A lot of open source code is reused unmodified, or with very few modifications. The LTSI project (Linux Foundation) showed that around 98% of the Linux kernel code in Android handsets is not modified and changes are minimal (mostly driver code).

Many other components are not changed at all. Consumer electronics mostly uses open source, and you can download & process that code easily.

Extracting string constants and other identifiers

There are great open source tools that can help extracting identifiers

- `xgettext` for string constants
- `ctags` for function names and variable names

Results might not be 100% perfect, but this works good enough.

You can store these results with other metainformation (file names, package names, version names, licenses)

Match and report

Now it is almost trivial to find out what package was used in a binary

1. extract identifiers from a binary
2. look up identifiers in a database containing identifiers extracted from source code
3. statistics, matching, reporting

This works *extremely* well: using just string constants I can often tell you which version was used.

Binary Analysis Tool

Binary Analysis Tool (or: BAT) is a lightweight tool under an open source license that automates binary analysis.

- demystify binary analysis by codifying knowledge
- make it easier to have reproducable results
- common language for binary analysis

BAT is a generic framework for binary analysis. Until now focus of BAT was primarily on software license compliance.

BAT has been in development since late 2009 and getting reasonably good.

Funding came from Linux Foundation, NLnet Foundation, Cisco, as well as my customers.

Binary Analysis Tool result demo

Let's look at results of one scan of a (crappy) consumer electronics router.

Combining fingerprinting with metadata

Right now I mostly deal with license compliance issues, but there are other applications as well.

Discovering security issues is by far the most important one (next talk, after the break).

BAT work that needs to be done

BAT is not perfect and there is work that needs to be done:

- more file system support (UBIFS)
- better GUI
- support for QNX ELF files
- detection of compiler optimization options
- detecting if "broken `sstrip`" was used

and there are many more research ideas I have. If you are still looking for RP1 or RP2 tasks, talk to me!

Questions?

Contact

- armijn@tjaldur.nl
- http://www.tjaldur.nl/
- Binary Analysis Tool: http://www.binaryanalysis.org/